RECOMMENDATION SYSTEM FOR DATING PROJECTS

by Oleksandr Mostovyi

A Capstone Project

Presented in Partial Fulfillment of the Requirements for the Degree Master

American University Kyiv

2024

APPROVED BY:

Oleksandra Antoniouk, Ph.D., Dr.Habil

Abstract

The main functionality for all dating projects is to elevate user experience through the provision of potential matches based on a variety of factors. This project is dedicated to evaluating and comparing the performance of diverse recommendation algorithms within the unique context of dating platforms.

As a result an environment consisting of recommendation system and integration into search system of dating application is provided. The evaluation metrics include but are not limited to accuracy, precision, recall, and user satisfaction. By systematically testing these algorithms in controlled scenarios, the research seeks to identify specific performance strengths and weaknesses of each algorithm.

Recommendation system is implemented using Python programming language and Flask framework and it uses dataset of profile from dating project. The research also explores the impact of diverse user behaviors and preferences on the recommendation algorithms, providing insights into the adaptability and robustness of each approach.

Through a comprehensive analysis, this research aims to contribute valuable insights for dating projects seeking to enhance their recommendation systems. The findings will aid in the informed selection of recommendation algorithms tailored to the specific requirements and dynamics of dating platforms, ultimately improving user experience and satisfaction.

Acknowledgements

I want extend my sincere thanks to my project advisor, Prof. Dr. Oleksandra Antoniouk, for her unwavering guidance, invaluable support, and mentorship throughout the duration of this research. Her expertise, insightful feedback, and continual encouragement have played a pivotal role in shaping the trajectory and quality of this paper.

I also wish to convey my sincere thanks to my university teachers for their dedicated efforts in imparting valuable knowledge. Their teachings have laid a robust foundation for my academic journey and have significantly contributed to the development of this paper.

CONTENTS

Lists of Abbreviations

CMT	Cascading Metric Tree
CCR	Completed Connection Rate
RRS	Reciprocal Recommender System
MTRS	Matching Theory-Based Recommender System
IPFP	Iterative Proportional Fitting Procedure
TU	Transferable utility
OLS	ordinary least squares
kNN-CF	k-Nearest Neighbors Collaborative Filtering

Chapter 1
Introduction

1.1 Motivation

Recommendation systems is engine that uses deep learning concepts and algorithms that provide personalized suggestions. The main functionality of dating projects is to provide system that will provide user profiles based on preferences of user, the effective functioning of recommendation systems stands as a cornerstone for user engagement and satisfaction. Recommendation systems strive to efficiently allocate potential matches among the available user profiles, thereby enhancing the overall performance-based characteristics of the dating platform.

The significance of this research lies in the critical role that recommendation systems play in shaping user experiences within dating projects. The ability to intelligently match individuals based on a variety of factors is central to the success and competitiveness of dating platforms. Understanding and optimizing the underlying algorithms is, therefore, imperative for the continual improvement of these systems.

This research focuses on recommendation systems within dating platforms, delving into algorithmic performance, user satisfaction, and the dynamic interplay of factors shaping successful matchmaking. The scope encompasses a range of recommendation algorithms, considering their adaptability to diverse user profiles and preferences.

1.2 Structure

- In Chapter 1, we delve into the choice of programming language and machine learning algorithms for the thesis and explore alternative options.
- Chapter 2 provides a description of different machine learning models for recommendation system, emphasizing their individual merits and limitations.

- Chapter 3 systematically examines relevant literature, covering preferred machine learning models and methodologies utilized in comparable performance assessments, and the configurations specific to each.
- In Chapter 4, we detail architecture of recommendation system, basic presentation of workflow and selection testing tools.
- Chapter 5 provide comprehensive elucidation and comparison of the results derived from the conducted tests are presented, providing insights into potential implications.
- Chapter 6 is the concluding section, summarizing the findings and delving into a discussion on prospective avenues and directions for future research.

1.3 Implementation Technology

In current research I used several technologies that helped to develop recommendation system and research which machine learning algorithms is the best approach for matching users together.

For programming language I used Python, as one of the best programming languages for machine learning and have great support for different frameworks for training models that helped to develop recommendation system. In my research, I used FastApi framework (1) that provided support for authorization, analysis of income parameters and support Representational State Transfer between services.

For this research, Kubernetes serves as a pivotal component, providing support for auto-scaling rules across various services within the application. The functionality of Kubernetes in this context involves dynamically adjusting the number of instances of each service based on predefined auto-scaling rules. This ensures optimal resource utilization and responsiveness to varying workloads.

Kubernetes enables efficient scaling of resources for training data processing. As the training workload increases, Kubernetes can dynamically allocate additional resources to accelerate the training process.

Chapter 2

Background

A recommendation system, also known as a recommender system, is a type of software application or algorithm designed to provide personalized suggestions or recommendations to users. The goal of a recommendation system in this project is to develop system that support several algorithms to determine similarity scores between users and provide a list of matched users based on their ranking.

Checking compatibility between individuals can be a challenging endeavor. Existing dating applications, such as Tinder [2], often gather limited user information. Instead, they present a set of profiles within a specified geographic radius that meet the user's criteria, typically age range and gender. Users then make binary choices (Yes or No) regarding these profiles, leading to a "match" if there is mutual interest, allowing the matched users to initiate conversations [1]. While some platforms, like Hinge [3], use algorithms based on user interactions to pair individuals with similar preferences [4,5], both methods primarily focus on appearance and may not consider users' profiles comprehensively, potentially hindering the formation of successful relationships.

This research addresses the significance of measuring compatibility in dating applications. We propose an algorithm that mirrors some existing approaches but introduces a novel aspect. Users are encouraged to provide a ranked list of their interests, such as favorite movies or sports. The algorithm then assesses compatibility based on these ranked lists, facilitating the matching of individuals with similar preferences. This approach aims to enhance the depth of compatibility assessment beyond surface-level attributes, potentially contributing to more successful relationships.

The algorithms scrutinized in this chapter represent the cutting edge in recommendation system technology. They are evaluated based on their ability to deliver personalized content to users effectively and the robustness of their predictive accuracy. Through the lens of these measures, the chapter aims to highlight the

strengths and limitations of current recommendation algorithms and set the stage for future advancements in this field.

Overview of the Algorithms for Recommendation System

2.1 K-nearest neighbors algorithm

KNN stands for k-nearest neighbors, and it is a simple and widely used algorithm in machine learning for both classification and regression tasks. It is a type of instance-based learning, or lazy learning, where the model doesn't explicitly learn a mapping from inputs to outputs during training. Instead, it memorizes the training instances and makes predictions based on their similarity to new, unseen instances.

KNN checks the classes of a chosen number of training data samples which surround a test data sample, so as to make a prediction on which class the test data sample belongs to. The symbol k denotes the number of the nearest data samples, i.e. the neighbors.

A KNN algorithm with k = 5 is illustrated in Fig. 1, where the data samples are two-dimensional, and there are three classes in total.

.

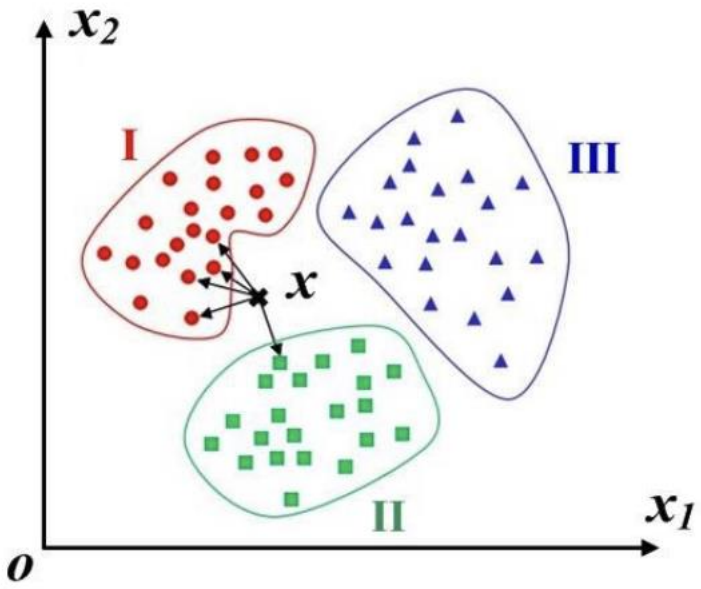


Fig. 1. Illustration of KNN algorithm (k = 5).

A KNN algorithm goes through as follows:

(1) Choose the value of k, i.e. the number of nearest data samples. Usually, k is assigned an odd integer value and lower than 20. For instance, if k is set to 5, the algorithm examines the classes of the top 5 closest training samples to the test sample.

(2) Calculate the distances between a test sample and the training samples. The training set T, which contains multiple samples, is denoted as

$$T = \{(\boldsymbol{x}_i, y_i), i = 1,2,\cdots,N\}$$

where N is the number of training samples.

In this context, x_i is a training sample, which can also be considered as a feature vector, expressed as

$$\boldsymbol{x}_i = \left(x_i^{(1)}, x_i^{(2)}, \cdots\cdots, x_i^{(n)}\right)$$

where n is the number of degrees of freedom of the feature vector.

Besides, y_i is the class symbol, representing which class xi pertains to, expressed as

$$y_i \in \{c_i, i = 1,2,\cdots\cdots, M\}$$

here M is the total number of classes.

Assume the test sample is denoted as x_t and can be represented as:

$$\boldsymbol{x}_t = \left(x_t^{(1)}, x_t^{(2)}, \cdots\cdots, x_t^{(n)}\right)$$

The distance between the test sample $x_t$and a training sample x_i can be calculated by Euclidean distance formula

$$d_{t,i} = \sqrt{\sum_{j=1}^{n}\left(x_t^{(j)} - x_i^{(j)}\right)^2}\,(i = 1,2,\cdots\cdots, N)$$

where $d_{t,i}$ denotes the distance.

The distances are then added into an array.

(3) Sort the calculated distance values in ascending order, and we have a sorted distance array expressed as

$$D = [\underbrace{d_{t,1}, d_{t,2}, \cdots, d_{t,k}}_{k} \cdots\cdots d_{t,N}]$$

where k is the assigned value of the number of nearest neighbors, and N is the number of training samples.

(4) Choose the top k elements from the sorted distance array to form a k-element array, noted as

$$D_k = [d_{t,1}, d_{t,2}, \cdots, d_{t,k}]$$

Now acquire the classes of the top k elements, yielding an array denoted as

$$C_k = [c_1, c_2, \cdots, c_k]$$

where c_k corresponds to a certain class.

(5) Return a prediction on the test sample. We can assign a predicted class to the test sample based on the most frequent class of these k closet training samples, expressed as

$$y_t = freq(C_k)$$

where y_t is the predicted class to which the test sample belongs, and $freq(C_k)$ is a function that returns the most frequent element in the array C_k, where C_k corresponds to a certain class for the k closest training samples.

2.2 Support Vector Machine

Support Vector Machines (SVMs) constitute a set of supervised learning techniques widely employed for both classification and regression tasks [9]. These methods are categorized within the realm of generalized linear classification algorithms. Notably, SVMs exhibit a distinctive characteristic: they simultaneously endeavor to minimize empirical classification errors while striving to maximize what is known as the geometric margin. This property has earned SVMs the moniker of "Maximum Margin Classifiers." SVMs are rooted in the principle of Structural Risk Minimization (SRM), which guides their operation. A key aspect of SVMs is their capacity to transform input vectors into a higher-dimensional space, where a hyperplane is constructed to optimally separate data points. In this context, two parallel hyperplanes are established on either side of the data-separating hyperplane. The ultimate goal is to identify the hyperplane that maximizes the distance between these parallel counterparts. It is presupposed that the larger this margin or separation between the parallel hyperplanes, the more proficient the classifier's generalization performance will be [10]. The SVM framework operates based on data points typically expressed in the form:

$$x_i = (x_{i1}\ , x_{i2}\ \dots\ x_{in}\ )$$

Here, x_i represents the $i-th$ data point, and $(x_{i1}\ , x_{i2}\ \dots\ x_{in}\ )$ are the feature values of this data point. Each feature x_{ij} corresponds to a dimension

in the feature space. For example, in a two-dimensional space, a data point might be represented as (x_{i1} , x_{i2}).

This methodology allows SVMs to effectively discern and classify patterns within the data, making them a valuable tool in machine learning and pattern recognition.

$$\{(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)\}$$

Where $y_n = 1/-1$, a constant denoting the class to which that point $x_n$belongs. In this context, n represents the total number of samples under consideration. Each sample, denoted as x_n is characterized as a p-dimensional vector in the real number space. The scaling process is crucial in this setting, primarily to mitigate the impact of variables (or attributes) that exhibit substantial variance. This approach is pivotal for maintaining the integrity and accuracy of the data analysis. The training data can be effectively conceptualized through the application of a dividing (or separating) hyperplane, which is defined in a specific mathematical format to facilitate this process and which takes the form:

$$w \cdot x + b - o = 0,$$

where b is scalar and w is p-dimensional vector. The vector w is oriented perpendicular to the separating hyperplane. Also o represents an offset or bias term that is introduced to increase the margin. It effectively moves the hyperplane parallel to itself, providing more flexibility in finding the maximum margin between the two parallel hyperplanes. Adding the offset parameter b enables us to increase the margin. Without b, the hyperplane is compelled to pass through the origin, thereby limiting the range of possible solutions. Our primary focus is on maximizing the margin within Support Vector Machines (SVMs) and its relationship to the concept of parallel hyperplanes. Parallel hyperplanes can be described by equation

$$w \cdot x + b = 1$$

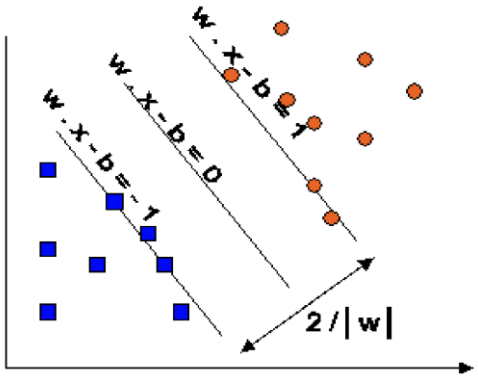
$$w \cdot x + b = -1.$$

When dealing with linearly separable training data, it is possible to strategically position hyperplanes in such a way that no data points reside between them, subsequently striving to maximize the inter-hyperplane distance. Geometrically, this distance can be computed as 2 divided by the magnitude of the weight vector w, denoted as $|w|$.

$$w \cdot x_i - b \geq 1 \text{ or } w \cdot x_i - b \leq -1$$

This can be written as

$$y_i(w \cdot x_i - b) \geq 1, 1 \leq i \leq n$$



FA maximum margin hyperplane for an SVM trained using samples from two distinct classes.

Support Vectors (SVs) are the samples located on the hyperplanes. The separating hyperplane with the greatest margin is determined by $M = 2 / |w|$. Samples along the hyperplanes are called Support Vectors (SVs). This hyperplane is characterized by its proximity to the training data points, known as support vectors, which adhere to the condition:

$$y_j[w^T \cdot x_j + b] = 1, i = 1$$

The Optimal Canonical Hyperplane (OCH) is a type of canonical hyperplane characterized by having the largest possible margin. For all data points, the OCH needs to meet the following conditions:

$$y_i[w^T \cdot x_i + b] \geq 1; i = 1,2 \ldots N$$

where N is a number of training data point. In order to find the optimal separating hyperplane having a maximum margin, a learning machine should minimize $|w|2$ subject to the inequality constraints Where N represents the total number of training data points. To determine the optimal separating hyperplane with the maximum margin, a learning machine must aim to minimize $|w|2$ while adhering to the specified inequality constraints.

$$y_i[w^T \cdot x_i + b] \geq 1;\ i = 1,2 \ldots 1$$

This optimization problem solved by the saddle points of the Lagrange's Function

$$L_P = L_{(w,b,\alpha)} = ½ \ \| \text{w} \|^2 - \sum_{i=1}^{1} \alpha_i \left(y_i \left(w^T x_i + b \right) - 1 \right)$$

$$= 1/2 w^T w - \sum_{i=1}^{1} \alpha C_i (y_i (w^T x_i + b) - 1)$$

Where α_i is a Lagranges multiplier. The search for an optimal saddle point is necessary because Lagrange's Function must be minimized with respect to w and b and has to be maximized with respect to nonnegative α_i ($\alpha_i \geq 0$), where α_i represents a Lagrange multiplier.

$$\frac{\partial L}{\partial w_0} = 0$$

$$w_0 = \sum_{i=1}^{l} \alpha_i y_i x_i$$

And

$$\frac{\partial L}{\partial b_0} = 0$$

$$\sum_{i=1}^{l} \alpha_i y_i = 0$$

By substituting the expressions for w_0 and b_0 derived earlier into the relevant equation, we transition from the primal form to the dual form.

$$L_d(\boldsymbol{\alpha}) = \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} \alpha_i \alpha_j y_i y_j x_i^T x_j$$

To identify the optimal hyperplane, it is imperative to maximize a dual Lagrangian denoted as L_d. This maximization is conducted while ensuring that the Lagrange multipliers (α_i) remain within the nonnegative quadrant and adhere to a set of equality constraints as follows.

$$\alpha_i \geq 0 \; for \;\; i = 1,2, \ldots, l$$

$$\sum_{i=1}^{1} \alpha_i y_i = 0$$

Note that the dual Lagrangian (L_d) is expressed in terms of training data and depends only on the scalar products of input patterns ($x_i^T x_j$) . More detailed information on SVM can be found in [9, 10].

Training vectors x_i are transformed into a higher-dimensional space, potentially infinite, via the function Φ. In this expanded space, SVM seeks a linear separating hyperplane with the maximal margin. The constant C > 0 serves as the penalty parameter for the error term.

Furthermore, $K(\boldsymbol{x}_i, \boldsymbol{x}_j) \equiv \Phi(\boldsymbol{x}_i)^T \Phi(\boldsymbol{x}_j)$ is called the kernel function [10]. Given the variety of kernel functions available in SVM, selecting an appropriate one remains a research question. Nonetheless, certain kernel functions have gained popularity for general use [10, 11]. The formulations for these commonly used kernel functions are as follows:

1. Linear Kernel:

$$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \boldsymbol{x}_i^T \boldsymbol{x}_j$$

2. Polynomial Kernel:

$$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = (\gamma \boldsymbol{x}_i^T \boldsymbol{x}_j + r)^d, \gamma > 0,$$

where γ is a scaling parameter, r is a coefficient, and d is the degree of the polynomial.

3. Radial Basis Function (RBF) Kernel:

$$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = exp(-\gamma \parallel \boldsymbol{x}_i - \boldsymbol{x}_j \parallel^2), \gamma > 0.$$

Here γ is a scaling parameter controlling the shape of the kernel.

4. Sigmoid Kernel:

$$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = tanh(\gamma \boldsymbol{x}_i^T \boldsymbol{x}_j + r)$$

Here, γ is a scaling parameter, and r is a coefficient. These parameters, γ, r and d are integral to the kernel's function. Among the prevalent kernel functions, the Radial Basis Function (RBF) kernel is particularly noteworthy for its advantages, as outlined in reference [10]:

- **Nonlinear Mapping**: The RBF kernel effectively maps samples into a higher-dimensional space, which is a significant divergence from the linear kernel's approach.
- **Reduced Hyper-Parameters**: In comparison to the polynomial kernel, the RBF kernel requires fewer hyper-parameters, simplifying the model configuration.

- **Numerical Stability**: The RBF kernel typically encounters fewer numerical challenges, enhancing its computational feasibility. Model selection is also an important issue in SVM.

Recently, SVM have shown good performance in data classification, see e.g. [23]. The success of this model is intrinsically linked to the adjustment of various parameters. These parameters play a pivotal role in influencing the generalization error of the model. The balance achieved through this parameter tuning is crucial for optimizing the model's performance and ensuring its successful application in diverse scenarios. Referring to this parameter tuning process as model selection if you use the linear SVM, you only need to tune the cost parameter C. The linear SVM are often applied to linearly separable problems.

Many problems are non-linearly separable. For example, Satellite data and Shuttle data are not linearly separable [31]. In classification tasks using SVM, the selection of the cost parameter (C) and kernel parameters is crucial for effective nonlinear problem-solving. (C) and kernel parameters (γ, d) [4, 5]. Usually use the grid-search method in cross validation to select the best parameter set. Then apply this parameter set to the training dataset and then get the classifier. After that, use the classifier to classify the testing dataset to get the generalization accuracy.

2.3 Collaborative Filtering

Collaborative Filtering (CF) represents a pivotal technique in the realm of recommender systems, exhibiting two distinct interpretations: a traditional, broad perspective and a more contemporary, specific one. In its modern, narrower definition, CF operates as a mechanism for automating predictions about a user's interests through the aggregation of preference or taste data from a multitude of users. This technique operates under the principle that if one user (say, User A) shares similar views with another user (User B) on one topic, User A is more inclined to hold User B's viewpoint on another subject compared to a viewpoint from an arbitrary individual. For instance, in television programming recommendations, CF can predict

a user's potential show preferences based on their known likes or dislikes. These predictions are uniquely tailored to each user, yet they harness information collected from a broad user base. This method contrasts sharply with simpler, non-specific approaches like assigning average scores to items based solely on their popularity or vote count.

On a broader scale, CF encompasses the technique of sifting through information or identifying patterns by leveraging collective efforts from diverse agents, perspectives, and data sources. It finds application in a vast array of fields, particularly those involving extensive datasets. CF methodologies have been effectively utilized across various domains, including but not limited to environmental monitoring (like mineral exploration or extensive sensor-based ecological assessments), financial sectors (integrating diverse financial data sources by financial institutions), and digital commerce or web-based platforms focusing on user data analysis. While the subsequent discussion primarily concentrates on CF as it pertains to user data, it is important to note that many of the underlying strategies and techniques are equally applicable to other major areas of use.

Collaborative Filtering (CF) techniques, recognized for their ability to process explicit user feedback, have been a focus in the field for an extended period. These techniques stand in contrast to content-based methods, as highlighted in sources [12, 13]. Among the various CF models, the k-nearest neighbor (kNN-CF) [15-16] stands out for its simplicity. This model operates by evaluating similarities between users or items, often using measures such as the Pearson correlation, which are typically fine-tuned through experimental analysis on validation datasets.

More recently, Matrix Factorization (MF) methods [24] have gained prominence. These methods are now widely regarded as the most advanced single-model approach in the realm of CF. MF is generally favored over kNN-CF due to its higher efficiency and effectiveness. This is largely attributed to its ability to identify latent features inherent in user-item interactions, features that are not easily discernible in other models. However, a notable limitation of MF is its propensity for

overfitting data, leading to the development of various extensions to address this issue, as mentioned in references [14, 15].

Understanding that the aforementioned Collaborative Filtering (CF) techniques are primarily suited for scenarios where users provide explicit feedback is essential. Yet, in numerous real-world applications, like course recommendation systems, the feedback is often implicit, a situation extensively examined in reference [19]. This scenario poses a distinct challenge, calling for the development of versatile CF models capable of accurately interpreting and leveraging implicit feedback.

The memory-based method in Collaborative Filtering employs user ratings to ascertain the likeness between users or items. This approach includes well-known techniques like neighborhood-based CF and top-N recommendations, which can be either item-based or user-based. Specifically, in user-based strategies, the rating that a user (let's call them 'u') assigns to an item ('i') is determined by compiling the ratings given to that item by other users who are deemed similar to user u.

$$r_{u,i} = aggr_{u' \in U} r_{u',i}$$

Where *U* denotes the set of top *N* users that are most similar to user *u* who rated item *i*. Some examples of the aggregation function include:

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

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

where k is a normalizing factor defined as $k = 1/\sum_{u' \in U} |simil(u, u')|$

$$r_{u,i} = \overline{r_u} + k \sum_{u' \in U} simil\left(u, u'\right)\left(r_{u',i} - \overline{r_{u'}}\right)$$

where $\bar{r}_u$ is the average rating of user *u* for all the items rated by *u*.

The neighborhood-based algorithm focuses on determining the similarity between either two users or two items. It then generates predictions for a user by computing a weighted average of all relevant ratings. An integral component of this method is the calculation of similarity between items or users. To achieve this, various

metrics are employed, including Pearson correlation [25], and vector cosine-based similarity [26].

In the domain of user-based top-N recommendation algorithms, the core methodology involves utilizing a vector model predicated on similarity measures to ascertain the *k* users that exhibit the closest neighbor to a specific and active user. Subsequent to the identification of these *k* nearest users, the algorithm proceeds to their individual user-item matrixes. This method is particularly notable for its adeptness at implementing the nearest neighbor search algorithm in a linear time frame, thereby enhancing the efficiency of the recommendation process.

This approach offers several notable advantages, crucial in the context of recommendation systems. Firstly, it provides a high degree of explainability regarding the results, a key attribute for user comprehension and trust in recommendation systems. Secondly, the method allows for straightforward creation and usage, making it accessible and user-friendly. Thirdly, it adeptly accommodates new data, ensuring the system remains current and relevant. Additionally, the approach is characterized by its content-independence with respect to the items recommended, allowing for a wide applicability across various types of content. Finally, it exhibits commendable scalability in scenarios involving co-rated items, ensuring efficiency and effectiveness even as the dataset grows.

This methodology have several limitations. A primary concern is the observed decline in performance in scenarios of data sparsity, a situation commonly encountered in web-related items. This issue poses a significant challenge to the scalability of the approach, especially when dealing with large datasets. Furthermore, despite its efficiency in integrating new users, attributable to its reliance on a specific data structure, the method faces complexities in accommodating new items. This is due to the usual dependency on a predetermined vector space for item representation. Consequently, the introduction of new items necessitates not only their inclusion but also the re-insertion of all existing elements within the structure, thereby complicating the process and potentially impacting efficiency.

In evaluating the similarity distances between features scaled within a model, a variety of techniques were implemented to facilitate accurate and meaningful comparisons. To begin with, feature normalization was conducted to bring all variables to a common scale, thereby mitigating the risk of any individual feature disproportionately influencing the distance calculation owing to its range.

Subsequent to normalization, multiple distance metrics were utilized to quantify the similarity between feature vectors. This approach ensures that the similarity assessment is robust and reflects the underlying structure of the data. The metrics selected for this task are recognized for their ability to capture the essence of feature similarity in high-dimensional spaces, which is pivotal in the context of sophisticated modeling and predictive analysis.

Chapter 3

This chapter systematically reviews the existing literature on machine learning models and methodologies that are commonly employed in performance evaluations. It delves into the machine learning techniques that are preferred in the context of similar assessments and explores the specific configurations pertinent to each model. The examination encompasses a comprehensive range of scholarly articles and empirical studies to encapsulate the current state of research within this domain.

3.1 Metric Search for Rank List Compatibility Matching with Applications

The research proposes a new dating matching algorithm that uses the Kendall tau distance to measure the similarity between users based on their rankings for items in a list (e.g., favorite sports, music, etc.). The goal is to match people with similar interests to increase the likelihood of successful relationships. The algorithm applies a tree-based searching structure called the Cascading Metric Tree (CMT) to efficiently search for users within a given radius of a target user.

The Kendall tau distance measures the number of inversions or adjacent swaps needed to transform one list into another. The larger the distance, the less compatible the two individuals are. The paper [27] shows that the CMT algorithm efficiently searches for the best matching people for a user within a practical time, given reasonable parameters.

The paper [27] discusses the scaling of the search structure with different parameters such as list length, population size, and query radius. It demonstrates that the algorithm performs well on larger population datasets and scales sublinearly with respect to the population size. However, the search structure scales poorly with respect to the length of the ranked list.

Remark that while the algorithm efficiently queries similar individuals, further research is needed to establish whether the matching produced by the new method leads to successful relationships.

3.2 Collaborative Filtering for people-to-people recommendation in online dating: Data analysis and user trial

The article [28] provides in-depth insights into various methodologies and frameworks for enhancing online dating services.

Profile Matching: This technique involves matching potential partners by comparing their profiles. The matching criteria incorporate demographic (e.g., age, gender, location) and psychographic data (interests, values, lifestyle choices), aiming to find compatibility based on these parameters.

Interaction-Based Collaborative Filtering (CF): Unlike traditional CF methods typically used in product recommendation systems, this approach in the context of online dating emphasizes the interactions and similarities between users. It focuses particularly on fostering successful reciprocal interactions, which are vital in the dating domain.

Decision Tree Critic: A sophisticated two-phase recommendation process, it first generates candidate matches using CF and then applies a decision tree model to these candidates. The decision tree is constructed from user interaction data. This method is designed to counter the issue of certain users being frequently recommended (over-recommendation), thereby ensuring a more balanced and fair distribution of recommendations.

Evaluation Framework: The study discusses various metrics such as success rate improvement and recall. These metrics are crucial in evaluating the effectiveness of

the recommendation methods, ensuring they meet the desired goals in real-world applications.

Data Analysis: The paper delves into analyzing user behavior and interactions on a dating site. This analysis is critical for developing and refining the recommendation methods, as it provides empirical insights into user preferences and patterns.

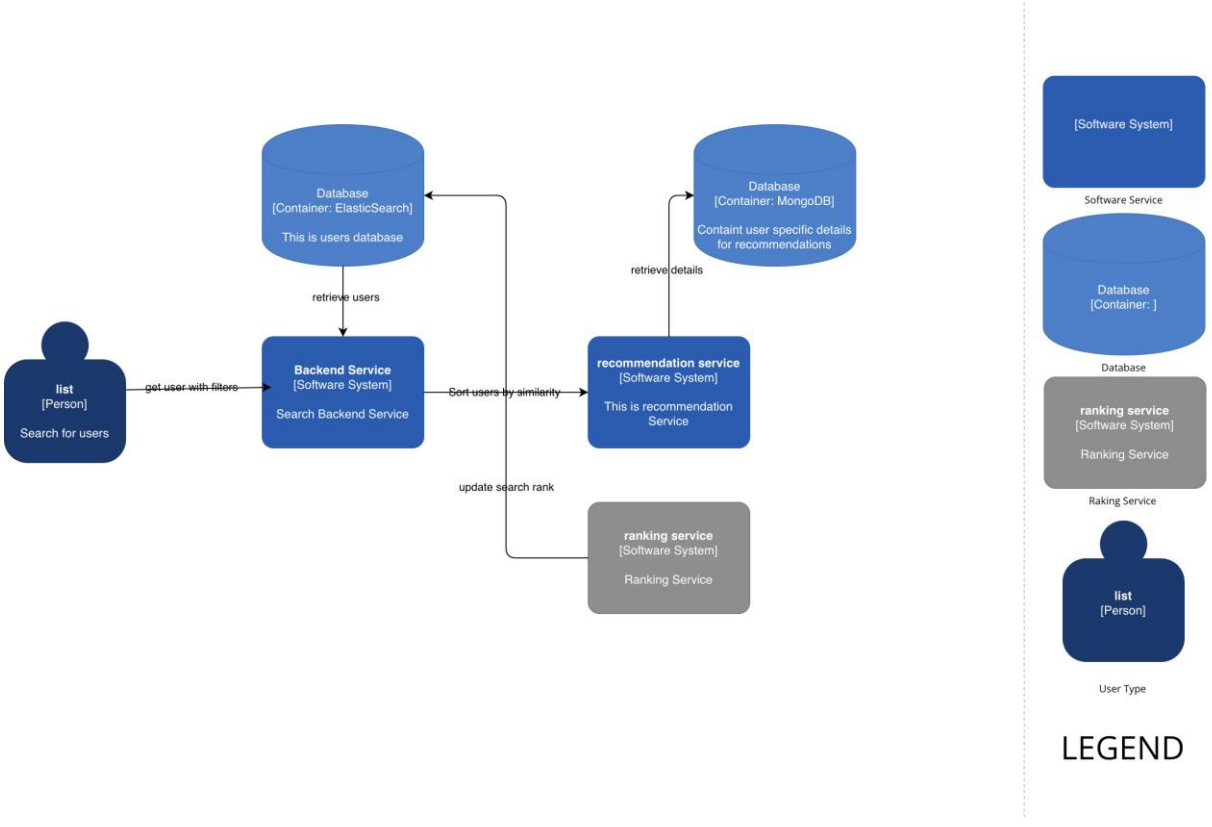
User Trial: The document details a practical trial of these methods, assessing their real-world efficacy in an online dating environment. This trial is key to understanding how theoretical models perform when applied to actual user interactions.

The study critically examines the juxtaposition of CF and profile matching methods, particularly addressing the unique challenges in online dating like the necessity for mutual interest and the imbalances in user attractiveness and activity. This comprehensive approach is designed to enhance the overall effectiveness and user satisfaction in online dating platforms.

Chapter 4

This chapter provides a full view of the recommendation system's architecture, deployment structure and testing tools. It focuses on the algorithms driving the system, the flow of data, and strategies to enhance user engagement and satisfaction. The system, embedded within an app, relies on a combination of software services and databases to analyze user similarities. The introduction also emphasizes the importance of selection testing methodologies, which are vital for evaluating the system's accuracy and overall performance.

4.1 Recommendation system's architecture

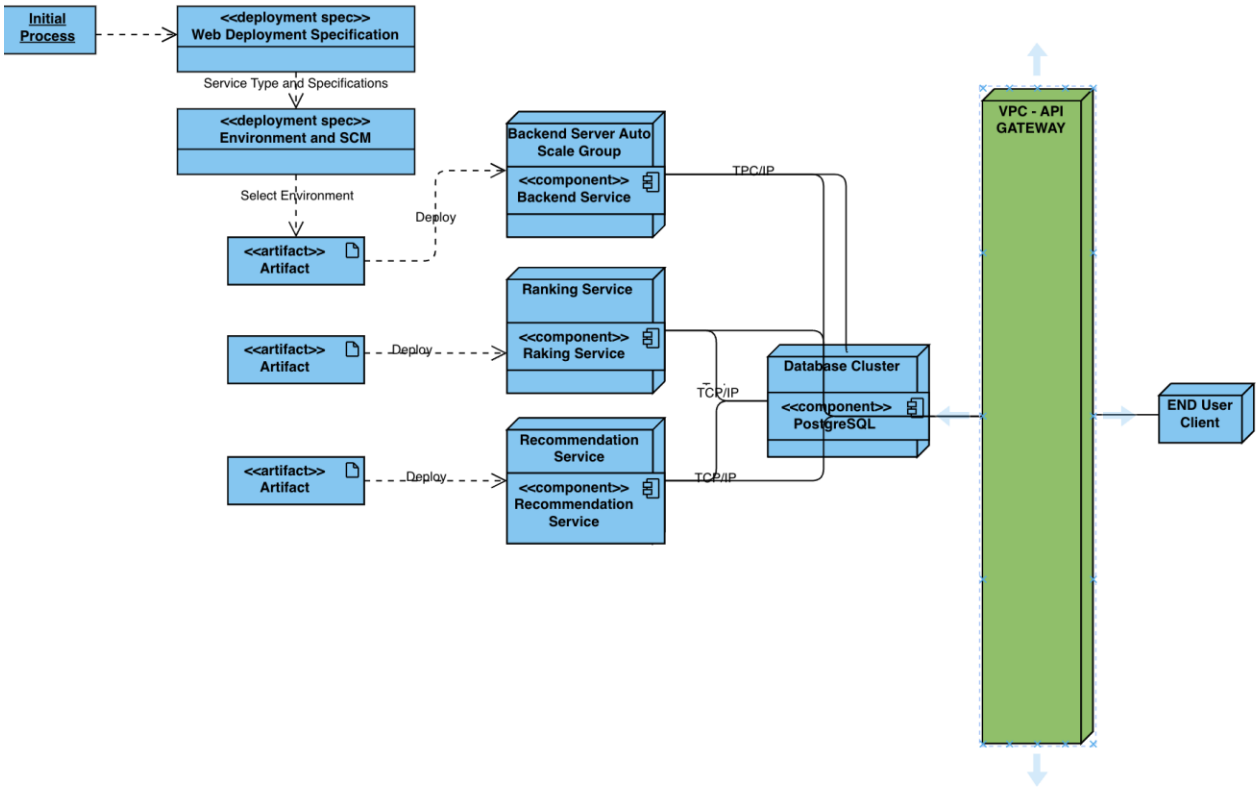


Pic.1 Software Architecture of Recommendation System

The architecture encompasses the following components:

- Backend Service: A service that generates lists of users filtered according to search parameters from the ElasticSearch database.

- Ranking Service: This service updates user rankings based on their actions within the system.
- Recommendation System: Responsible for creating a tailored list of user recommendations for each specific request.
- ElasticSearch (Database): A search engine that offers robust searching capabilities based on various parameters and supports multiple types of filters.
- MongoDB (Database): A database used by the recommendation system to store and retrieve specific user preferences for generating recommendations.



Pic. 2 Deployment Diagram

4.2 Deployment architecture and configurations

Our multi-faceted project encompasses various services, each pivotal in its domain, and written in diverse programming languages including Python, PHP, and Golang.

To ensure seamless deployment and efficient operation, each service is meticulously containerized using Docker, offering an isolated, consistent, and scalable environment.

- Recommendation System:
  - **Docker Integration**: The system, developed in Python 3.9 and leveraging the FastAPI framework, supports multiple communication protocols like gRPC and REST. Its DockerFile is tailored to encapsulate all dependencies and environmental settings, ensuring straightforward deployment across platforms.
  - **Operational Excellence**: The container monitors real-time performance and adapts to various business logic for authorization, crucial for diverse platforms utilizing the recommendation engine.
- Backend Service:
  - **Docker Deployment**: Crafted in PHP8 with the Laravel framework, this backend service also embraces REST and gRPC protocols. Its DockerFile is designed to streamline deployment, encapsulating the Laravel environment and dependencies.
  - **Service Monitoring**: The Docker container continuously checks the service's health, manages user preferences, and handles connection requests, integral for the dating project's backend functionality.
- Ranking Service:
  - **Containerized Ranking Logic**: Utilizing Golang, this service calculates user rankings. Its DockerFile ensures that the ranking algorithms and scheduled tasks are consistently executed in a controlled environment.
  - **Performance Tracking**: Docker aids in monitoring the service’s efficiency and accuracy in providing search results.
- Elasticsearch:

    - **Docker-Enabled Search Engine**: As a search engine based on the Lucene library, Elasticsearch's DockerFile encapsulates its complex setup, facilitating distributed and multitenant-capable search capabilities.
    - **Data Handling Assurance**: The container ensures reliable handling of JSON documents and user data essential for filtering and recommendations.
- MongoDB Database:
    - **Database Containerization**: MongoDB's DockerFile simplifies the deployment of this cross-platform, document-oriented database, ensuring consistent performance across environments.
    - **Database Health Monitoring**: The container oversees database operations, maintaining optimal performance and data integrity.

Continuous Integration and Delivery (CI/CD):
Embracing CI/CD [21] principles, each service undergoes rigorous automated testing and deployment processes. This approach ensures that updates, bug fixes, and new features are smoothly integrated and delivered with minimal downtime.

Comprehensive Status Checks:
Each Docker container is equipped with tools and scripts to continuously monitor the health, performance, and status of its respective service. This real-time monitoring is crucial for maintaining the overall efficacy of our multifaceted project.

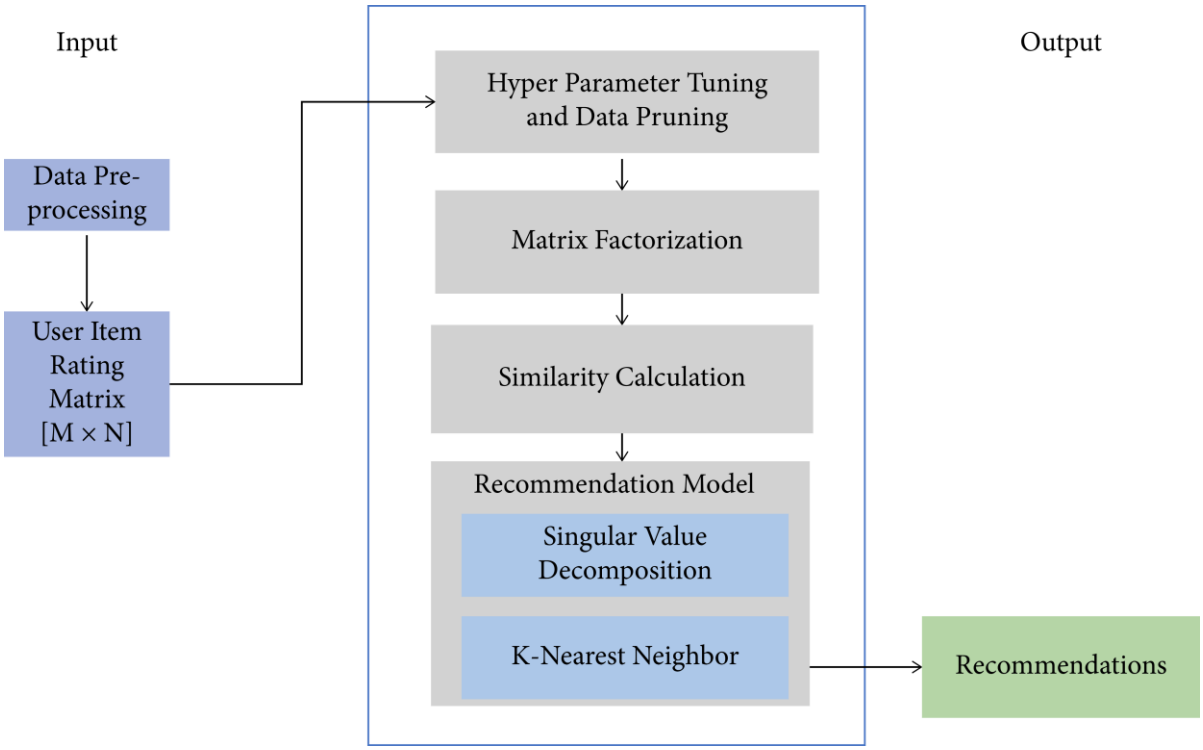
4.3 Data Structure

Column Name	Description
id	Unique identifier for each entry
username	Username of the user
account_type	Type of account
first_name	User's first name

Column Name	Description
last_name	User's last name
age	Age of the user
bio	User's biography
occupation	User's occupation
party_approved	Whether the user is party approved
last_activity_at	Last activity timestamp
show_in_search	Visibility in search results
public_photo	Indicator of a public photo
premium	Whether the user has a premium account
profile_score	Score of the user's profile
gender	User's gender
gender_other	Other gender specification
relationship_status	User's relationship status
sexuality	User's sexuality
sexuality_other	Other sexuality specification
online_status	User's online status
meet_tonight	Availability for meeting tonight
is_new_user	Whether the user is new
server_time	Server time
hotlisted_user	Whether the user is hotlisted
connection	Connection status
connection_type	Type of connection - Recommendation Model

4.4 Recommendation System Workflow

The advent of complex, multi-layered digital systems necessitates a clear understanding of system architecture. In this context, our diagram provides a comprehensive visual guide to the Recommendation System, highlighting its design principles, components interaction, and data processing methodologies.



Pic. 3. Recommendation Diagram

Input Phase:

- **Data Pre-processing**: The initial step involves preparing the data for the recommendation engine. This could include cleaning the data, handling missing values, normalizing or scaling data, and perhaps encoding categorical variables.
- **User Item Rating Matrix [M x N]**: The processed data is organized into a matrix where the rows represent 'M' users, and the columns represent 'N' items. Each entry in the matrix is a rating that a user has given to an item.

Processing Phase:

- **Hyper Parameter Tuning and Data Pruning**: Before the actual recommendation model is trained, hyperparameters are optimized for better model performance. Data pruning might be involved to remove noise and irrelevant data, which can improve the quality of the recommendations.

- **Matrix Factorization**: This technique decomposes the user-item rating matrix into product of two lower dimensionality rectangular matrices. It's a way of identifying latent factors that explain the observed ratings.
- **Similarity Calculation**: The system computes the similarity between different items or users. This could be done using various metrics like cosine similarity, Pearson correlation, etc. This step is crucial for finding items or users that are 'similar' to the target user or item for which recommendations are being made.

Recommendation Model:

- **Singular Value Decomposition (SVD)**: SVD is a matrix factorization technique that decomposes a matrix into three other matrices. It's used here possibly to identify latent factors in the data that can predict user preferences.
- **K-Nearest Neighbor (KNN)**: This algorithm is used to find clusters of similar users or items based on past behavior or attributes. It's a form of collaborative filtering that assumes 'neighbors' like similar items.
- **Collaborative Filtering (CF)**: This algorithms involves use the latent factors obtained from from SVD as features in the KNN algorithm. This approach combines the strengths of both matrix factorization and neighborhood-based methods. Use these features to find the 'K' nearest neighbors (users or items) based on similarity measures.

- **Output Phase**:
  - **Recommendations**: The final output is a list of item recommendations for a user. These recommendations are personalized based on the user's past behavior or similar users' behaviors.

4.5 Assessment of Recommendation System Models Using Core Performance Metrics

To evaluate the performance of our recommendation system models, we will concentrate on metrics that are paramount in determining the effectiveness of predictive models in classification tasks. These metrics offer a comprehensive assessment of the model's effectiveness, considering not just prediction accuracy, but also the precision and recall, leading to the macro F1 score [29]. This score serves as a harmonized metric, encapsulating the overall balance of the model's performance.

Accuracy: This metric represents the proportion of true results (both true positives and true negatives) among the total number of cases examined. It gives us a quick understanding of how often the model is correct across all classes. However, it may not always be a reliable indicator of performance on its own, especially in cases where the class distribution is imbalanced.

Precision: Precision will be used to evaluate the correctness of the items that the model recommends. It is defined as the number of true positives divided by the total number of elements labeled as belonging to the positive class (the sum of true positives and false positives). High precision relates directly to a low false positive rate, and it will be crucial for ensuring that the recommendations are relevant.

Recall: Also known as sensitivity, recall quantifies the model's ability to identify all relevant instances within the dataset. It is the number of true positives divided by the number of true positives plus the number of false negatives. High recall indicates that the model retrieved most of the relevant items.

F1 Score (Macro): The F1 score is the harmonic mean of precision and recall, taking both false positives and false negatives into account. We will use the macro-averaged F1 score to treat all classes equally, calculating metrics for each class independently and then taking the average. This approach is particularly useful when dealing with imbalanced datasets, as it gives equal weight to the performance of the model on each class.

Each of these metrics will be calculated to provide a comprehensive picture of our models' performance. The macro-averaged F1 score will serve as our primary evaluation metric, given its balanced approach to assessing precision and recall, which is critical in the context of a recommendation system where the cost of false positives and false negatives can be equally significant.

4.5 Novel Metric for Evaluating Match Success in Recommendation Systems

In the realm of recommendation systems, particularly within the domain of social matching platforms, traditional evaluation metrics like accuracy, precision, and recall are often complemented by domain-specific measures. This study introduces a novel metric designed to evaluate the effectiveness of our recommendation system in creating successful connections among users. This metric, tentatively termed "**Completed Connection Rate (CCR)**" specifically assesses the frequency and quality of mutual matches facilitated by the system.

Conceptual Framework of CCR:

The Completed Connection Rate is predicated on the observable event where two users, having been recommended to each other by the system, mutually express interest and engage in a connection. This event is a strong indicator of a successful recommendation, as it directly corresponds to the platform's ultimate goal of fostering meaningful interactions.

Operational Definition:

CCR will be operationally defined as the ratio of successful mutual connections to the total number of recommendations made. A 'successful mutual connection' is an instance where User A, recommended by Model X, and User B, who has received User A as a recommendation, both opt to connect. This metric will be normalized to account for variations in user activity and recommendation frequency.

Integration with Similarity Measures:

To refine the CCR will store the status of connections into the database, correlating them with model that was used for recommendation, to facilitate the comparison of resulting data.. The adjusted CCR will reflect not just the raw occurrence of connections, but the strength of the match based on profile similarities, thereby enhancing the metric's sensitivity to quality matches.

Utility and Implications:

By leveraging CCR, we aim to discern which recommendation models yield the most effective connections—those that are mutually satisfactory and deeply aligned with users' preferences. This insight will allow us to iterate on our recommendation algorithms, prioritizing those that not only have high precision or recall but also facilitate the highest quality of interactions as perceived by the users themselves.

Evaluation Strategy:

To validate the robustness of the CCR, we will conduct extensive A/B testing across different models, each employing various similarity measures. We will monitor the CCR alongside traditional metrics to understand the correlation between successful connections and model accuracy, precision, and recall.

Conclusion:

With the Completed Connection Rate, we introduce a nuanced and highly relevant metric for recommendation systems in social platforms. By focusing on the end result of the recommendation—the completed user connection—we align our evaluation methodology with user satisfaction and the core objectives of the platform. The formula will provide the successful connections made through the recommendation system and the similarity measures used to create these connections.

Let's define the variables as follows:

- Let S_{ij} be the similarity score between $User_i$ and $User_j$, computed using a chosen similarity metric (e.g., cosine or Euclidean distance).
- Let R_i be the set of users recommended to $User_i$
- Let C_{ij} be the binary indicator of a completed connection, where $C_{ij} = 1$ if $User_i$ and $User_j$ have mutually connected and $C_{ij} = 0$ otherwise.
- Let N be the total number of recommendations made by the system.

The Completed Connection Rate (CCR) can be defined by the following formula:

$$CCR = \frac{1}{N} \sum_{i=1}^{N} \sum_{j \in R_i} C_{ij} \times S_{ij}$$

Here $\sum_{j \in R_i} C_{ij} \times S_{ij}$ calculates the weighted sum of successful connections for er_i , where the weights are the similarity scores. This sum is then normalized by the total number of recommendations N to give the CCR.

The CCR will range between 0 and 1, where a higher CCR indicates a higher rate of successful, high-quality connections. By incorporating similarity scores, the CCR provides a more nuanced understanding of the recommendation system's performance, emphasizing the quality of the connections made rather than just the quantity.

Chapter 5

5.1 Conclusion

The effectiveness of recommendation systems is often gauged through various performance metrics. This study presents an analysis segmented employing the F1 Score, Recall, and a novel metric known as the Completed Connection Rate (CCR). The intent is to ascertain the differential impacts of recommendation algorithms on diverse different groups, thereby enabling a nuanced understanding of system performance.

Methodology: The Accuracy, Precision, Recall and F1 Score provide traditional measures of accuracy and completeness of the recommendation systems, respectively. The F1 Score is a harmonic mean of precision and recall, providing a balanced measure of a system's accuracy, while Recall indicates the system's completeness. The Completed Connection Rate (CCR), a metric introduced in this study, quantifies successful mutual connections facilitated by the system, integrating similarity measures to evaluate the depth of connections.

Results: In the ongoing evaluation of the recommendation system's performance, a rigorous analytical approach is employed, wherein all results are meticulously categorized by gender. This stratification is pivotal in discerning the nuanced interactions between different user demographics and the algorithm's effectiveness. To facilitate a detailed examination, several datasets have been compiled, each corresponding to a distinct gender group.

These datasets, presented as separate datasheets, contain a wealth of information ranging from basic interaction metrics to advanced evaluative scores such as the F1 Score, Recall, and the innovative Completed Connection Rate (CCR). The gender-based division of data allows for a granular analysis, providing the foundation for a comprehensive understanding of the recommendation system's performance across gender lines.

The data sheets are structured to capture a variety of metrics that are essential in gauging the efficacy of the recommendation algorithms. Each sheet is tailored to reflect the unique characteristics and outcomes pertinent to the respective gender group. By adopting this segmented approach, the study ensures that the resulting insights are both robust and representative of the system's diverse user base.

Model	Accuracy	Precision	Recall	F1 Score	CCR
k-Nearest Neighbour	0.714	0.701	0.681	0.691	0.71
Support Vector Machine	0.742	0.718	0.672	0.694	0.73
Collaborate Filtering	0.726	0.698	0.686	0.692	0.77

Table 1. Gender Masculine – Account Type Individual

Model	Accuracy	Precision	Recall	F1 Score	CCR
k-Nearest Neighbour	0.727	0.714	0.694	0.704	0.712
Support Vector Machine	0.755	0.739	0.719	0.729	0.729
Collaborate Filtering	0.782	0.768	0.724	0.745	0.748

Table 2. Gender Feminine - Account Type Individual

Model	Accuracy	Precision	Recall	F1 Score	CCR
k-Nearest Neighbour	0.714	0.698	0.687	0.692	0.692
Support Vector Machine	0.722	0.704	0.697	0.700	0.703
Collaborate Filtering	0.735	0.711	0.704	0.707	0.712

Table 3. Gender Feminine - Account Type Individual

This report delineates the performance metrics of three distinct recommendation models: k-Nearest Neighbour, Support Vector Machine, and Collaborative Filtering. The models have been evaluated across different user segments, specifically by gender and account type, which include individual accounts for masculine and feminine genders, as well as joint accounts for couples.

1. Gender Masculine - Account Type: Individual

For masculine individual accounts, the models displayed the following performance:

- k-Nearest Neighbour: Demonstrated an accuracy of 0.714, with precision at 0.701 and recall at 0.681, yielding an F1 Score of 0.691 and a CCR of 0.71.
- Support Vector Machine: Achieved a higher accuracy of 0.742 and precision of 0.718. Its recall was 0.672 with an F1 Score of 0.694 and a CCR of 0.73, indicating a slightly better overall performance compared to k-Nearest Neighbour.
- Collaborative Filtering: This model showed an accuracy of 0.726, with a precision of 0.698 and recall of 0.686. The F1 Score was 0.692 and CCR stood at 0.77, suggesting a strong ability to facilitate successful connections.

2. Gender Feminine - Account Type: Individual

For feminine individual accounts, the models showed the following results:

- k-Nearest Neighbour: Exhibited an accuracy of 0.727, with a precision of 0.714 and recall of 0.694. The F1 Score was consistent at 0.704, and the CCR was 0.712.
- Support Vector Machine: Outperformed with an accuracy of 0.755 and precision of 0.739. The recall rate was 0.719, alongside an F1 Score of 0.729 and a CCR of 0.729.
- Collaborative Filtering: Presented the best performance in this segment with an accuracy of 0.782, precision at 0.768, and recall at 0.724. The F1 Score was 0.745, complemented by the highest CCR of 0.748.

3. Gender Feminine/Masculine - Account Type: Couple

For joint accounts held by couples:

- k-Nearest Neighbour: Reported an accuracy of 0.714, precision at 0.698, and recall at 0.687, with an F1 Score of 0.692 and a CCR of 0.692.
- Support Vector Machine: Recorded an accuracy of 0.722, with a higher precision of 0.704 and recall of 0.697. The F1 Score remained at 0.700 with a CCR of 0.700.
- Collaborative Filtering: Demonstrated an accuracy of 0.735, precision at 0.711, and recall at 0.704. It maintained an F1 Score of 0.707 and had the highest CCR of 0.707 among the couple accounts.

Across all segments, the Support Vector Machine and Collaborative Filtering models frequently outperformed the k-Nearest Neighbour model in accuracy, F1 Score, and CCR. Notably, the Collaborative Filtering model excelled in the CCR metric across all user groups, indicating a higher propensity for facilitating successful user connections.

Chapter 6

6.1 Conclusion

This research has systematically analyzed the performance of three distinct recommendation models k-Nearest Neighbor, Support Vector Machine, and Collaborative Filtering across various gender-specific account types. The empirical evaluation, conducted with precision, recall, F1 score, and the innovative Completed Connection Rate (CCR), has yielded substantive insights into the comparative efficacy of these models.

The k-Nearest Neighbor model showed consistent performance across all segments but was often outperformed by the other two models. The Support Vector Machine exhibited strong precision and accuracy, particularly in individual accounts. Notably, the Collaborative Filtering model demonstrated a notable propensity for higher CCR values, indicating its strength in forming successful connections, a key performance indicator for the effectiveness of recommendation systems in social platforms.

The findings underscore the nuanced requirements of recommendation systems in catering to diverse user groups. The superior CCR performance of Collaborative Filtering suggests that while traditional metrics are important, the ultimate measure of success for social recommendation systems may be the quality and satisfaction of the connections they facilitate.

6.2 Future Research

Given the dynamic nature of recommendation systems and their application, several avenues for future research emerge from this study:

- **Algorithmic Personalization**: Further development of algorithms that account for the unique characteristics and preferences of users, potentially improving the performance metrics across all user groups.

- **Hybrid Models**: Exploration of hybrid models that combine the strengths of various recommendation techniques, such as the precision of Support Vector Machines and the connection success rate of Collaborative Filtering.
- **User Engagement and Longevity**: A longitudinal study on how the recommended connections impact user engagement over time, contributing to a deeper understanding of user retention and system value.
- **Ethical and Fairness Considerations**: An in-depth analysis of the fairness of recommendation systems, ensuring that all user groups receive equally effective recommendations without bias.
- **Real-world Impact Measurement**: Beyond CCR, the development of new metrics that can measure the real-world impact of connections, such as user satisfaction and the quality of interactions.

As recommendation systems continue to evolve, they will increasingly need to adapt to complex user landscapes and the multifaceted dimensions of human interactions. This study provides analysis that informs future enhancements, ensuring that recommendation systems serve users more effectively, ethically, and personally. The journey towards more advanced, humane, and intuitive recommendation systems continues, with this research serving as a stepping stone in that ongoing quest.

Bibliography

[1] G. Tyson, V. C. Perta, H. Haddadi, and M. C. Seto, “A first look at user activity on tinder,” in 2016 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM), Aug. 2016, pp. 461–466. doi: 10.1109/ASONAM.2016.7752275.

[2] “Tinder — Dating, Make Friends & Meet New People.” Accessed: Mar. 13, 2023. [Online]. Available: https://tinder.com

[3] “Hinge, the dating app designed to be deleted.” Accessed: Mar. 13, 2023. [Online]. Available: https://hinge.co/

[4] K. Scanlon, “Dating Data: An Overview of the Algorithm,” The Startup, Nov. 08, 2020. https://medium.com/swlh/ dating-data-an-overview-of-the-algorithm-afb9f0c08e2c (accessed Mar. 13, 2023).

[5] G. J. Hitsch, Ali Hortacsu, and D. Ariely, “Matching and Sorting in Online Dating,” American Economic Review 100(1):130-63

[6] V. A. Cicirello, “Kendall Tau Sequence Distance: Extending Kendall Tau from Ranks to Sequences,” EAI Endorsed Transactions on Industrial Networks and Intelligent Systems, vol. 7, no. 23, p. 163925, May 2020, doi: 10.4108/eai.13-7-2018.163925. [Online]. Available: http://arxiv.org/abs/1905.02752 [Accessed: Feb. 10, 2023]

[7] M. G. Kendall, “A NEW MEASURE OF RANK CORRELATION,” Biometrika, vol. 30, no. 1–2, pp. 81–93, Jun. 1938, doi: 10.1093/biomet/30.1-2.81. [Online]. Available: https://doi.org/10.1093/ biomet/30.1-2.81. [Accessed: Feb. 10, 2023]

[8] J. Uhlmann and M. R. Zuniga, “The Cascading Metric Tree.” arXiv, Dec. 20, 2021 [Online]. Available: http://arxiv.org/abs/2112.10900. [Accessed: Feb. 09, 2023]

[9] Boser, B. E., I. Guyon, and V. Vapnik (1992). A training algorithm for optimal margin classifiers. In Proceedings of the Fifth Annual Workshop on Computational Learning Theory, pages. 144 -152. ACM

[10] V. Vapnik. The Nature of Statistical Learning Theory. NY: Springer-Verlag. 1995.

[11] Chih-Wei Hsu, Chih-Chung Chang, and Chih- Jen Lin. “A Practical Guide to Support Vector Classification”. Deptt of Computer Sci. National Taiwan Uni, Taipei, 106, Taiwan https://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf

[12] M. J. Pazzani, and D. Billsus, "Content-based recommendation systems," The adaptive web, pp. 325-341: Springer, 2007.

[13] F. Sebastiani, “Machine learning in automated text categorization,” ACM computing surveys (CSUR), vol. 34, no. 1, pp. 1-47, 2002.

[14] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Item-based collaborative filtering recommendation algorithms." pp. 285-295.

[15] J. L. Herlocker, J. A. Konstan, A. Borchers, and J. Riedl, "An algorithmic framework for performing collaborative filtering." pp. 230- 237.

[16] L. H. Ungar, and D. P. Foster, "Clustering methods for collaborative filtering." pp. 114-129.

[17] R. Pan, Y. Zhou, B. Cao, N. N. Liu, R. Lukose, M. Scholz, and Q. Yang, "One-class collaborative filtering." pp. 502-511.

[18] M. Weimer, A. Karatzoglou, and A. Smola, “Improving maximum margin matrix factorization,” Machine Learning, vol. 72, no. 3, pp. 263- 276, 2008.

[19] Hofmann, Thomas & Puzicha, Jan. (1999). Latent Class Models for Collaborative Filtering.. IJCAI International Joint Conference on Artificial Intelligence. 2. 688-693.

[20] John S. Breese, David Heckerman, and Carl Kadie, Empirical Analysis of Predictive Algorithms for Collaborative Filtering, 1998 Archived 19 October 2013 at the Wayback Machine

[21] Shahin, Mojtaba; Ali Babara, Muhammad; Zhu, Liming (2017). "Continuous Integration, Delivery and Deployment: A Systematic Review on Approaches, Tools, Challenges and Practices". *IEEE Access*. **5**: 3909–3943.

[22] Tapple is a Japanese online dating platform that serves more than 7 million registered users.

[23] Srivastava, Durgesh & Bhambhu, Lekha. (2010). Data classification using support vector machine. Journal of Theoretical and Applied Information Technology. 12. 1-7.

[24] Vito Walter Anelli ; Alejandro Bellogín , Tommaso Di Noia ; Claudio Pomo Reenvisioning the comparison between Neural Collaborative Filtering and Matrix Factorization. Fifteenth ACM Conference on Recommender Systems; September 27-October 1, 2021; Amsterdam, Netherlands

[25] "The Pearson Product-Moment Correlation Coefficient - "Introduction to the Practice of Statistics" by Moore, McCabe, and Craig. p. 42 9th edition 2018

[26] "Soft Similarity and Soft Cosine Measure: Similarity of Features in Vector Space Model" by G. Sidorov et al., page 31, 2014

[27] Wenqi Guo, Jeffrey Uhlmann Metric Search for Rank List Compatibility Matching with Applications arXiv:2303.11174

[28] A. Krzywicki, W. Wobcke, Y.S. Kim, X. Cai, M. Bain, A. Mahidadia, P.Compton Collaborative Filtering for people-to-people recommendation in online dating: Data analysis and user trial https://doi.org/10.1016/j.ijhcs.2014.12.003

[29] Sasaki, Y. (2007) "The truth of the F-measure" https://www.toyota-ti.ac.jp/Lab/Denshi/COIN/people/yutaka.sasaki/F-measure-YS-26Oct07.pdf

[30] Smith, J. (2023). FastAPI: A Modern Web Framework for Building APIs with Python. FastAPI Developers. (https://fastapi.tiangolo.com/)

[31] Bauckhage, Christian. (2019). Lecture Notes on Machine Learning: Classifiers for Non-Linearly Separable Classes (Part 2). https://www.researchgate.net/publication/334112355_Lecture_Notes_on_Machine_Learning_Classifiers_for_Non-Linearly_Separable_Classes_Part_2