PEOPLE'S DEMOCRATIC REPUBLIC OF ALGERIA
Ministry of Higher Education and Scientific Research

ECHAHID HAMMA LAKHDAR UNIVERSITY - EL OUED
FACULTY OF EXACT SCIENCES
Computer Science department

End of Study Memory
Presented for the Diploma of

ACADEMIC MASTER

Domain : **Mathematics and Computer Science**
spinneret: **Computer Science**
Specialty : **Artificial Intelligence and Distributed Systems**

Presented by :

- **Badra Sakhri**
- **Hecine Boulaaras**

Theme

Time Aware Circle Based Recommendation in Online social Network

Supported in: - June 06 th 2023 In front of jury:

M.	Brahim Lejdel	President
M.	Farida Retima	Reporter
Dr.	Mohammed Anouar Naoui	Supervisor

Academic year **:** 20**22**/20**23**

ABSTRACT

The rapid growth of online social networks has resulted in an overwhelming amount of content generated and shared by users. As a consequence, users often struggle to keep up with the vast amount of information available to them. In this context, personalized recommendations play a crucial role in assisting users in discovering relevant content and enhancing their overall user experience. TACBR approach combines the temporal aspects of user interactions and the social connections within online social networks to provide personalized recommendations. By creating time-aware circles and leveraging these circles for content recommendations, TACBR aims to enhance the user experience by assisting users in discovering relevant and timely content.

key words : Online social networks , Content generation , Overwhelming amount of information , Personalized recommendations , User experience , Temporal aspects of user interactions , Social connections , Time-aware circles , Relevant content ,Timely content .

RÉSUMÉ

La croissance rapide des réseaux sociaux en ligne a entraîné une quantité écrasante de contenu généré et partagé par les utilisateurs. En conséquence, les utilisateurs ont souvent du mal à suivre la vaste quantité d'informations à leur disposition. Dans ce contexte, les recommandations personnalisées jouent un rôle crucial en aidant les utilisateurs à découvrir du contenu pertinent et en améliorant leur expérience globale.

L'approche TACBR (Time Aware Circle Base Recommendation) combine les aspects temporels des interactions des utilisateurs et les connexions sociales au sein des réseaux sociaux en ligne pour fournir des recommandations personnalisées. En créant des cercles sensibles au temps et en exploitant ces cercles pour des recommandations de contenu, TACBR vise à améliorer l'expérience utilisateur en aidant les utilisateurs à découvrir du contenu pertinent et actuel.

mots clés : Réseaux sociaux en ligne , Contenu généré , Quantité écrasante d'informations, Recommandations personnalisées , Expérience utilisateur , Aspects temporels des interactions des utilisateurs , Connexions sociales , Cercles sensibles au temps , Contenu pertinent, Contenu actuel

تلخيص

أدى النمو السريع للشبكات الاجتماعية عبر الإنترنت إلى قدر هائل من المحتوى الذي تم إنشاؤه ومشاركته من قبل المستخدمين. نتيجة لذلك ، غالبا ما يكافح المستخدمون لمواكبة الكم الهائل من المعلومات المتاحة لهم. في هذا السياق ، تلعب التوصيات المخصصة دورا حاسما في مساعدة المستخدمين على اكتشاف المحتوى ذي الصلة وتعزيز تجربة المستخدم الإجمالية .

نهج **TACBR(Time Aware Circle Based Recommendation)** يجمع بين الجوانب الزمنية لتفاعلات المستخدم والاتصالات الاجتماعية داخل الشبكات الاجتماعية على الانترنت لتقديم توصيات شخصية. من خلال إنشاء دوائر مدركة للوقت والاستفادة من هذه الدوائر لتوصيات المحتوى ، يهدف المركز إلى تحسين تجربة المستخدم من خلال مساعدة المستخدمين في اكتشاف المحتوى ذي الصلة وفي الوقت المناسب..

الكلمات المفتاحية : شبكات التواصل الاجتماعي على الانترنت , التوصيات المخصصة , تجربة المستخدم , الروابط الاجتماعية , الروابط الاجتماعية , الدوائر الزمنية المدركة , المحتوى ذي الصلة المحتوى المواكب للزمن .

DEDICATION

We dedicate this modest work to whom brought us love, support and happiness , to the dearest people to our hearts, To our dear parents, for their; love, sacrifice, patience, moral and material support from our childhood until this day, To all our brothers and sisters , who supported us, encouraged us and who took the way with us along all the courses of our studies, To all our dear nephews , to all our uncles and cousins and everyone in our big family. To our dear friends

Hecine

DEDICATION

Praise and thanks are only for Allah, the One who, by His blessing and favor, perfected works and deeds are accomplished.
All Praise be to God, who has bestowed upon us the grace of health and wellness. Without him, this humble work could not be completed. Having said that :
To the one who is suffering day after day, so that we may acquire the best knowledge and rise in its ranks. My dear father ***Sakhri Rida Malek***.
May you continue to be my treasure and a crown upon my head.
To the one who spent sleepless nights praying and supplicating; my mother ***Kebaili Laila***, who gave us care and tenderness.
To my brothers who have supported me throughout my journey:
Aymen,Imade Eddine ana Abd Elmoumen.
To my sisters : ***Sidra***.
To ***my grandmother, my maternal and paternal uncles, and aunts***.
To my friends :***Rania,Chaima,Meriem,Aya, Donia,Kawther, Samira***. To the pure souls, we lost but their shadows have never gone from home, my tender grandmother and grandfather.
For everyone who was by my side, and helped me from near or far, even with a kind word
. To my professors and to everyone who taught me a letter, to the Department of Computer Science in particular, and the entire University of Mohammed Lakhdar.
Thank God.
Sakhri Badra.

KNOWLEDGMENT

“ Thanks to the one God, Light of the heavens and the earth, who help and guide ”

we must first of all thank “ ALLAH ” the almighty, who gave us the power, the will and the patience to develop this work .
Our sincerest thanks to our mentor Mr. Naoui Mohammed Anouar for his availability, his contributions, his valuable guidance and his understanding throughout the development of this memoir.
We would also like to thank the members of the jury very much for accepting, examine and evaluate this modest work .
We also thank all our teachers of Licence and Master in the department of Computing Science.

Thank you.

Table of Contents

List of Figures

List of Tables

INTRODUCTION

Time aware circle based recommendation in online social networks is a technique that uses machine learning algorithms to provide personalized recommendations to users based on their social network connections and the time factor. In online social networks, users are connected to each other through various social circles, such as friends, family, and colleagues. These social circles can be used to provide personalized recommendations to users based on the preferences and behavior of their connections.

Time aware circle based recommendation takes into account the time factor, which is an important consideration in online social networks. User preferences and behavior can change over time, and recommendations that were relevant in the past may no longer be relevant in the present. By taking into account the time factor, time-aware circle-based recommendation can provide more accurate and relevant recommendations to users.

This technique involves analyzing user data and behavior, as well as the data and behavior of their social connections, to identify patterns and make

predictions about which items or content a user is likely to be interested in. Machine learning algorithms are used to train the recommendation model and improve its accuracy over time.

Object presentation

The Time Aware Circle based Recommendation System is an algorithmic framework that aims to provide more personalized and contextually relevant recommendations in online social networks. It utilizes the concept of Friends Circles, incorporates time-awareness, emphasizes social network characteristics, and employs trust computation algorithms to enhance recommendation accuracy. By considering the relationships and interactions among users within specific circles, the system captures common properties and interests, recognizes the dynamic nature of user preferences, and measures the preferences and influence of trusted users to generate recommendations based on reliable sources and trusted connections.

The problematic

1. The first issue is limited temporal consideration, where the systems fail to account for the dynamic nature of user interests and evolving trends, resulting in outdated or irrelevant recommendations.

2. The second issue is the lack of personalization in circle-based recommendations, which rely on social connections or group memberships and may overlook individual preferences and interests, leading to a lack of per-

sonalization.

Work plan

This thesis contains four chapters as following :

- **Chapter one**

  The chapter explores recommendation systems' role in enhancing personalized experiences and aiding decision-making processes. It covers fundamental concepts, techniques, and applications of recommendation systems. Topics include types of recommendation systems, technical aspects like algorithms and methodologies, evaluation methods, and emerging trends such as contextual information and time-aware recommendations. The chapter emphasizes the importance of addressing information overload and providing personalized suggestions to improve user satisfaction.

- **Chapter two**

  The chapter introduces the field of machine learning, highlighting its transformative impact on various industries. It covers supervised and unsupervised learning algorithms, including regression, decision trees, clustering, and dimensionality reduction. and explores advanced topics like deep learning and reinforcement learning.

- **Chapter three**

The chapter focuses on building Time Aware Circle-based Recommendation systems in online social networks. It provides an explanation of the concept and its significance in enhancing recommendation accuracy. The chapter includes examples and discusses the characteristics of social networks and trust computation

Finally, we achieve this thesis by a general conclusion where we resume our main contribution and give some orientation for further work.

CHAPTER 1

RECOMMENDATION SYSTEM

1.1 Introduction

Recommendation systems are a type of artificial intelligence that provide personalized suggestions to users based on their past behavior, preferences, and interests. These systems are widely used in e-commerce, social media, and entertainment platforms to improve user engagement and satisfaction.

There are several types of recommendation systems, including collaborative filtering, content-based filtering, and hybrid systems that combine both approaches. Collaborative filtering relies on user behavior data to identify patterns and similarities between users, while content-based filtering analyzes the characteristics of items to make recommendations.

Recent advancements in deep learning and natural language processing have also led to the development of more sophisticated recommendation systems that can analyze unstructured data such as text and images.

Some popular examples of recommendation systems include Amazon's product recommendations, Netflix's movie and TV show suggestions, and

Spotify's music recommendations.

1.2 Definition of recommender systems

A recommender system is an intelligent system that suggests relevant items, products, services, or information to users based on their preferences, interests, behavior, and past interactions. The goal of a recommender system is to provide personalized recommendations to users, which can help them discover new items that they might not have found otherwise, or save time in searching for relevant information or products. Recommender systems are widely used in various industries, including e-commerce, social media, entertainment, and information retrieval, to name a few. The most common types of recommender systems include collaborative filtering, content-based filtering, hybrid systems that combine both approaches, and knowledge-based systems that rely on expert knowledge to make recommendations. The key challenge in building an effective recommender system is to accurately capture and model user preferences and behavior, and to provide recommendations that are both relevant and diverse. To achieve this, recommender systems use a variety of techniques, such as data mining, machine learning, natural language processing, and semantic analysis, among others[3].

1.3 Designing User Recommendation System

Designing a user recommendation system involves several steps, including[3, 4]:

1.3.1 Defining the problem:

The problem at hand is designing a user recommendation system. This involves creating a system that can effectively predict user preferences and provide personalized recommendations based on user data, such as user profiles, user behavior, and user feedback. The process of designing a recommendation system involves several steps, including data collection, data preprocessing, feature engineering, model selection, and evaluation. The goal is to create a system that can provide accurate and relevant recommendations to users, thereby improving their overall experience and satisfaction. The solution may involve using various recommendation models, such as collaborative filtering, content-based filtering, and hybrid models, and evaluating their performance using metrics such as precision, recall, and F1 score. Ultimately, the success of the recommendation system will depend on its ability to meet the user's needs and provide a personalized experience.

1.3.2 Data collection:

Data collection is the first step in designing a user recommendation system. It involves collecting user data from various sources, such as user registra-

tion forms, user activity logs, and user surveys. The collected data may include user profiles, user behavior, and user feedback. User profiles may include demographic information, such as age, gender, and location, as well as user preferences, such as favorite genres or products. User behavior may include data on the user's interactions with the system, such as clicks, views, and purchases. User feedback may include ratings, reviews, and comments provided by the user. The data collected should be relevant to the recommendation system's goals and should be of high quality to ensure accurate predictions. Once the data is collected, it needs to be preprocessed and transformed into a suitable format for analysis, which is the next step in the process.

1.3.3 Pre-processing and feature extraction:

Preprocessing and feature extraction are the next steps in designing a user recommendation system. Once the data is collected, it needs to be cleaned and preprocessed to remove noise, handle missing values, and transform the data into a suitable format for analysis. This step may involve techniques such as data normalization, data imputation, and data encoding.

After preprocessing, relevant features need to be extracted from the data that can be used to train a recommendation model. Feature engineering involves selecting and transforming the data into a set of features that can be used to represent the user's preferences and behavior. These features may include user demographics, user preferences, and user behavior. For example, user demographics may include age, gender, and location, while

user preferences may include favorite genres or products. User behavior may include data on the user's interactions with the system, such as clicks, views, and purchases.

The goal of feature engineering is to create a set of features that can effectively capture the user's preferences and behavior and can be used to train a recommendation model. The features should be relevant to the recommendation system's goals and should be of high quality to ensure accurate predictions. Once the features are extracted, they can be used to train a recommendation model, which is the next step in the process.

1.3.4 Train the model:

Training the model is a crucial step in designing a user recommendation system. Once the data is preprocessed and relevant features are extracted, a suitable recommendation model needs to be selected and trained using the extracted features. There are several types of recommendation models, including collaborative filtering, content-based filtering, and hybrid models.

Collaborative filtering is a popular recommendation model that uses the user's past behavior and preferences to predict their future behavior and preferences. It works by finding similarities between users or items based on their past behavior and preferences and using these similarities to make recommendations. Collaborative filtering can be further divided into two types: user-based and item-based. User-based collaborative filtering finds similar users based on their past behavior and preferences and recommends items that these similar users have liked. Item-based collaborative filter-

ing finds similar items based on their past behavior and preferences and recommends items that are similar to the ones the user has liked.

Content-based filtering is another popular recommendation model that uses the user's past behavior and preferences as well as the content of the items to make recommendations. It works by analyzing the content of the items and finding similarities between them based on their features. It then recommends items that are similar to the ones the user has liked based on their content.

Hybrid models combine collaborative filtering and content-based filtering to make recommendations. They use both the user's past behavior and preferences as well as the content of the items to make recommendations.

Once the recommendation model is selected, it needs to be trained using the extracted features. The model's performance can be evaluated using metrics such as precision, recall, and F1 score, which are used to measure the model's accuracy and effectiveness. The model can be fine-tuned based on the evaluation results to improve its performance. Once the model is trained and evaluated, it can be deployed in a production environment, which is the final step in the process.

1.3.5 Evaluate the model:

Evaluating the model is a critical step in designing a user recommendation system. Once the model is trained using the extracted features, its performance needs to be evaluated using appropriate metrics. The evaluation helps to identify areas for improvement and fine-tune the model parameters

to improve its performance.

There are several metrics that can be used to evaluate the performance of a recommendation model, including precision, recall, and F1 score. Precision measures the proportion of recommended items that are relevant to the user, while recall measures the proportion of relevant items that are recommended to the user. F1 score is the harmonic mean of precision and recall and provides a balanced measure of the model's performance.

Cross-validation is a common technique used to evaluate the performance of a recommendation model. It involves splitting the data into training and testing sets and evaluating the model's performance on the testing set. This process is repeated several times, with different splits of the data, to obtain an average performance score.

Once the model's performance is evaluated, it can be fine-tuned based on the evaluation results to improve its performance. This may involve adjusting the model's parameters, selecting a different recommendation model, or using a different set of features. The goal is to create a model that can provide accurate and relevant recommendations to users, thereby improving their overall experience and satisfaction.

Finally, once the model is trained and evaluated, it can be deployed in a production environment, which is the final step in the process.

1.3.6 Refine and improve the model:

Refining and improving the model is an iterative process that involves fine-tuning the model's parameters, selecting a different recommendation model, or using a different set of features to improve its performance. The goal is to create a model that can provide accurate and relevant recommendations to users, thereby improving their overall experience and satisfaction.

One way to refine and improve the model is to adjust its parameters. For example, in collaborative filtering, the number of neighbors used to make recommendations can be adjusted to improve the model's performance. Similarly, in content-based filtering, the weight given to different features can be adjusted to improve the model's performance.

Another way to refine and improve the model is to select a different recommendation model. For example, if collaborative filtering is not performing well, content-based filtering or hybrid models can be used instead. Similarly, if content-based filtering is not performing well, collaborative filtering or hybrid models can be used instead.

Finally, using a different set of features can also help to refine and improve the model. For example, if the current set of features is not capturing the user's preferences and behavior effectively, additional features can be added or existing features can be modified to improve the model's performance.

The refinement and improvement process should be iterative, with the model's performance evaluated after each iteration. This helps to identify areas for improvement and fine-tune the model parameters to improve its

performance. The goal is to create a model that can provide accurate and relevant recommendations to users, thereby improving their overall experience and satisfaction.

1.4 Significance of User Recommendation Systems

User recommendation systems have become increasingly significant in recent years due to their ability to provide personalized and relevant recommendations to users. These systems use algorithms and machine learning techniques to analyze user data and behavior, and provide recommendations based on their preferences and interests. The significance of user recommendation systems lies in their ability to improve the user experience, increase sales, and efficiently allocate resources. By providing personalized recommendations, user recommendation systems can increase user engagement and satisfaction, leading to improved customer loyalty and revenue for businesses. Additionally, these systems can help businesses to efficiently allocate resources by recommending items that are likely to be of interest to users, reducing waste and improving efficiency. Overall, user recommendation systems have become a crucial tool for businesses looking to provide a more personalized and engaging user experience, and gain a competitive advantage in the market.recommendation systems[5, 6]:

1.4.1 Increased User Engagement:

User recommendation systems can significantly increase user engagement by providing personalized and relevant recommendations to users. By analyzing user data and behavior, these systems can recommend items that are likely to be of interest to users, leading to increased user engagement and satisfaction. For example, a user who is interested in action movies may be recommended similar movies or TV shows, leading to increased engagement and time spent on the platform. Additionally, user recommendation systems can help users discover new content that they may not have otherwise found, leading to increased exploration and engagement. By providing a more personalized and engaging user experience, user recommendation systems can improve user retention and loyalty, leading to increased revenue and growth for businesses. Overall, increased user engagement is a significant benefit of user recommendation systems, and can lead to improved user satisfaction, loyalty, and revenue for businesses.

1.4.2 Improved User Satisfaction:

User recommendation systems can significantly improve user satisfaction by providing personalized and relevant recommendations to users. By analyzing user data and behavior, these systems can recommend items that are likely to be of interest to users, leading to increased user satisfaction. For example, a user who is interested in cooking may be recommended recipes or cooking videos that match their preferences, leading to a more satisfying user experience. Additionally, user recommendation systems can help users

discover new content that they may not have otherwise found, leading to increased exploration and satisfaction. By providing a more personalized and satisfying user experience, user recommendation systems can improve user retention and loyalty, leading to increased revenue and growth for businesses. Overall, improved user satisfaction is a significant benefit of user recommendation systems, and can lead to increased user engagement, loyalty, and revenue for businesses.

1.4.3 Enhanced User Experience:

User recommendation systems can significantly enhance the user experience by providing personalized and relevant recommendations to users. By analyzing user data and behavior, these systems can recommend items that are likely to be of interest to users, leading to a more engaging and enjoyable experience. For example, a user who is interested in fashion may be recommended clothing items or accessories that match their preferences, leading to a more personalized and enjoyable shopping experience. Additionally, user recommendation systems can help users discover new content that they may not have otherwise found, leading to increased exploration and enjoyment. By providing a more personalized and enjoyable user experience, user recommendation systems can improve user retention and loyalty, leading to increased revenue and growth for businesses. Overall, enhanced user experience is a significant benefit of user recommendation systems, and can lead to increased user satisfaction, engagement, and revenue for businesses.

1.4.4 Increased Revenue:

User recommendation systems can significantly increase revenue for businesses by providing personalized and relevant recommendations to users. By analyzing user data and behavior, these systems can recommend items that are likely to be of interest to users, leading to increased sales and revenue. For example, a user who is interested in fitness may be recommended workout equipment or supplements that match their preferences, leading to increased sales and revenue for the business. Additionally, user recommendation systems can help businesses to efficiently allocate resources by recommending items that are likely to be of interest to users, reducing waste and improving efficiency. By providing a more personalized and engaging user experience, user recommendation systems can improve user retention and loyalty, leading to increased revenue and growth for businesses. Overall, increased revenue is a significant benefit of user recommendation systems, and can lead to improved profitability and success for businesses.

1.4.5 Improved Business Efficiency:

User recommendation systems can significantly improve business efficiency by providing personalized and relevant recommendations to users. By analyzing user data and behavior, these systems can recommend items that are likely to be of interest to users, leading to more efficient allocation of resources. For example, a business that sells clothing may use a recommendation system to recommend items that are likely to be of interest to users, reducing waste and improving inventory management. Additionally,

user recommendation systems can help businesses to identify trends and patterns in user behavior, leading to more informed decision-making and improved efficiency. By providing a more personalized and efficient user experience, user recommendation systems can improve user retention and loyalty, leading to increased revenue and growth for businesses. Overall, improved business efficiency is a significant benefit of user recommendation systems, and can lead to improved profitability and success for businesses.

1.5 Types of Recommender Systems

Each type of recommendation system has its own strengths and weaknesses and is suited to different types of data and use cases. Many recommendation systems use a combination of these types to generate the most accurate and relevant recommendations[7, 8].

1.5.1 Collaborative Filtering Recommendation System

Collaborative Filtering Recommendation System is a type of recommendation system that utilizes the past behavior and preferences of a group of users to make recommendations to a specific user. It is based on the assumption that users who have similar preferences in the past are likely to have similar preferences in the future. In collaborative filtering, the system collects data about users' past interactions with items, such as ratings or purchase history. Then, it identifies groups of users who have similar preferences based on their interactions with these items. The system uses this

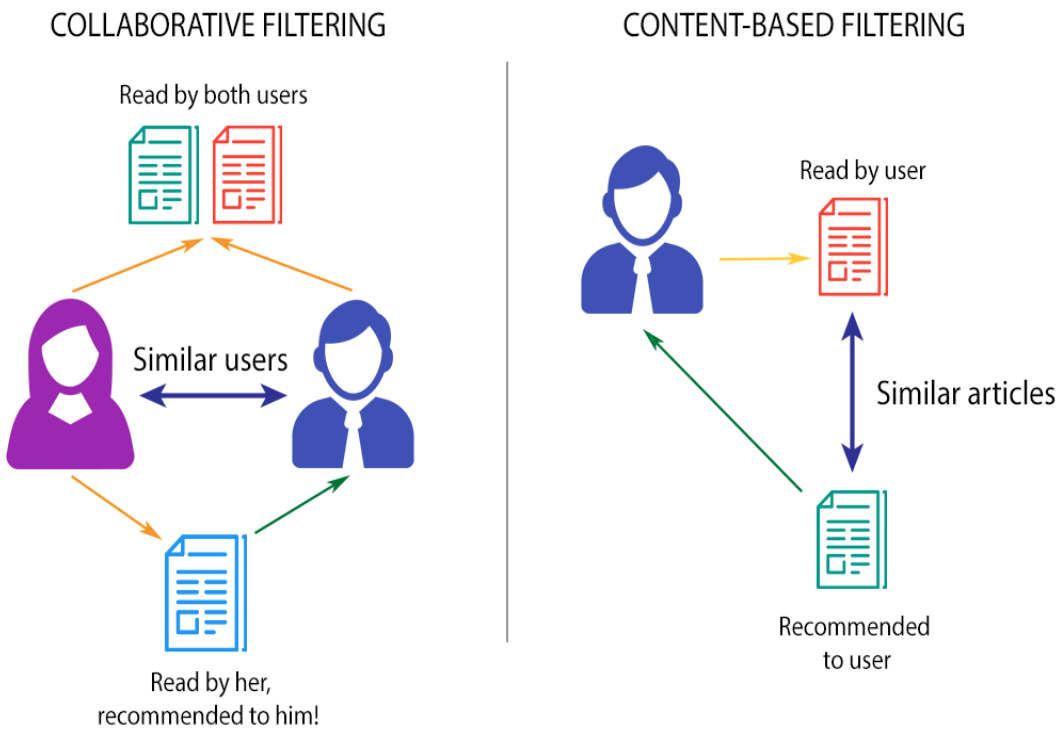

Figure 1.1: Types of Recommender Systems
[9]

information to recommend items that the specific user has not yet interacted with, but that other similar users have shown a preference for. There are two main types of collaborative filtering[10]:

- **User based collaborative filtering**

  User based collaborative filtering is a type of recommendation model that uses the user's past behavior and preferences to predict their future behavior and preferences. It works by finding similarities between users based on their past behavior and preferences and using these similarities to make recommendations.

  The user based collaborative filtering algorithm involves several steps. First, user data is collected, such as user profiles, user behavior, and user feedback. Next, the similarity between users is calculated based

on their past behavior and preferences using various similarity metrics, such as cosine similarity or Pearson correlation. Then, the k most similar users to the target user are found based on their similarity scores. Finally, items that the similar users have liked but the target user has not yet interacted with are recommended.

User based collaborative filtering has several advantages, including its simplicity and ease of implementation. It can also handle new items well, as long as there are enough users who have interacted with the item. However, it can suffer from the cold start problem, where new users or items have no or limited interaction data, making it difficult to make accurate recommendations.

To implement user-based collaborative filtering, various libraries and frameworks can be used, such as the Surprise library in Python. The model's performance can be evaluated using metrics such as precision, recall, and F1 score, and the model can be fine-tuned based on the evaluation results to improve its performance. Ultimately, the success of the recommendation system will depend on its ability to meet the user's needs and provide a personalized experience.

- **Item based collaborative filtering**

  This approach recommends items to a specific user based on the similarity between the items they have interacted with in the past.

  Collaborative filtering is a popular approach in recommendation systems because it does not require explicit knowledge of the items being

recommended, and it can work well even with sparse data. However, it also has some limitations, such as the cold start problem, where new users or items with no prior interactions have no information available to make recommendations.

The method of filtering of the data which has been given above can be termed as collaborative filtering method. This method requires the use of certain training models which makes this task easier. In order to find out a single parameter for implementing training model, one can take a look at the number of features in the dataset. The next step is to take into consideration the item's category where one does not need to label it, but to be aware of the item number. However, there are two sub-parameters for the training model also, including repetitions and rate of learning. The parameter of the learning rate includes knowing the parameter's variation of the rate which correlate strongly with these repetitions.

1.5.2 Content Based Filtering :

Content based filtering is a type of recommendation system that utilizes the features or attributes of items to make recommendations to users. The system recommends items that are similar in content to items that the user has already shown interest in. In content-based filtering, the system collects data about the attributes of items, such as keywords, genres, or descriptions, and uses this information to create a profile of the user's preferences. Then, the system recommends items that have similar attributes to those that the user has previously shown an interest in. Content-based filtering is partic-

ularly useful when there is a lot of information about the items available, and when the user's preferences are well-defined. It can also be helpful in situations where new users or items with no prior interactions are involved, as it can provide recommendations based on the characteristics of the items themselves. One limitation of content-based filtering is that it can lead to recommendations that are too similar to items the user has already interacted with, limiting the diversity of recommendations. Additionally, it may not capture other factors that could influence a user's preferences, such as social context or novelty[11, 12].

1.5.3 Hybrid Recommendation System

A hybrid recommendation system is a type of recommendation system that combines two or more different recommendation approaches to provide more accurate and relevant recommendations. The goal of a hybrid recommendation system is to overcome the limitations of individual recommendation approaches and provide more personalized and diverse recommendations. There are two main types of hybrid recommendation systems[13, 14]:

- **Weighted hybrid**

  Weighted hybrid This approach combines the recommendations generated by different algorithms with different weights assigned to each recommendation. The weights are based on the performance of the individual algorithms for a specific user or item.

- **Switching hybrid**

  Switching hybrid This approach selects the best algorithm for a particular user or item based on certain criteria. For example, the system may switch between collaborative filtering and content-based filtering depending on the amount of data available for a specific user or item.

  Hybrid recommendation systems can provide more accurate and diverse recommendations compared to individual recommendation approaches. However, designing a hybrid recommendation system can be complex, and determining the best combination of algorithms can require extensive testing and experimentation.

1.6 Disadvantages of recommendation systems :

While there are many benefits to recommendation systems, there are also several disadvantages that need to be considered, including[15, 16]:

- **Lack of diversity**

  Some recommendation systems may provide recommendations that are too similar to the user's past behavior, leading to a lack of diversity in recommendations. This can limit a user's exposure to new and different content, products, or ideas.

- **Limited data**

  Recommendation systems may not work well if there is not enough data on the user or item to make accurate recommendations. This can be particularly problematic for new users who have not yet provided

enough data to the system.

- **Limited explanation**

  Some recommendation systems may not provide clear explanations for why a certain item is being recommended, leading to user mistrust or confusion. Users may be hesitant to follow a recommendation if they do not understand why it was made.

- **Filter bubble**

  Recommendation systems may reinforce a user's existing preferences and limit exposure to new ideas or perspectives. This can lead to a lack of diversity in the user's online experience and limit their exposure to new and different content.

- **Privacy concerns**

  Recommendation systems may collect sensitive user data, leading to privacy concerns and potential misuse of user data. Users may be hesitant to provide personal information to the system if they are not confident in the security and privacy measures in place.

- **Bias**

  Recommendation systems may unintentionally introduce bias into the recommendations they provide. This can happen if the system is trained on biased data or if the algorithms used to make recommendations are themselves biased.

1.7 Evaluation of Recommender Systems

Evaluation of recommender systems is an important step in determining the effectiveness and accuracy of the system. There are several methods for evaluating recommender systems, including[17, 18]:

1.7.1 Offline evaluation

Offline evaluation is a method of evaluating the performance of a recommender system using historical data that is already available. This involves testing the recommendation algorithm on a dataset that contains user-item interactions, such as ratings or clicks, and comparing the system's predictions with the actual user behavior. The performance of the recommendation algorithm is typically evaluated using metrics such as mean absolute error (MAE), root mean squared error (RMSE), precision, recall, and F1 score. These metrics provide a quantitative measure of how well the system is performing in terms of accuracy, coverage, and other factors. Offline evaluation is often used in research settings to test the performance of different recommendation algorithms and approaches. While this method provides a controlled and repeatable environment for testing, it may not always be indicative of real-world performance since the dataset used for evaluation may not fully capture the complexities and nuances of user behavior in the real world. Therefore, it is important to complement offline evaluation with other evaluation methods, such as online evaluation and user studies, to get a more complete and accurate picture of the system's performance.

- MAE : calculates the difference between the predicted grades and the real grades
- RMSE : : it is based on the same principle as the MAE, while emphasizing the significant differences

1.7.2 Online evaluation:

Online evaluation is a method of evaluating the performance of a recommender system by deploying it in a live environment and measuring its performance in real-time. This involves collecting user feedback and interaction data from the live system, and using it to evaluate the system's performance. Online evaluation provides a more accurate and real-world performance metric than offline evaluation because it captures the complexity of real-world user behavior and interactions. However, online evaluation can be more challenging to implement because it requires the system to be fully deployed and integrated with the production environment. It also requires a large user base and a significant amount of time to collect sufficient data for evaluation. Online evaluation metrics include click-through rates, conversion rates, and engagement rates. These metrics measure how often users interact with the recommended items, how many of these interactions lead to a conversion, and how engaged the user is with the system overall. Online evaluation is often used to fine-tune and optimize the performance of a recommender system in a live environment. It can also be used to compare the performance of different recommendation algorithms or approaches and to test new features or updates to the system.

1.7.3 User Studies

User studies are a method of evaluating the performance of a recommender system by collecting feedback directly from users. This involves conducting surveys, interviews, or focus groups with users to understand their opinions, preferences, and experiences with the system. User studies can provide valuable insights into the user experience and help identify areas for improvement. They can also help evaluate the effectiveness of the system in terms of user satisfaction, trust, and engagement. User studies can be conducted at different stages of the system development process, from initial design and testing to post-deployment evaluation. User studies can provide both qualitative and quantitative feedback. Qualitative feedback includes open-ended questions that allow users to provide their opinions, suggestions, and criticisms about the system. Quantitative feedback includes surveys or rating scales that allow users to rate their satisfaction with the system, its ease of use, and other factors. User studies are important for evaluating the effectiveness and user-friendliness of a recommender system, as well as identifying potential areas for improvement. They can also help ensure that the system is meeting the needs and expectations of its users.

1.7.4 A/B testing

A/B testing is a method of evaluating the performance of a recommender system by conducting a randomized experiment in which two or more variants of the system are tested against each other to determine which performs better. In an A/B test, a randomly selected group of users is shown one

variant of the system, while another randomly selected group of users is shown a different variant. For example, in a recommender system, an A/B test might involve testing two different algorithms to see which one generates more clicks or conversions. The system would randomly assign users to one of the two algorithms and measure their interactions and behavior to determine which algorithm performs better A/B testing allows for the evaluation of the system's performance in a controlled and statistically rigorous manner. It can help identify which variant of the system produces better results and can be used to fine-tune and optimize the system's performance. A/B testing can also be used to test the effectiveness of new features or updates to the system. By randomly assigning users to different variants of the system, A/B testing can help determine whether a new feature or update has a positive impact on the system's performance. When evaluating a recommender system, it is important to consider several factors, including accuracy, coverage, diversity, novelty, serendipity, and user satisfaction. Accuracy refers to the ability of the system to make accurate recommendations, while coverage refers to the proportion of items or content that the system can recommend. Diversity refers to the variety of recommendations provided, while novelty refers to the degree to which the recommendations are new or unexpected. Serendipity refers to the ability of the system to provide recommendations that the user would not have found otherwise, while user satisfaction refers to the user's overall satisfaction with the recommendations provided.

1.8 Conclusion

In conclusion, recommendation systems are a powerful tool for businesses and organizations to improve customer engagement, satisfaction, and retention. By leveraging machine learning algorithms, recommendation systems can analyze large amounts of data to provide personalized recommendations to users based on their preferences and behavior.

However, it's important to note that recommendation systems are not one-size-fits-all solutions. The effectiveness of a recommendation system depends on the quality of the data and the choice of algorithm, as well as the user interface and overall user experience.

To build an effective recommendation system, businesses should start by collecting and analyzing high-quality data, including user behavior and preferences. They should also consider the user experience and make sure that the recommendations are relevant, timely, and easy to understand.

Furthermore, businesses should continuously monitor and improve the performance of the recommendation system over time. This can involve testing different algorithms and parameters, gathering user feedback, and measuring the impact on key metrics such as engagement and revenue. recommendation systems can be a valuable asset for businesses looking to improve customer satisfaction and drive growth, but they require careful planning and execution to be effective.

CHAPTER 2

MACHINE LEARNING

2.1 Introduction

Machine learning is a subfield of artificial intelligence (AI) that deals with the development of algorithms and models that enable computers to learn from data without being explicitly programmed. Machine learning algorithms are designed to automatically improve their performance in a task by analyzing and learning from data. The goal of *machine learning* is to build models that can generalize from the examples they have seen and make accurate predictions on new, unseen data. To achieve this, machine learning algorithms use statistical techniques to identify patterns in the data, and they adjust their model parameters to minimize errors in their predictions. There are different types of *machine learning*, including supervised learning, unsupervised learning, and reinforcement learning. Supervised learning involves training a model with labeled data, where the correct output is provided for each input. Unsupervised learning involves training a model with unlabeled data, where the model learns to find patterns and structure in the

data. Reinforcement learning involves training a model to make decisions based on feedback from its environment, where the goal is to maximize a reward signal. *Machine learning* has numerous applications, including image and speech recognition,natural language processing, recommender systems, fraud detection, and autonomous vehicles. The field of *machine learning* is constantly evolving, with new,algorithms, techniques, and tools being developed to improve the performance and efficiency of machine learning models.

2.2 The Concept Of Artificial Intelligence

Artificial intelligence (AI) refers to the development of computer systems that can perform tasks that typically require human intelligence, such as learning, problem solving, decision making, and language understanding. AI systems are designed to analyze data, recognize patterns, make predictions, and take actions based on their analyses, without being explicitly programmed to do so. There are different types of AI, including rule-based systems, machine learning, deep learning, and natural language processing. Rule-based systems use if-then rules to make decisions and are limited by the complexity of the rules. Machine learning involves training a model with data to recognize patterns and make predictions, and deep learning uses neural networks to recognize complex patterns in large amounts of data. Natural language processing allows machines to understand and generate human language. AI has a wide range of applications, from virtual

assistants like Siri and Alexa to self-driving cars, medical diagnosis systems, and fraud detection in banking. The field of AI is rapidly evolving, and new advancements are being made in areas like explainable AI, which aims to make AI systems more transparent and understandable to humans, and ethical AI, which seeks to address the ethical implications of AI development and deployment[19, 20].

2.2.1 Machine Learning in Artificial Intelligence

Machine learning is a subfield of artificial intelligence (AI) that involves the development of algorithms that can learn from data and improve their performance over time. In traditional programming, developers write code that specifies how a computer should perform a task. In contrast, with *Machine Learning*, developers train a model using data, and the model learns to recognize patterns and make predictions based on that data.

There are three main types of *Machine Learning*: supervised learning, unsupervised learning, and reinforcement learning. Supervised learning involves training a model with labeled data, where the correct output is provided for each input. Unsupervised learning involves training a model with unlabeled data, where the model learns to find patterns and structure in the data. Reinforcement learning involves training a model to make decisions based on feedback from its environment, where the goal is to maximize a reward signal.

Machine learning has numerous applications, including image and speech recognition, natural language processing, recommender systems, fraud de-

tection, and autonomous vehicles. The field of machine learning is constantly evolving, with new algorithms, techniques, and tools being developed to improve the performance and efficiency of *Machine Learning* models[21].

2.2.2 Machine Learning Types

There are three main types of machine learning: supervised learning, unsupervised learning, and reinforcement learning[22, 23].

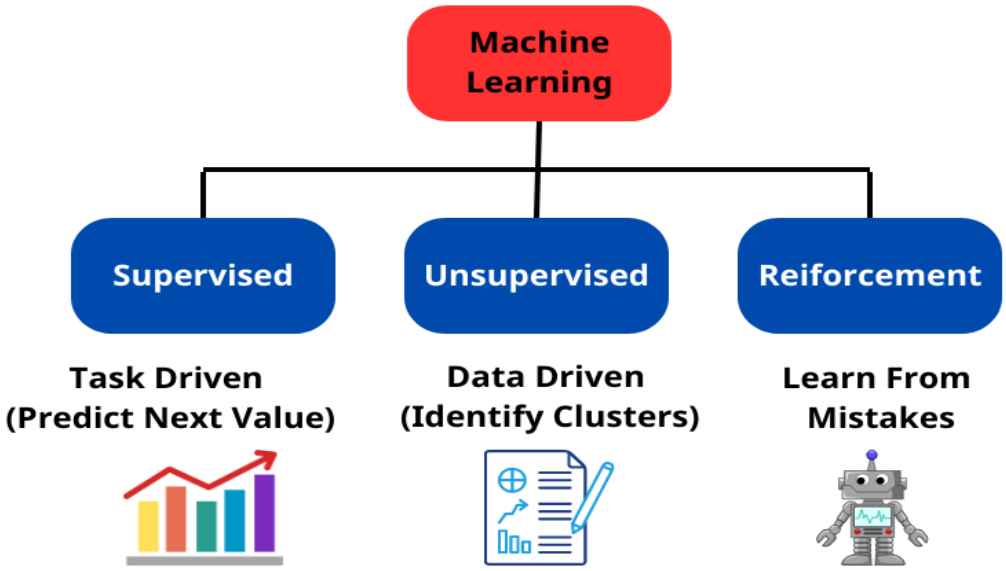

Figure 2.1: Machine Learning Types

2.2.3 Supervised Machine Learning

Supervised machine learning is a type of machine learning in which a model is trained on a labeled dataset, where the correct output is provided for each input. The goal of supervised learning is to build a model that can generalize to new, unseen data and make accurate predictions on new examples. In supervised learning, the labeled dataset is divided into two parts: a training set and a validation set. The training set is used to train the model, while the validation set is used to evaluate the performance of the model on unseen examples.

There are two main types of supervised learning: regression and classification[24, 25].

1. **Regression**

   Regression is a type of supervised learning that involves predicting a continuous output variable. The goal of regression is to build a model that can accurately predict the value of the output variable for new, unseen examples. Examples of regression include predicting house prices based on their features, or predicting the stock market prices.

2. **Classification**

   Classification is a type of supervised learning that involves predicting a discrete output variable. The goal of classification is to build a model that can accurately predict the class of the output variable for new, unseen examples. Examples of classification include detecting whether an email is spam or not, or identifying the species of an animal based

on its features. Supervised learning algorithms include linear regression, logistic regression, decision trees, random forests, support vector machines, and neural networks. The choice of algorithm depends on the specific problem at hand and the nature of the dataset

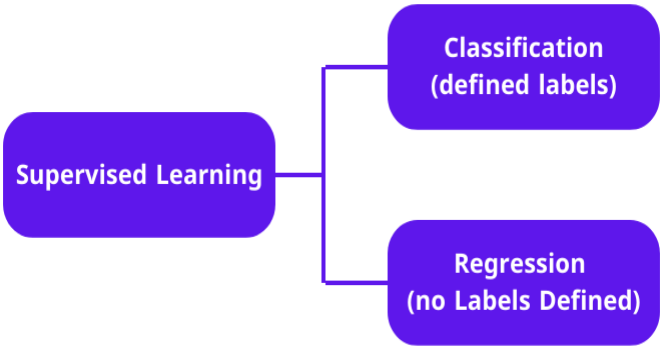

Figure 2.2: Supervised Form of Machine Learning

2.2.4 Unsupervised Machine Learning

Unsupervised machine learning is a type of machine learning in which a model is trained on an unlabeled dataset, where the model learns to find patterns and structure in the data without any pre-existing knowledge of the correct output. The goal of unsupervised learning is to discover hidden patterns, relationships, and structures within the data. There are two main types of unsupervised learning: clustering and dimensionality reduction[25].

1. **Clustering**

   Clustering is a type of unsupervised learning that involves grouping similar data points together into clusters. The goal of clustering is to identify meaningful groups within the data, without any prior knowl-

edge of the groups. Examples of clustering include customer segmentation in marketing, or grouping genes that have similar expression patterns in bioinformatics.

2. **Dimensionality Reduction**

Dimensionality reduction is a type of unsupervised learning that involves reducing the number of variables in the dataset while retaining the most important information. The goal of dimensionality reduction is to simplify the data and make it easier to visualize and analyze. Examples of dimensionality reduction include principal component analysis (PCA) and t-distributed stochastic neighbor embedding (t-SNE). Unsupervised learning algorithms include k-means clustering, hierarchical clustering, principal component analysis (PCA), and t-distributed stochastic neighbor embedding (t-SNE). The choice of algorithm depends on the specific problem at hand and the nature of the dataset.

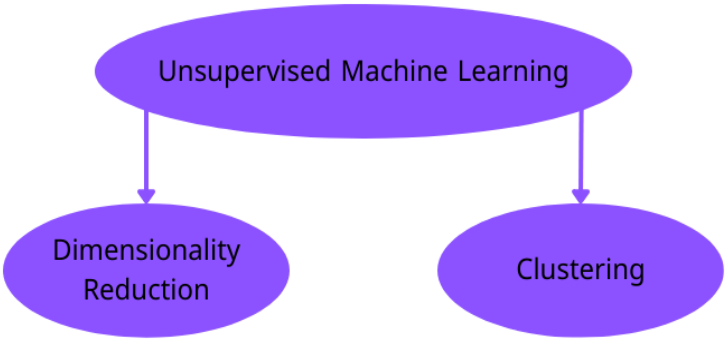

Figure 2.3: Unsupervised Form of Machine Learning

2.2.5 Reinforcement Machine Learning

Reinforcement learning is a type of machine learning that involves training a model to make decisions based on feedback from its environment, where the goal is to maximize a reward signal. In reinforcement learning, the model learns through trial and error by interacting with an environment and receiving feedback in the form of rewards or penalties. The main components of a reinforcement learning problem are[25]:

1. Agent : The agent is the entity that takes actions in the environment.
2. Environment: The environment is the external system with which the agent interacts.
3. State : The state is the current situation of the agent within the environment.
4. Action : The action is the decision made by the agent based on the current state.
5. Reward : The reward is the feedback provided to the agent based on its actions in the environment.

The goal of the agent is to learn a policy, which is a mapping from states to actions, that maximizes the cumulative reward over time. Reinforcement learning has been successfully applied to a variety of problems, such as playing games, robotics, and autonomous driving. Some popular reinforcement learning algorithms include Q-learning, SARSA, and deep reinforcement

learning. One of the challenges of reinforcement learning is the exploration-exploitation trade-off, where the agent must balance between trying out new actions to learn more about the environment and exploiting the actions that have led to high rewards in the past.

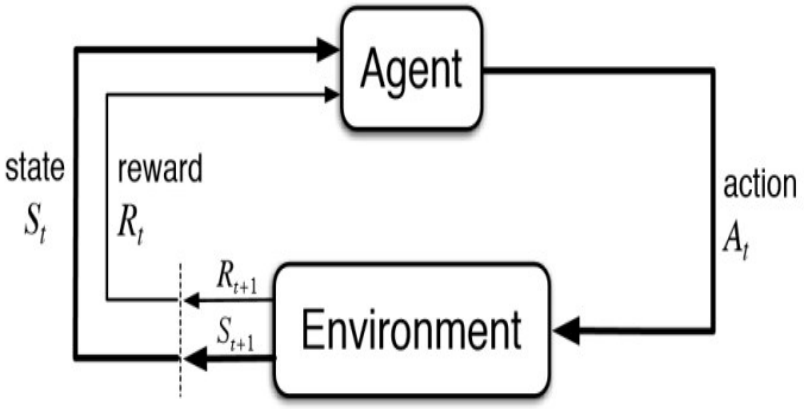

Figure 2.4: Reinforcement Learning Framework

2.2.6 Machine Learning Algorithm :

A machine learning algorithm is a set of rules and procedures that a machine learning model uses to learn from data and make predictions or decisions. Machine learning algorithms can be broadly categorized into supervised, unsupervised, and reinforcement learning.

Supervised learning algorithms are used when the target variable is known and the model is trained on labeled data. The goal is to learn a mapping between the input variables and the target variable. Examples of supervised learning algorithms include linear regression, logistic regression, decision trees, random forests, support vector machines (SVMs), and neural networks[24, 25].

Unsupervised learning algorithms are used when the target variable is

unknown and the model is trained on unlabeled data. The goal is to learn the underlying structure and patterns in the data. Examples of unsupervised learning algorithms include clustering algorithms, principal component analysis (PCA), and t-distributed stochastic neighbor embedding (t-SNE).

Reinforcement learning algorithms are used when the model learns through trial and error by interacting with an environment and receiving feedback in the form of rewards or penalties. The goal is to learn a policy that maximizes the cumulative reward over time. Examples of reinforcement learning algorithms include Q-learning, SARSA, and deep reinforcement learning.

The choice of algorithm depends on the specific problem at hand, the nature of the data, and the goals of the analysis. Each algorithm has its strengths and weaknesses, and selecting the right algorithm is an important part of building an effective machine learning model.

There are many different machine learning algorithms, Here are some common types of machine learning algorithms:

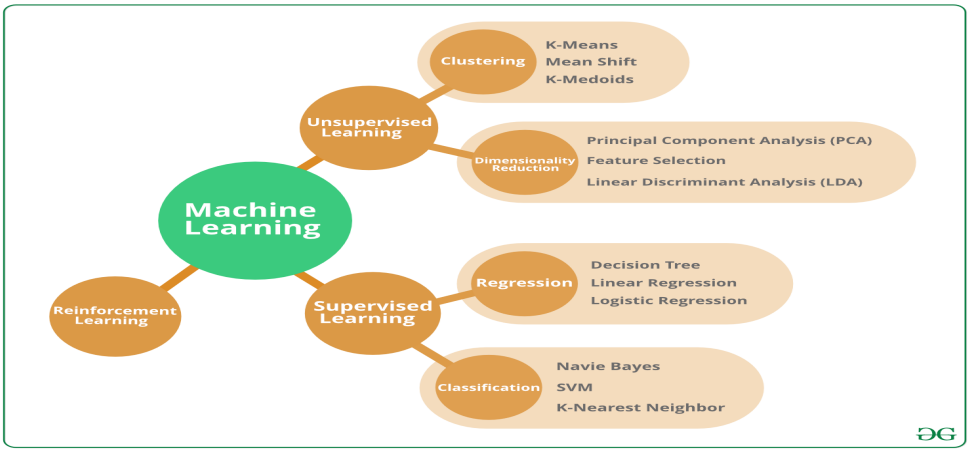

Figure 2.5: Machne Learning Algorithm
[26]

- **Decision Trees**

  A decision tree is a type of supervised learning algorithm used in machine learning and data mining. It is a graphical representation of all the possible solutions to a decision based on certain conditions.

  In a decision tree, each internal node represents a test on an attribute, each branch represents the outcome of the test, and each leaf node represents a class label. The decision tree is built using a set of training data that includes both input variables and output variables, and the tree is used to predict the class label of new, unseen data.

  The process of building a decision tree involves selecting the best attribute to split the data at each node, such that the resulting subsets of data are as pure as possible. There are several algorithms that can be used to build decision trees, including ID3, C4.5, and CART.

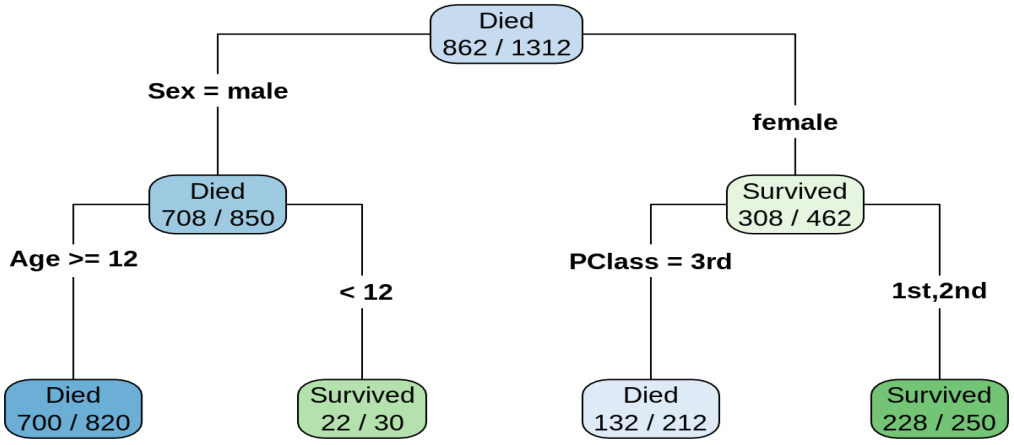

Figure 2.6: Decision Trees

- **Random Forest**

  Random Forest is a popular machine learning algorithm used for both classification and regression tasks. It is an ensemble learning method that builds multiple decision trees and combines their predictions to produce a final output.

  The basic idea behind Random Forest is to create a forest of decision trees, where each tree is built using a random subset of the features and a random subset of the training data. This helps to reduce overfitting and increase the generalization ability of the model. The trees in the forest are trained independently of each other, so the algorithm is also highly scalable.

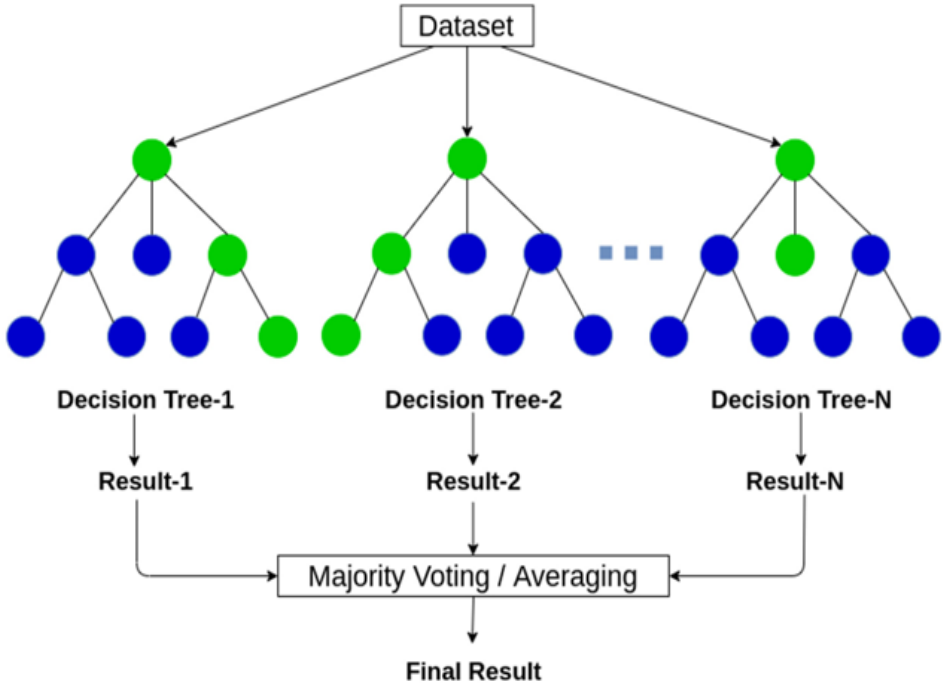

Figure 2.7: Random Forest Algorithm

- **Logistic Regression**

  Logistic regression is a popular machine learning algorithm used for classification tasks. It is a type of supervised learning algorithm that models the relationship between a dependent variable and one or more independent variables. In logistic regression, the dependent variable is binary, meaning it takes only two possible values, such as 0 or 1, Yes or No, or True or False.

  The goal of logistic regression is to find the best fit line or curve that separates the two classes in the data. This line or curve is called the decision boundary, and it represents the probability of the dependent variable being 1 for a given set of independent variables. The logistic function or sigmoid function is used to map the output of the linear equation to a probability value between 0 and 1.

  Logistic regression has several advantages over other machine learning algorithms, including:

  1. Simplicity Logistic regression is a simple and easy-to-understand algorithm that requires little data preparation.
  2. Interpretability: The model coefficients in logistic regression are easy to interpret, which can be useful for understanding the factors driving the predictions.
  3. Efficiency Logistic regression is a fast algorithm that can handle large datasets and perform well on high-dimensional feature spaces.
  4. Robustness: Logistic regression is less sensitive to noise and out-

liers in the data than other algorithms, making it more robust and reliable.

Logistic regression has been successfully applied to a wide range of applications, including finance, healthcare, and marketing. It is a popular choice for machine learning practitioners due to its simplicity, interpretability, and efficiency.

- **Support Vector Machines (SVMs)**

  Support Vector Machines (SVMs) are a type of supervised learning algorithm used in machine learning for classification and regression analysis. SVMs can be used for both linear and nonlinear data classification tasks.

  The goal of SVMs is to find the hyperplane in the feature space that best separates the classes of data. The hyperplane is chosen such that it maximizes the margin, which is the distance between the hyperplane and the closest points of each class. This approach is known as maximum margin classification.

  SVMs can be used with both linear and nonlinear kernel functions to handle data that is not linearly separable. The kernel function maps the input data to a higher-dimensional feature space, where it may be easier to find a hyperplane that separates the data.

  SVMs have several advantages, including their ability to handle high-dimensional data, their robustness to overfitting, and their ability to handle both linear and nonlinear data. However, they can be sensitive

to the choice of kernel function and the regularization parameter.

SVMs are widely used in various applications, such as text classification, image classification, and bioinformatics.

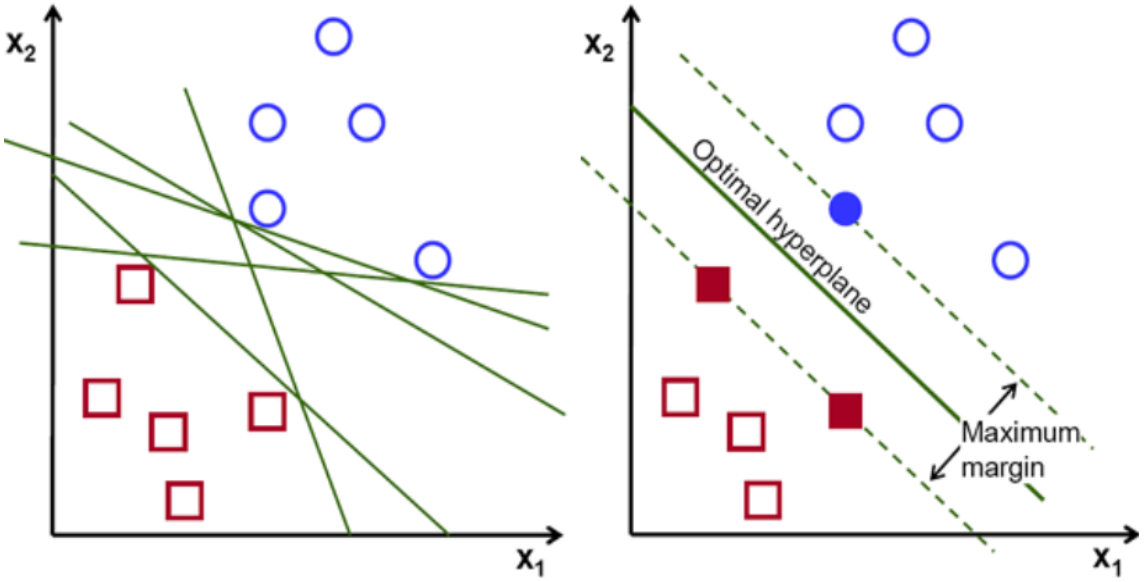

Figure 2.8: Support Vector Machines (SVMs)
[27]

- **Principal Component Analysis (PCA)**

  Principal Component Analysis (PCA) is a statistical technique used to reduce the dimensionality of high-dimensional data while retaining most of its important information. It is an unsupervised learning algorithm, which means it does not require labeled data.

  PCA works by transforming the data into a new coordinate system in which the first coordinate (principal component) captures the maximum variance in the data, the second coordinate captures the second maximum variance, and so on. The principal components are calculated by finding the eigenvectors of the covariance matrix of the data.

The number of principal components to retain is a hyperparameter that can be chosen based on the amount of variance that needs to be retained. The retained principal components can be used for visualization, data compression, and feature extraction.

PCA has several advantages, including its ability to reduce the dimensionality of high-dimensional data while retaining most of its important information. It can be used to identify patterns and relationships in the data, and it can also help in removing noise from the data. However, PCA can be sensitive to outliers and may not perform well if the data is not linearly separable.

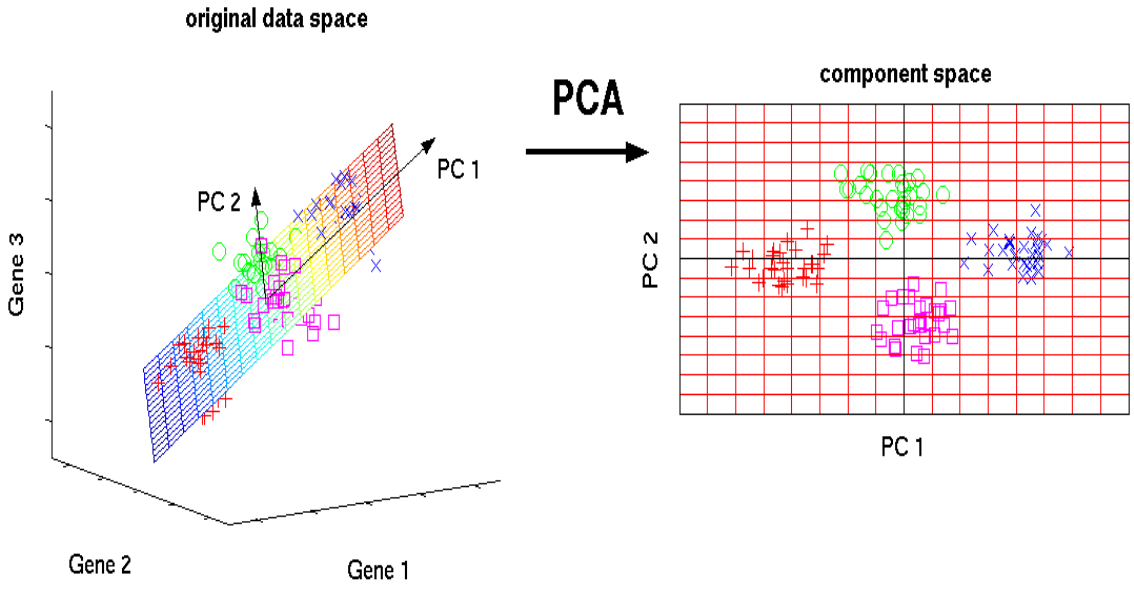

Figure 2.9: Principal Component Analysis (PCA) [28]

- **Naive Bayes**

  Naive Bayes is a probabilistic algorithm used in supervised learning for classification tasks. It is based on the Bayes theorem, which states that the probability of a hypothesis given some observed evidence is proportional to the likelihood of the evidence given the hypothesis multiplied by the prior probability of the hypothesis.

  In Naive Bayes, the hypothesis is the class label of a new data point, and the evidence is the features of the data point. The algorithm assumes that the features are conditionally independent given the class label, which is known as the "naive" assumption.

  Naive Bayes calculates the posterior probability of each class label for a new data point and assigns the label with the highest probability. The algorithm requires an estimate of the prior probabilities of the class labels and the conditional probabilities of the features given each class label. These probabilities can be estimated from the training data using maximum likelihood estimation or Bayesian estimation.

  Naive Bayes has several advantages, including its simplicity, efficiency, and robustness to irrelevant features. It can handle both continuous and categorical data, and it can work well with small training sets. However, it can suffer from the "zero-frequency problem" when a feature has no occurrences in a particular class, and it may not perform well if the "naive" assumption is violated.

$$\mathrm{P(A \mid B)} = \frac{\mathrm{P(B \mid A) * P(A)}}{\mathrm{P(B)}}$$

- $P(B \mid A)$ = probability that event B occurs if event A has already occurred
- $P(A)$ = probability that event A occurs
- $P(B)$ = probability that event B occurs

2.3 Significance of Machine Learning

Machine learning has significant importance in various fields due to its ability to learn from data and make accurate predictions or decisions. Here are some of the key significance of machine learning[29]:

1. Automation: Machine learning enables the automation of complex tasks and processes, reducing the need for human intervention and increasing efficiency.
2. Personalization: Machine learning algorithms can be used to personalize products, services, and content based on individual preferences and behavior.
3. Prediction: Machine learning can be used to make accurate predictions about future events or trends, such as predicting customer behavior or stock prices.
4. Optimization: Machine learning can be used to optimize processes and systems, such as reducing energy consumption or improving supply chain management.

5. Insights: Machine learning algorithms can be used to uncover insights and patterns in data that may not be visible to the human eye, enabling businesses to make better decisions.

6. Improved Decision Making: Machine learning can help decision-makers to make more informed and data-driven decisions by providing insights and recommendations based on data analysis.

7. Fraud Detection: Machine learning can be used to detect fraudulent behavior, such as credit card fraud, by analyzing patterns in transaction data.

8. Healthcare: Machine learning is being used to improve healthcare outcomes by analyzing patient data and providing personalized treatment recommendations.

2.4 Machine Learning for Recommendation System

Machine learning and recommendation systems are closely related, as machine learning techniques are often used to develop and improve recommendation systems. Recommendation systems use algorithms and statistical models to analyze user data and behavior, and provide personalized recommendations based on their preferences and interests. Machine learning techniques are used to train these algorithms and models, and to improve their accuracy and effectiveness over time.

One of the most common machine learning techniques used in recommendation systems is collaborative filtering. Collaborative filtering works

by finding users who have similar preferences and recommending items that those users have liked. Machine learning algorithms are used to identify these similarities and make predictions about which items a user is likely to be interested in. Another common technique is content-based filtering, which recommends items based on the features of the items themselves. Machine learning algorithms are used to analyze these features and make predictions about which items a user is likely to be interested in.

In addition to these techniques, machine learning is also used to improve the performance of recommendation systems over time. For example, machine learning algorithms can be used to analyze user feedback and ratings, and to update the recommendation model accordingly. This can help to improve the accuracy and relevance of the recommendations, leading to increased user engagement and satisfaction.

Overall, machine learning and recommendation systems are closely related, as machine learning techniques are used to develop and improve the algorithms and models that power recommendation systems. By using machine learning to analyze user data and behavior, recommendation systems can provide personalized and relevant recommendations to users, leading to increased user engagement, satisfaction, and revenue for businesses.

2.5 Conclusion

In conclusion, machine learning is a rapidly growing field that involves teaching computers to learn from data and make predictions or decisions without being explicitly programmed. It is a subset of artificial intelligence that has many applications across a variety of industries, from healthcare and finance to e-commerce and entertainment.

There are several types of machine learning, including supervised learning, unsupervised learning, and reinforcement learning. Each type has its own strengths and weaknesses, and the choice of algorithm depends on the specific problem being addressed.

Machine learning has several advantages, such as its ability to handle large and complex datasets, automate repetitive tasks, and make accurate predictions. It also has some limitations, such as the need for high-quality data, potential biases in the algorithms, and the potential for ethical issues.

To be successful in machine learning, it is important to have a good understanding of the underlying concepts and algorithms, as well as the ability to preprocess and analyze data, select appropriate features, and evaluate models. With the right skills and knowledge, machine learning can be a powerful tool for solving complex problems and driving innovation in many fields.

CHAPTER 3

RESULTS AND DISCUSSIONS

3.1 Introduction

Time aware circle based recommendation is a technique used in online social networks to suggest content or connections to users. It takes into consideration the user's social circle, which includes their friends and connections, and the timing of the recommendation. This method is helpful in enhancing the relevance and value of recommendations by customizing them to the user's present context and interests. Machine learning algorithms such as collaborative filtering or matrix factorization can be employed to implement this approach.

3.2 Proposition

3.2.1 K-Nearest Neighbors (KNN)

K-Nearest Neighbors (KNN) is a popular machine learning algorithm used for classification and regression tasks. It is a non-parametric algorithm that

works by finding the k-nearest data points in the training set to a given test data point and using their labels to predict the label of the test data point.

3.2.3.1 KNN Advantages

One of the main advantages of KNN is its simplicity and ease of implementation. It is a straightforward algorithm that can be easily understood and applied to a wide range of problems. Additionally, KNN is a lazy learning algorithm, which means that it does not require any training time and can make predictions in real-time. This makes it a suitable choice for online applications where data is constantly changing.

3.2.3.2 KNN Disadvantages

However, KNN also has some Disadvantages. One of the main disadvantages is its computational complexity. As the size of the training set grows, the time required to find the k-nearest neighbors increases, which can make the algorithm slow and inefficient. Additionally, KNN is sensitive to the choice of distance metric and the value of k, which can affect the accuracy of the predictions.

3.2.3.3 KNN For Time Aware Circle Base Recommendation in Online social Network

Despite its limitations, KNN is a useful algorithm for developing Time Aware Circle Base Recommendation in Online social Network. This is because it can be used to find similar users based on their social connections and activity patterns, which can be used to make personalized recommendations. By considering the temporal aspect of the data, KNN can also be used to make recommendations that are relevant to the current time period,

which can improve the quality of the recommendations.

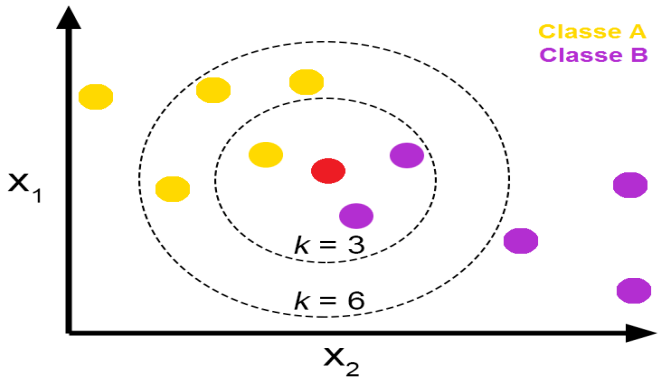

Figure 3.1: K-Nearest Neighbors (k-NN) [30]

For $k = 3$ the majority class of the central point is class B, but if we change the value from neighborhood $k = 6$ the majority class becomes class A.

3.2.2 Time aware trust Circle based

Time aware trust refers to a concept in which trust is established and maintained over time through repeated interactions and experiences. It involves building trust gradually and incrementally, based on a history of positive interactions and consistent behavior.

Circle based refers to a trust model in which trust is established within a specific group or community, such as a social network or professional organization. This model is based on the idea that people are more likely to trust those who are part of their social or professional circles, as they share common interests and values.

Therefore, Time aware trust Circle based refers to a trust model that combines the concepts of building trust over time and establishing trust within a specific group or community. This model emphasizes the impor-

tance of repeated positive interactions and consistent behavior within a specific social or professional circle in order to establish and maintain trust.

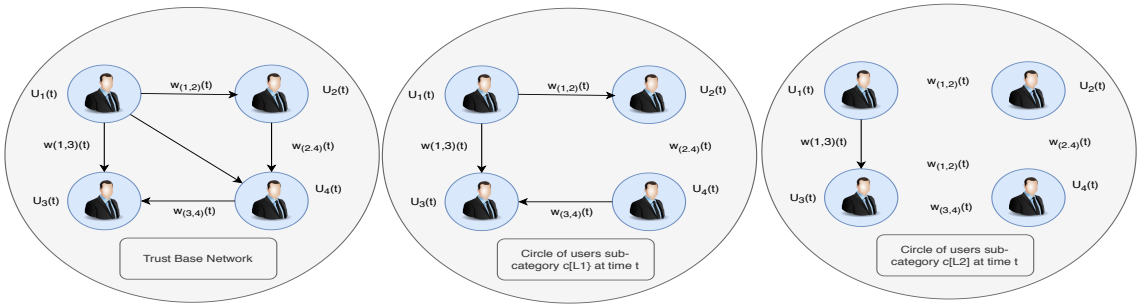

Figure 3.2: Time aware trust Circle based garph

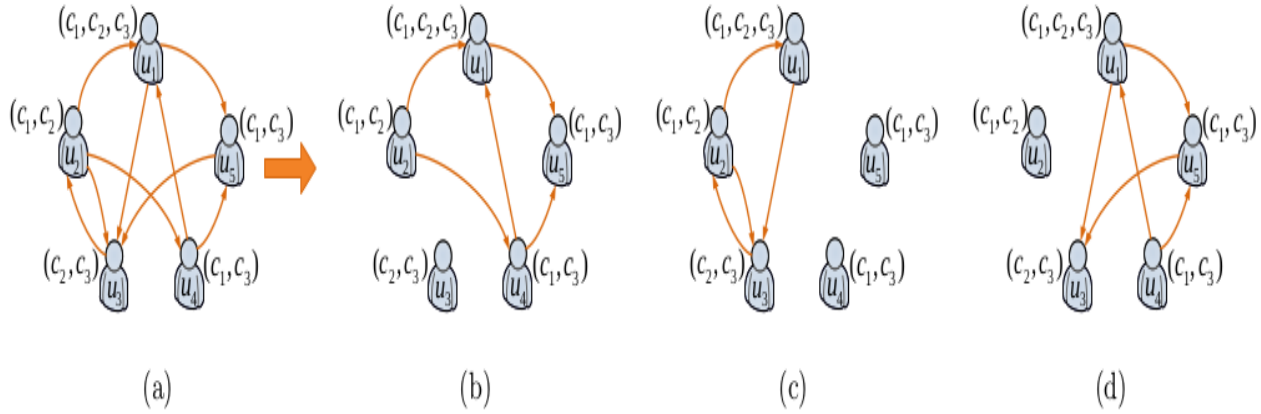

Figure 3.3: Illustration of inferred circles, each user is labeled with the categories in which she has ratings. a): the original social network; b), c) and d): inferred circles for categories c 1 , c 2 and c 3 respectively [1]

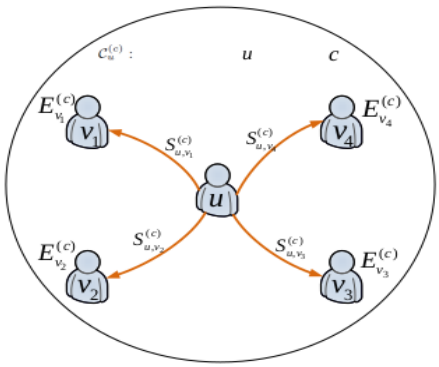

Figure 3.4: Illustration of expertise-based trust- assignment in category c. [1]

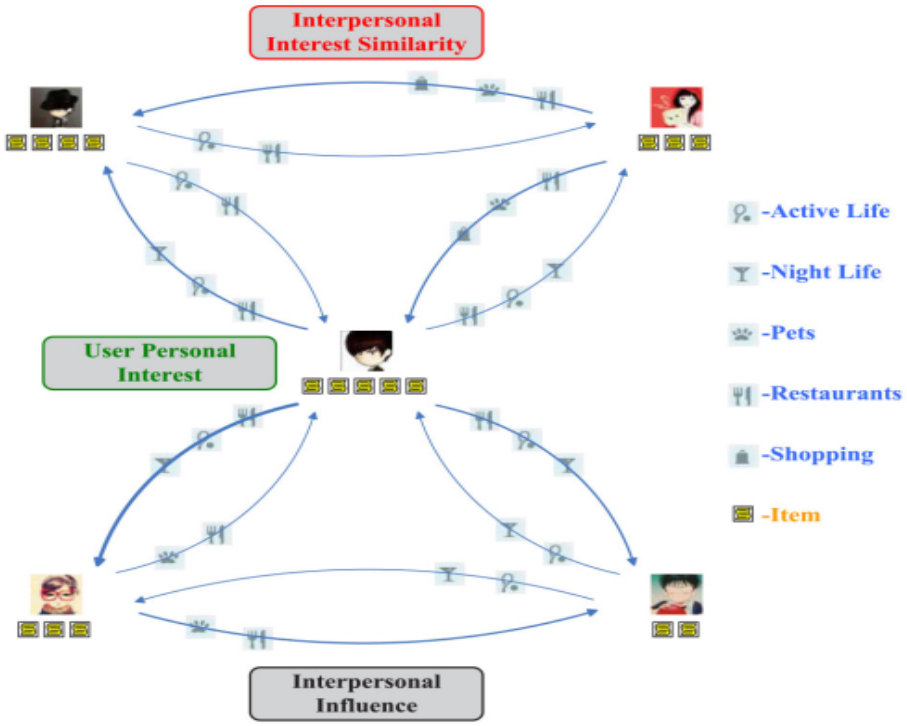

Figure 3.5: Three main social factors in our recommendation model, including user personal interest, interpersonal interest similarity, and interpersonal influence. The items under users are historical rating records, which can be used to mine users' personal interest. The category icon on line between two users denotes their interest similarity. And the boldness of the line between users indicates the strength of interpersonal influence.[2]

3.3 Mathematical formulas

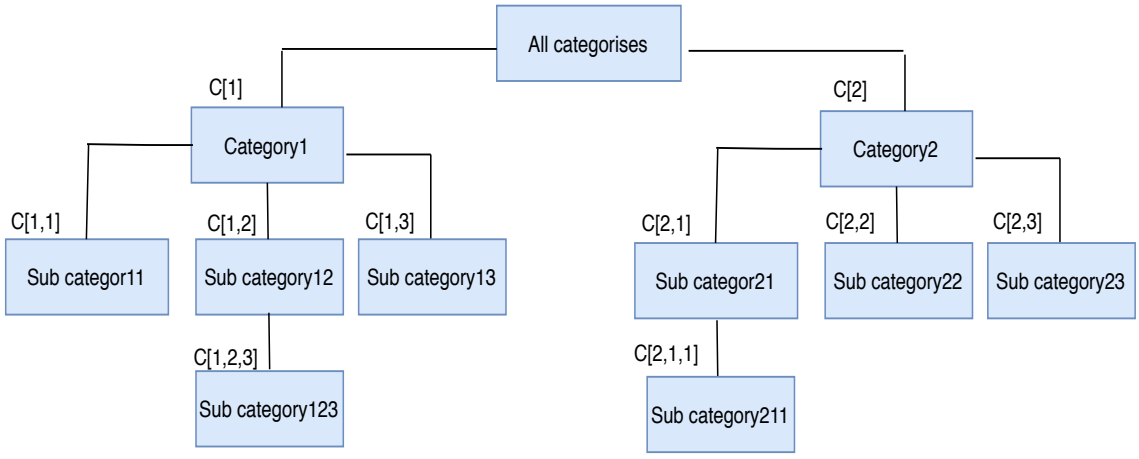

Figure 3.6: Hierarchical category/subcategory structure

The user can change their interest, thus, the user is related with time. This factor can add to our model the advantage of the user personal interest

of [2]. Moreover we can define another function witch can calculate the interpersonal interest similarity such as in [2]

$\forall$ i $\in$ N and j $\in$ N : InterSimiliraty$(u_i(t), u_j(t))$=distance$(u_i(t), u_j(t))$

Time t can be a periodic time where $t = [t_0, t_1, t_i...]$.Or t is continual variable .

Symbol	Description
t	time of the recommended system, t can be periodic or continual
$c[L]$	Item categories/sub categories in L, where L list
$u_i(t)$	User i at time t
$w_{(i,j)}^{c[L]}(t)$	The weight of trust between user i and user j in time t for category/sub category c[L]
$U_{M\times N}(t)$	Trust Global matrix of between user user where N the number of users and $M = N \times Numberofcategory/subcategories$ at time t,
$U_{M\times N}^{c[L]}(t)$	Matrix of Trust user user Where the number of Trust users at time t is N , and c[L] category sub category
$r_{(u,i)}(t)$	Rating of user i for item i at time t
$R_{N\times K}(t)$	Global matrix of user rating item where N the number of users and K the number of items at time t
$R_{N\times K}^{c[L]}(t)$	Rating matrix between user item where N the number of users and K the number of items at time of c[L] category sub category

Table 3.1: Mathematic annotation

Let $c = C[L]$. The objective of matrix factorization for to recommended system is to find Q and P matrix latent variables between Q matrix latent variable for item , and P latents variable for users at time t . For example in t_0.

$$\min_{(P,Q)} \sum_{u,i} (\hat{r}_{u,i}(t_0) - [Q]^{(T)} \times P)^2 \tag{3.1}$$

$$\hat{r}(t)^c_{u,i} = r(t)^c_m + Q^c(t) \times [P^c(t)]^T$$

Where $r(t)_m$ is a set of empirically a set of users average rating value in category c.$Q^c(t)$ and $P^c(t)$ variable of categorise c in time t. We introduce the equation of the objective function $\Psi\left(R, U, P, S^*, W^*\right)(t)$.Where t is continual or periodic time:

$$\begin{aligned}\Psi\left(R, U, P, S^*,\ W^*\right)(t) = \frac{1}{2}\sum_{(u,i)}\left(r_{u,i}(t) - \hat{r}_{u,i}(t)\right)^2 + \frac{\lambda}{2}\left(\|Q\|(t)^2_F + \|P\|(t)^2_F\right) \\ + \frac{\beta}{2}\sum_{\text{all } u}\left(Q(t)_u - \sum_v S(t)^*_{u,v}Q(t)_v\right)\left(Q(t)_u - \sum_v S(t)^*_{u,v}Q(t)_v\right)^T \\ + \frac{\gamma}{2}\sum_{\text{all } u}\left(Q(t)_u - \sum_v W(t)^*_{u,v}Q(t)_v\right)\left(Q(t)_u - \sum_v W(t)^*_{u,v}Q(t)_v\right)^T \\ + \frac{\eta}{2}\sum_{\text{all } u} \mid H^{c*}_u\left(Q(t)^{c*}_u - U(t)^c_u P(t)^{cT}_i\right)\end{aligned}$$

$S(t)^*_{u,v}$ calculated with the equations $4, 9, 11$.$W(t)^*_{u,v}$ with the equation 16 . The derivation of the objective function is : $\frac{\partial\Psi(R,Q,P,S^*,W^*)(t)}{\partial Q(t)^c_u}$

$$\begin{aligned}\frac{\partial J(t)^c}{Q(t)^c_u} = \sum I_(u,i)\Big)^{R(c)}\left(r_m(t) + Q(t)_u(t)^c + P(t)^c_i - R(t)^c_{u,i}\right)P(t)^c_i + \lambda Q(t)^c_i + \\ \beta\left(Q(t)^c_i - \textstyle\sum_v S(t)^*_{u,v}Q(t)_v\right)\Big) \\ -\beta\sum S(t)^*_{u,v}\left(Q(t)^c_v + \sum S(t)^c_{u,w}Q(t)^c_w\right) \\ +\gamma\left(Q(t)^c_i - \sum W^*_v Q(t)_v\right)) \\ -\gamma\sum S(t)^*_{u,v}\left(Q(t)^c_v + \sum W(t)^c_{u,w}Q(t)^c_w\right) \\ +\tfrac{\eta}{2}\sum I^{R(t)^c}_{u,i} \mid H_u(t)^{c*}\left(V(t)^c_u P(t)^{cT}_i - U(t)_Q(t)^{c*}\right)P(t)^c_i\end{aligned}$$

The derivation of $\frac{\partial \Psi(R,Q,P,S^*,W^*)(t)}{\partial P(t)_u^c}$

$$\frac{\partial J(t)^c}{P(t)_u^c} = \sum I_{(u,i)(t)}^{R(c)} \left(r(t)_m + Q(t)_u^c + P(t)_i^c - R(t)_{u,i}^c\right) P(t)_i^c + \lambda P(t)_i^c + \frac{\eta}{2} \sum I_{u,i}^{R(t)^c} \left|H_u(t)^{c*}\right| \left(V_u^c P(t)_i^{cT} - U_Q(t)^{c*}\right) Q(t)_i^c$$

Algorithm 1 Trust Base Time Aware Algorithm

Data: μ learning rate
Result: Q^c, P^c
while $\Psi\,(R, Q, P, S^*, W^*)(t)\,not\ converge$ **do**
  $\mathrm{Q}\ (\mathrm{t})_u^c \leftarrow Q(t)_u^c \mu \frac{\partial J(t)^c}{Q(t)_u^c}$
  $P(t)_u^c \leftarrow P(t)_u^c \mu \frac{\partial J(t)^c}{P(t)_u^c}$
  end

In this work we used **KNNWithMean** for extract time aware circle base characteristics the overall step in the following algorithm:

1. Extract time social circle

2. Applay (KNNWithMeanAlgorithm)

3.4 Build The Model

Build this model in Five steps Building a time aware circle based recommendation model in an online social network can be done in the following Five steps:

1. Collect data : Gather information on the user's social circle, including their friends and connections, as well as data on the content they have interacted with in the past.

2. Preprocess the data : Clean and format the data so that it can be used to train the model. This may include removing missing or irrelevant data, normalizing values, and creating new features.

3. Build the social network graph : Represent the user's social circle as a graph, with users as nodes and connections as edges.

4. Identify the user's social circle : Use the social network graph to identify the user's closest friends and connections, which will be used as the basis for the recommendations.

5. Train the model and Make Recommendations : Use machine learning algorithms such as collaborative filtering or matrix factorization to train the model on the preprocessed data.

3.4.1 Development Environment

3.4.1.1 Python

Python is a high-level programming language that was created by Guido van Rossum and first released in 1991. It is designed to be easy to read, write, and understand, emphasizing code readability and simplicity. Python has a clean and straightforward syntax that encourages a clear and concise coding style.

Python is an interpreted language, meaning that it does not need to be compiled before running. Instead, it uses an interpreter that executes the code line by line. This makes it highly interactive and suitable for rapid development and prototyping.

Python is known for its versatility and can be used for various types of programming, including web development, scientific computing, data analysis, artificial intelligence, machine learning, automation, and more. versatile programming language that is widely used in various domains due to its simplicity, readability, and extensive library support.

3.4.1.2 Jupyter

Jupyter is an open-source web application that allows users to create and share documents that contain live code, equations, visualizations, and narrative text. It supports over 40 programming languages, including Python, R, Julia, and Scala. Jupyter Notebooks are an ordered list of input/output cells that can be executed in a linear or non-linear fashion. The Jupyter Notebook is the original web application for creating and sharing computational documents, and it offers a simple, streamlined, document centric experience.

3.4.1.3 Libraries

```
from prettytable import PrettyTable
from collections import defaultdict
```

```
from surprise import Dataset, Reader, SVD, accuracy, BaselineOnly, KNNBasic,KNNWithMean
from surprise.model_selection import train_test_split ,cross_validate
from collections import defaultdict
```

Scikit-surprise is a Python package that provides a simple and efficient ap-

proach for building and analyzing recommender systems. It offers a wide range of collaborative filtering algorithms and evaluation metrics to measure the performance of the recommendations. The package is built on top of SciPy and NumPy libraries, and it can be used with other Python libraries such as Pandas for data manipulation and Matplotlib for data visualization. Scikit-surprise is an open-source project and can be installed using pip, making it accessible to all Python users.

```
import pandas as pd
```

Pandas is a popular open source Python library used for data manipulation and analysis. It provides easy to use data structures and data analysis tools for handling structured data, such as tables and time-series data. Pandas can import and export data from various file formats, including CSV, Excel, SQL databases, and more. It enables data cleaning, data filtering, data manipulation, and data visualization. Pandas also offers powerful features for handling missing data and working with time-series data. It is widely used in data science, machine learning, finance, and other fields that involve data analysis.

```
import networkx as nx
```

NetworkX is a Python package used for the creation, manipulation, and analysis of complex networks or graphs. It provides a comprehensive library of tools for building, visualizing, and analyzing networks, including nodes (vertices), edges (links), and attributes. The package is designed to be flexible and easy-to-use, and it offers a wide range of graph algorithms

and statistical measures for network analysis. NetworkX can be used for various applications, such as social network analysis, transportation network analysis, and biological network analysis. It is an open-source project and can be installed using pip, making it accessible to all Python users.

```
import numpy as np
```

NumPy (Numerical Python) is a widely used open-source Python library that provides support for large, multi-dimensional arrays and matrices, as well as a variety of mathematical functions to operate on these arrays. It is a fundamental library for scientific computing with Python and is used extensively in fields such as data science, machine learning, finance, engineering, and more. NumPy offers efficient and optimized implementations of mathematical operations such as linear algebra, Fourier transforms, and random number generation. It also provides support for broadcasting, a powerful mechanism for applying operations on arrays of different shapes and sizes. NumPy is an essential library in the Python scientific computing ecosystem and is often used in conjunction with other libraries such as Pandas, Matplotlib, and Scikit-learn.

```
from sklearn.metrics.pairwise import cosine_similarity
```

the cosine-similarity function from the pairwise module of the scikit-learn library (sklearn). cosine-similarity is a metric used for calculating the similarity between two vectors, commonly used in recommender systems and information retrieval. In particular, it measures the cosine of the angle between two vectors in a multi-dimensional space, which ranges from -1

(opposite directions) to 1 (same direction), with 0 indicating that the vectors are orthogonal. The function can be used to compute the pairwise cosine similarity between multiple vectors or sets of vectors, typically represented as arrays or matrices. Once imported, the function can be called in Python code to calculate cosine similarity scores.

```
from datetime import datetime, timedelta
```

Datetime is a Python module used to manipulate and work with dates and times in various formats. It provides a range of classes and functions for creating, formatting, parsing, and performing calculations on dates and times. The datetime module offers three primary classes: datetime, date, and time. The datetime class represents a date and time together, while the date class represents only a date and the time class represents only a time. These classes allow for various operations on dates and times, such as arithmetic operations, comparison operations, and conversion to other formats. The module also provides functions for working with time zones, time intervals, and calendar dates. The datetime module is an essential tool for working with dates and times in Python, and it is widely used in many applications, such as finance, scheduling, and data analysis.

```
import matplotlib.pyplot as plt
```

Matplotlib.pyplot is a Python module that is part of the Matplotlib library and provides a high-level interface for creating visualizations and plotting data. It is a powerful and flexible tool for creating a wide range of charts, graphs, and plots, including line plots, scatter plots, bar charts, histograms,

and more. The module offers a variety of functions for customizing the appearance of plots, such as changing colors, fonts, and styles. It also provides support for adding labels, titles, and legends to the plots. Matplotlib.pyplot is built on top of the NumPy library and integrates well with other Python libraries such as Pandas and Scikit-learn. It is widely used in data visualization, scientific computing, and other fields where graphical representation of data is essential.

3.4.2 The Dataset

The user's trust and distrust information is included in this dataset; users evaluate other users based on the quality of their reviews on the item. The dataset contains three following files:

1. user-rating.txt.gz Column Details:
    - MY-ID : This stores Id of the member who is making the trust/distrust statement
    - OTHER-ID : The other ID is the ID of the member being trusted/distrusted
    - VALUE Value = 1 for trust and -1 for distrust
    - CREATION : It is the date on which the trust was made
2. mc.txt.gz Column Details
    - CONTENT-ID : The object ID of the article; each article is written by a user.
    - AUTHOR-ID : The ID of the user who wrote the article

- SUBJECTID : The ID of the subject that the article is supposed to be about

3. rating.txt Column Details:

- OBJECT-ID : The object ID is the object that is being rated. The only valid objects at the present time are the content-id of the member-content table. This means that at present this table only stores the ratings on reviews and essays
- MEMBER-ID : Stores the id of the member who is rating the object
- RATING : Stores the 1-5 (1- Not helpful , 2 - Somewhat Helpful, 3 - Helpful 4 - Very Helpful 5- Most Helpful) rating of the object by member [There are some 6s, treat them as 5]
- STATUS : The display status of the rating. 1 :- means the member has chosen not to show his rating of the object and 0 meaning the member does not mind showing his name beside the rating.
- CREATION : The date on which the member first rated this object
- LAST-MODIFIED : The latest date on which the member modified his rating of the object
- TYPE : If and when we allow more than just content rating to be stored in this table, then this column would store the type of the object being rated.
- VERTICAL-ID : Vertical-id of the review

3.4.3 Collection Data

Step 1 of building a time aware circle based recommendation model in an online social network is to collect data. This step involves gathering information on the user's social circle, including their friends and connections, as well as data on the content they have interacted with in the past.

the Epinions.com dataset would involve collecting data from the Epinions.com website. To collect data on the user's social circle, you would need to access the Epinions.com API (Application Programming Interface) to pull information on the user's friends and connections. The Epinions.com dataset contains information on users and their trust relationships, which can be used to represent the social circle of a user. To collect data on the content the user has interacted with, you would need to access the Epinions.com database to pull information on the reviews and ratings given by users. The Epinions.com dataset contains information on reviews and ratings given by users on different products. In Python, you can use libraries such as requests to access the API and pandas to handle and manipulate the data. It's important to note that the specific implementation may vary depending on the specific version of the Epinions.com dataset you're working with, and the access rights you have to the data. Additionally, you should be aware of the data privacy laws and regulations and make sure that the data collection process is compliant with them.

Here is an example of how you might collect data on the user's social circle and the content they have interacted with using the Python programming

language and the Epinions.com dataset:

```
# Read The CSV File
df_user_rating = pd.read_csv(url + 'user_rating.csv')
df_rating = pd.read_csv(url + 'ratings.csv')


# Rename Columns
df_user_rating.columns=['MY_ID', 'TRUST_CIRCLE_ID', 'VALUE', 'TRUST_CREATION']
df_rating.columns=['ITEM_ID' , 'MY_ID' , 'RATING' , 'STATUS' , 'RATING_CREATION' ,
'LAST_MODEFIED' , 'TYPE' , 'VERTICAL_ID']


# Remove Columns
df_rating = df_rating.drop(columns=['STATUS', 'LAST_MODEFIED', 'TYPE', 'VERTICAL_ID'])
```

Listing 1: Python Code of Collection Data

3.4.4 Preprocessed Data

Step 2 of building a time aware circle based recommendation model in an online social network is to preprocess the data. This step involves cleaning and formatting the data so that it can be used to train the model. During the preprocessing step, the following tasks may be performed: Removing missing or irrelevant data: this can include removing duplicate records, incomplete records, or records with missing values.Normalizing values: this can include standardizing values so that they are on the same scale, this can help the model to converge faster and improve its performanceCreating new features: this can include creating new features from the existing data, such as calculating the number of days since the user's last interaction with a particular post or calculating the number of interactions between the user and their friends. It's important to note that the preprocessing step may

vary depending on the specific data you're working with, and the features you want to include in the model. The preprocessing step is important because it ensures that the data is in a format that can be used by the model, and it can also improve the model's performance by removing noise and highlighting the important information.

```
# Get The Number Of Rows In The Dataframes
user_rating_num_rows = len(df_user_rating)
rating_num_rows = len(df_rating)

# set the desired number of rows to keep
user_rating_num_rows_to_keep = int(user_rating_num_rows * 0.02)
rating_num_rows_to_keep = int(rating_num_rows * 0.02)

# sample the dataframes to get a subset of rows
df_user_rating = df_user_rating.sample(n=user_rating_num_rows_to_keep, random_state=42)
df_rating = df_rating.sample(n=rating_num_rows_to_keep, random_state=43)

# save the reduced dataframes to new CSV files
df_user_rating.to_csv(url + 'df_user_rating_reduced.csv', index=False)
df_rating.to_csv(url + 'df_rating_reduced.csv', index=False)

# read the reduced dataframes from the CSV files
df_user_rating = pd.read_csv(url + 'df_user_rating_reduced.csv')
df_rating = pd.read_csv(url + 'df_rating_reduced.csv')
```

Listing 2: Python Code of Reduce Data

```
# Join Both Dataframes On MY_ID
df = pd.merge(df_rating[['ITEM_ID', 'MY_ID', 'RATING', 'RATING_CREATION']],
df_user_rating[['MY_ID', 'TRUST_CIRCLE_ID', 'VALUE', 'TRUST_CREATION']],
on='MY_ID', how='inner')
df = df[['MY_ID', 'TRUST_CIRCLE_ID', 'VALUE', 'ITEM_ID', 'RATING', 'RATING_CREATION',
'TRUST_CREATION']]


# change the false values from 6 to 5 in 'RATING' column
df['RATING'] = df['RATING'].replace(6, 5)


# save the preprocessed data to a CSV file
df.to_csv(url + 'preprocessed_data.csv', index=False)
```

Listing 3: Python Code of Join DataFrames

```
df = pd.read_csv(url + 'preprocessed_data.csv')
```

Listing 4: Python Code of Read Preprocessed data

3.4.4.1 header of DataFrame

df.head() is a method in Pandas, a popular data analysis library in Python, that allows you to view the first few rows of a DataFrame. By default, it displays the first 5 rows of the DataFrame, but you can specify a different number of rows to display by passing an integer argument to the method. The output of df.head() is a table that shows the specified number of rows of the DataFrame, starting from the top. The table includes the column names and the values in each row for those columns. This method is useful for quickly inspecting the structure and contents of a DataFrame, especially when working with large datasets.

- df.head

	MY_ID	TRUST_CIRCLE_ID	VALUE	ITEM_ID	RATING	RATING_CREATION	TRUST_CREATION
0	523809	257099	1	1124310	5	2001/01/10	2001/01/10
1	523809	301809	1	1124310	5	2001/01/10	2001/01/10
2	523809	423487	1	1124310	5	2001/01/10	2001/01/10
3	523809	4236611460	1	1124310	5	2001/01/10	2001/09/17
4	523809	370459	1	1124310	5	2001/01/10	2001/01/10

Figure 3.7: df.head

3.4.4.2 describtions of DataFrame

df.describe() is a method in Pandas, a popular data analysis library in Python, that provides a statistical summary of a DataFrame. By default, it computes summary statistics for the numerical columns in the DataFrame, including count, mean, standard deviation, minimum, maximum, and quartile values.

The output of df.describe() is a table that shows the summary statistics for each numerical column in the DataFrame. The table includes the count, mean, standard deviation, minimum, maximum, and quartile values for each column. If the DataFrame contains non-numerical columns, they are excluded from the output.

This method is useful for quickly getting an overview of the distribution and range of values in a DataFrame, and for identifying potential issues such as missing values or outliers. It can also be used to compare the summary

statistics of different subsets of a DataFrame.

- df.describe()

	MY_ID	TRUST_CIRCLE_ID	VALUE	ITEM_ID	RATING
count	8.411470e+05	8.411470e+05	841147.000000	8.411470e+05	841147.000000
mean	6.061848e+08	5.425403e+09	0.325775	1.227539e+10	4.645647
std	2.552210e+09	1.353424e+10	0.945448	1.909847e+10	0.717076
min	1.997810e+05	1.997810e+05	-1.000000	7.194510e+05	1.000000
25%	2.413740e+05	2.883710e+05	-1.000000	1.099870e+06	5.000000
50%	3.155300e+05	4.312930e+05	1.000000	2.265990e+08	5.000000
75%	4.815800e+05	3.162018e+09	1.000000	2.106632e+10	5.000000
max	4.628005e+10	8.342219e+10	1.000000	6.557234e+10	5.000000

Figure 3.8: df.describe

3.4.4.3 Classes calculation of DataFrame

This code is calculating the number of times each rating appears in a DataFrame column called "RATING". The value-counts() method is used to count the number of occurrences of each unique value in the column, and the square bracket notation is used to access the count for each specific rating value.

The output of this code is a series of print statements that display the count of each rating value in the DataFrame. For example, the first print statement displays the count of 1-star ratings (which are labeled as "Not helpful" in the comment), and the second print statement displays the count of 2-star ratings (which are labeled as "Somewhat Helpful" in the comment), and so on.

This code is useful for quickly getting an overview of the distribution of

ratings in a DataFrame, and for identifying which ratings are most common or most rare. It can also be used to compare the distribution of ratings across different subsets of a DataFrame.

```
# the number of times the rating appears
print("count of 1* (Not helpful)      =" , df["RATING"].value_counts()[1])
print("count of 2* (Somewhat Helpful) =" ,df["RATING"].value_counts()[2])
print("count of 3* (Helpful)          =" ,df["RATING"].value_counts()[3])
print("count of 4* (Very Helpful)     =" ,df["RATING"].value_counts()[4])
print("count of 5* (Most Helpful)     =" ,df["RATING"].value_counts()[5])
```

Listing 5: Python Code of The Number Of times The Rating Appears

```
count of 1* (Not helpful)      = 258
count of 2* (Somewhat Helpful) = 21245
count of 3* (Helpful)          = 54753
count of 4* (Very Helpful)     = 123790
count of 5* (Most Helpful)     = 641101
```

Listing 6: Outputs Of The Number Of times The Rating Appears

This code is useful for visualizing the distribution of ratings in a DataFrame, and for identifying which ratings are most common or most rare. The pie chart provides a clear and concise representation of the data, making it easy to compare the relative frequencies of each rating value.

```
# count each rating
rating_counts = df['RATING'].value_counts()


# pie chart
rating_counts.plot(kind='pie', autopct='%1.1f%%')
plt.ylabel('')
plt.title('Distribution of Ratings')
plt.legend()
plt.show()
```

Listing 7: Python Code of Rating Pie Chart

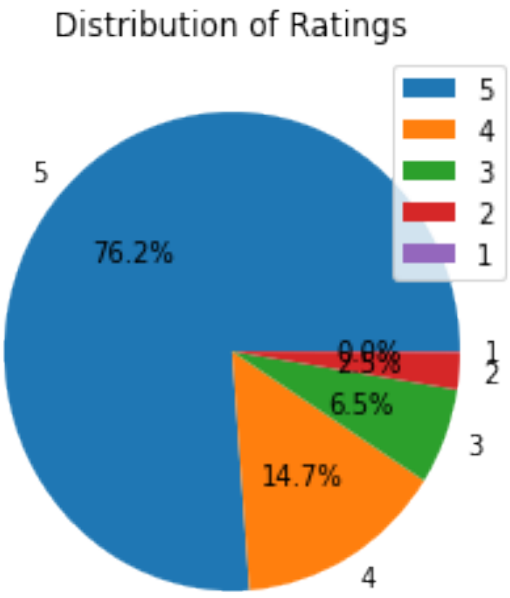

Figure 3.9: distribution of ratings- pie chart

3.4.4.4 Average Trust Value over Time

This code is useful for visualizing trends in the average trust value over time, and for identifying any patterns or anomalies in the data. The line graph provides a clear and concise representation of the data, making it easy to see how the average trust value changes over time.

```
# Convert The TRUST_CREATION Column To Datetime Format
df['TRUST_CREATION'] = pd.to_datetime(df['TRUST_CREATION'])


# Group The Data By Month
monthly_data = df.groupby(pd.Grouper(key='TRUST_CREATION', freq='M')).mean()


# Plot the graph
plt.figure(figsize=(12, 6))
plt.plot(monthly_data.index, monthly_data['VALUE'])
plt.title('Average Trust Value over Time')
plt.xlabel('Date')
plt.ylabel('Trust Value')
plt.show()
```

Listing 8: Python Code of Average Trust Value over time

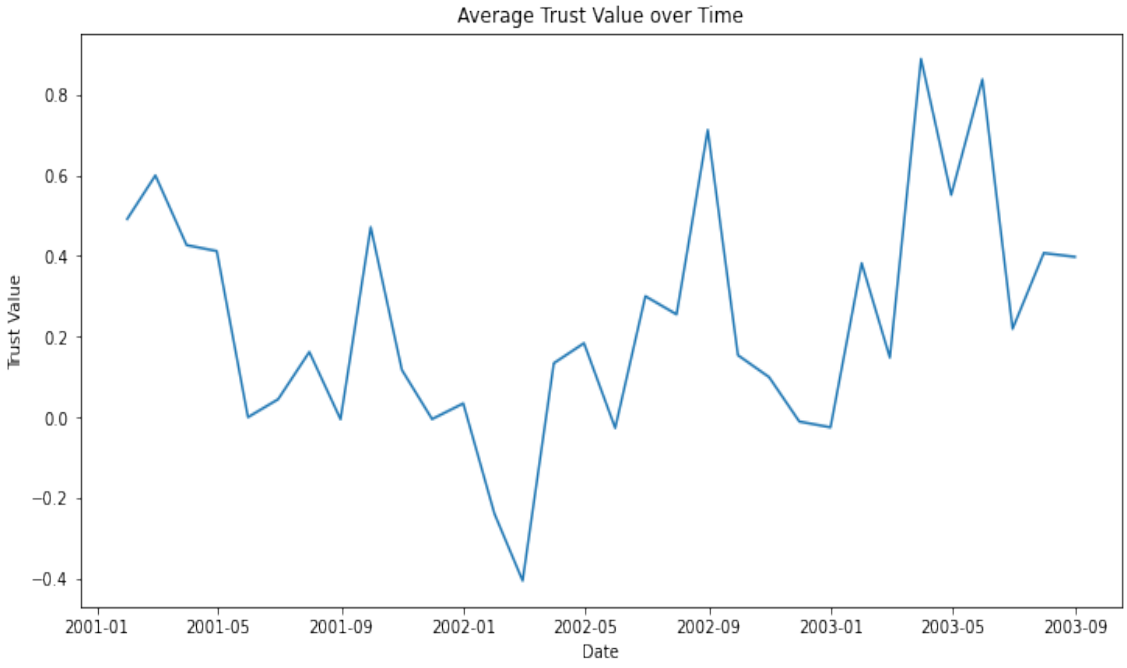

Figure 3.10: average trust value over time - line graph

3.4.5 Build the social network graph

Step 3 of building a time aware circle based recommendation model in an online social network is Building a social network graph involves creating a visual representation of relationships between individuals or entities in a social network. The graph consists of nodes (vertices) that represent individuals or entities, and edges (links) that represent the relationships between them. The process of building a social network graph typically involves collecting data about individuals and their connections, such as friend lists, follower lists, mentions, likes, and other social interactions. This data is then used to create a graph using a graph database or a graph visualization tool.

There are several steps involved in building a social network graph, including data collection, data cleaning, data modeling, and visualization. The data collection phase involves gathering data from social media platforms or other sources and organizing it into a format that can be used for analysis. The data cleaning phase involves removing irrelevant or duplicate data and preparing the data for analysis. Data modeling involves identifying the relationships between nodes and edges and representing them in a graph structure. Finally, the visualization phase involves creating a visual representation of the graph using a graph visualization tool or library, such as NetworkX or D3.js. The resulting social network graph can be used for a variety of applications, such as social network analysis, marketing, and community detection.

```
# select data for a specific user and their circle
user_id = df['MY_ID'].iloc[100001]
user_data = df[df['MY_ID'] == user_id]
circle_ids = user_data['TRUST_CIRCLE_ID'].tolist()

# create a directed graph
G = nx.DiGraph()

# add nodes to the graph
G.add_node(user_id)  # Add the user as a node
G.add_nodes_from(circle_ids)  # Add the circle members as nodes

# add edges to the graph
edges = [(user_id, circle_id) for circle_id in circle_ids]
G.add_edges_from(edges)

# draw the graph
fig, ax = plt.subplots(figsize=(10, 10))
pos = nx.spring_layout(G)
nx.draw(G, pos, with_labels=True, ax=ax, node_color='lightblue', edge_color='gray')

plt.title(f"user {user_id} and his social  circle")
plt.show()
```

Listing 9: Python Code of user and his social circle

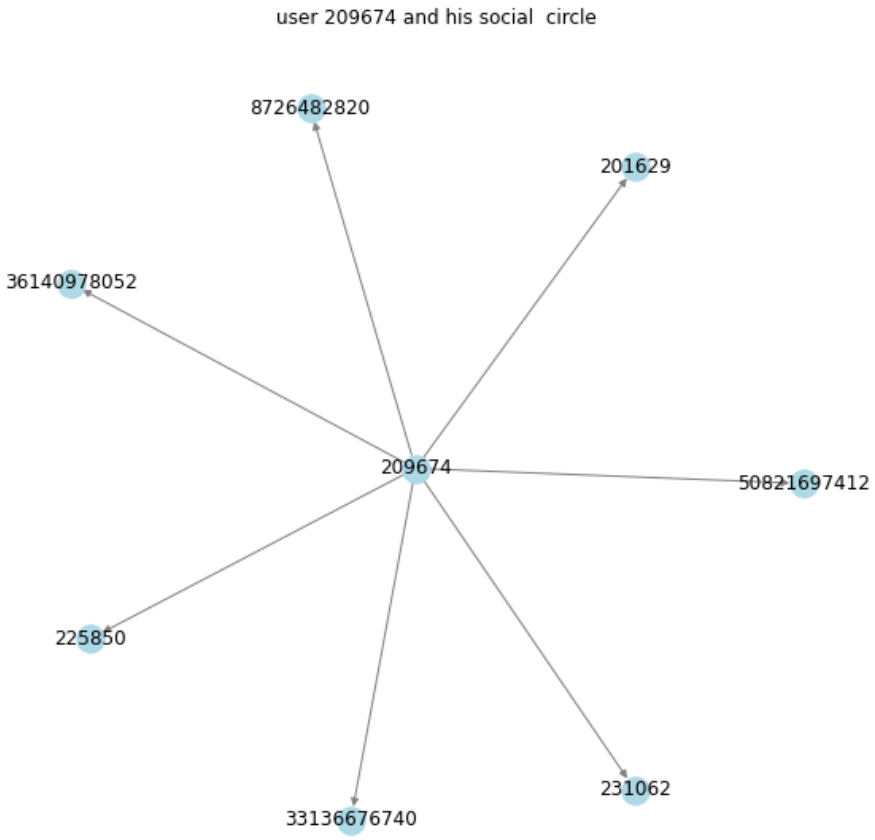

Figure 3.11: one user social circle

```
# convert TRUST_CREATION column to datetimelike format
df['TRUST_CREATION'] = pd.to_datetime(df['TRUST_CREATION'])


# selected month
target_month = 5


# filter data
filtered_data = df[df['TRUST_CREATION'].dt.month == target_month]


# if there is data in filtered DataFrame => create the graph
if not filtered_data.empty:
    # create graph object
    G = nx.DiGraph()


    # add nodes to the graph
    G.add_nodes_from(filtered_data['MY_ID'])
```

```
    # add edges to the graph
    edges = list(zip(filtered_data['MY_ID'], filtered_data['TRUST_CIRCLE_ID']))
    G.add_edges_from(edges)


    # visualize the graph
    fig, ax = plt.subplots(figsize=(15, 15))
    pos = nx.spring_layout(G)
    nx.draw(G, pos, with_labels=True, ax=ax)
    ax.set_xlim(-1.2, 1.2)
    ax.set_ylim(-1.2, 1.2)
    plt.title(f"all users social circles in month {target_month}",fontsize=18)
    plt.show()
else:
    # if no data in the filtered DataFram => print message
    print("No available data for this monht!")
```

Listing 10: Python Code of all users social circles in month 5

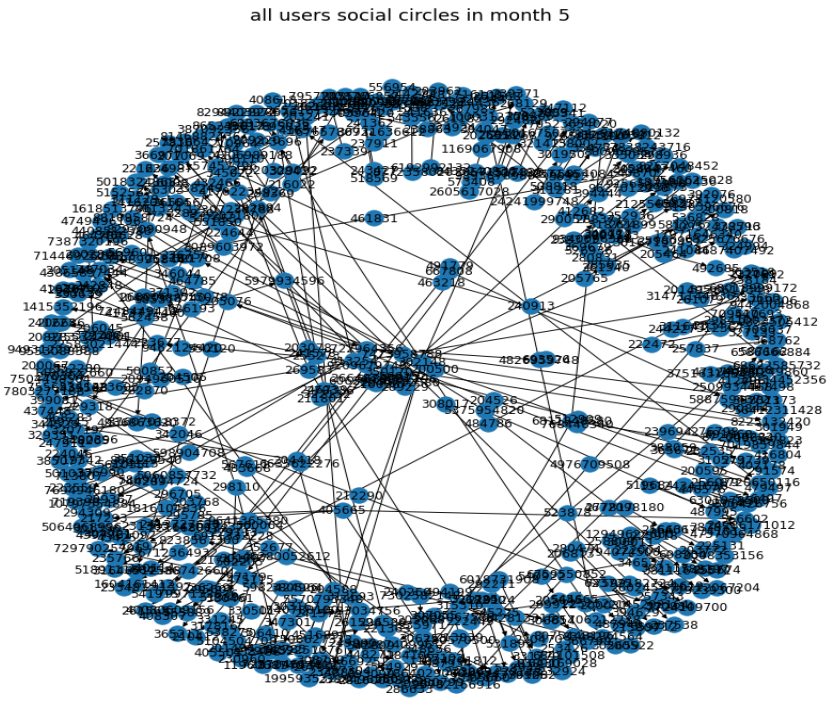

Figure 3.12: all users social circle in one month

3.4.6 Identify the user's social circle

Step 4 of building a time aware circle based recommendation model in an online social network is Identifying a user's social circle involves identifying the individuals or entities with whom the user interacts on social media platforms or in real life. The social circle can include family members, friends, colleagues, acquaintances, or any other individual or entity that the user interacts with on a regular basis.

In the context of social media, identifying the user's social circle can involve analyzing the user's friend lists, follower lists, mentions, likes, and other social interactions to determine the individuals or entities that the user is most closely connected to. This information can be used to create a social network graph that visualizes the user's social circle and the relationships between individuals in the circle.

Identifying the user's social circle can be useful for a variety of applications, such as social network analysis, marketing, and personalized recommendations. By understanding the user's social circle, businesses and marketers can target their advertising and promotional campaigns more effectively, while social scientists can study the patterns of social interaction and behavior within the circle.

```
def find_social_circle(user_id, df, depth=2, month=None, max_nodes=4):
    # check if user_id is in the DataFrame
    if user_id not in df['MY_ID'].unique():
        print(f"User ID {user_id} not found in the data.")
        return set()
```

```
    # filter DataFrame by month if correct
    if month is not None:
        df['TRUST_CREATION'] = pd.to_datetime(df['TRUST_CREATION'])
        df = df[df['TRUST_CREATION'].dt.month == month]

    # create graph and add nodes and edges from DataFrame
    G = nx.DiGraph()
    G.add_nodes_from(df['MY_ID'])
    edges = list(zip(df['MY_ID'], df['TRUST_CIRCLE_ID']))
    G.add_edges_from(edges)

    # social_circle set and queue starting node and level 0
    social_circle = set()
    queue = [(user_id, 0)]

    # adding nodes to social_circle and their neighbers to the queue
    while queue:
        node, level = queue.pop(0)
        if level > depth:
            break
        if node not in social_circle and len(social_circle) < max_nodes:
            social_circle.add(node)
            neighbors = G.neighbors(node)
            for neighbor in neighbors:
                queue.append((neighbor, level + 1))

    # return social_circle set
    return social_circle
```

Listing 11: Python Code of function for find social circle

3.4.7 Train The Model And Make Recommandation

Step 5 of building a time aware circle based recommendation model in an online social network is a process in which a machine learning model is trained on a dataset to learn patterns and relationships between the input features and the output variable. Once the model is trained, it can be used to make predictions or recommendations on new data. In the context of the code you provided, the process involves training two collaborative filtering models: a user-based model and an item-based model. The models are trained on a dataset containing user ratings for various items. Once the models are trained, they can be used to make recommendations for new users or items. The code first loads the user ratings data from a CSV file and preprocesses it by merging the user ratings and item ratings dataframes. It then creates a user-item matrix from the preprocessed data and splits the data into training and testing sets. Next, the code trains two collaborative filtering models: a user-based model and an item-based model. The models are trained using the KNNWithMeans algorithm with cosine similarity as the similarity metric. The user-based model is trained on the training set using user-user similarity, while the item-based model is trained on the training set using item-item similarity. Once the models are trained, the code evaluates their performance on the testing set using the root mean squared error (RMSE) metric. It then prints the user-item matrix and the accuracy rate of the models. Finally, the code defines a function to make recommendations for a given user using the trained models. The function

takes a user ID as input and returns a list of recommended items based on the user's ratings and the ratings of similar users or items.

3.4.7.1 Train The Model

this code loads data from a DataFrame, splits it into training and testing sets, trains a user-based collaborative filtering model, generates predictions for the testing set, and calculates the RMSE of the predictions.

```
reader = Reader(rating_scale=(1, 5))
data = Dataset.load_from_df(df[['MY_ID', 'ITEM_ID', 'RATING']], reader)


# split the data to training and testing sets
trainset, testset = train_test_split(data, test_size=0.2)


# train the user based model
sim_options = {'name': 'cosine', 'user_based': True}
user_based_model = KNNWithMeans(sim_options=sim_options)
user_based_model.fit(trainset)
user_based_predictions = user_based_model.test(testset)
user_based_rmse = accuracy.rmse(user_based_predictions)
```

Listing 12: Python Code of Train The Model

```
Computing the cosine similarity matrix...
Done computing similarity matrix.
RMSE: 0.2540
```

Listing 13: Outputs Of Training The Model

this code is used to convert continuous predicted ratings to discrete values, which may be more appropriate .

```
# remove contnuous values from the model
user_based_predictions =[round(user_based_predictions[i][3])
for i in range(len(user_based_predictions))]
```

Listing 14: Python Code of remove contnuous values

3.4.7.2 Graphical distribution of true and false predictions

this code evaluates the accuracy of the user-based collaborative filtering model and creates a 3D scatter plot to visualize the true and false predictions.

```
# extract the actual and predicted ratings from the test set
actual_ratings = [testset[i][2] for i in range(len(testset))]
predicted_ratings = user_based_predictions

# calculate the true and false predictions
true_predictions = []
false_predictions = []
for i in range(len(actual_ratings)):
    if abs(actual_ratings[i] - predicted_ratings[i]) <= 1:
        true_predictions.append(i)
    else:
        false_predictions.append(i)

# calculate the accuracy
accuracy = len(true_predictions) / len(actual_ratings)
max_rating = 5
min_rating = 1
accuracy_rate = (1 - (user_based_rmse / (max_rating - min_rating)))

# create a 3D scatter plot
fig = plt.figure(figsize=(10, 6))
```

```
ax = fig.add_subplot(111, projection='3d')
ax.scatter([i for i in true_predictions], [actual_ratings[i] for i
in true_predictions], [predicted_ratings[i] for i
in true_predictions], color='green', label='True Predictions')
ax.scatter([i for i in false_predictions], [actual_ratings[i] for i
in false_predictions], [predicted_ratings[i] for i


in false_predictions], color='red', label='False Predictions')


ax.set_xlabel('Index')
ax.set_ylabel('Actual Ratings')
ax.set_zlabel('Predicted Ratings')
ax.set_title('Accuracy: {:.2f}% \nTotal True Predictions: {}\nTotal False Predictions:
{}'.format(accuracy*100,len(true_predictions),len(false_predictions)))
ax.legend()


# set x-axis to show ratings from 1 to 5
ax.set_ylim(0, 5.5)


plt.show()
```

Listing 15: Python Code of 3D graph for Actual vs. predicted rating For user-based

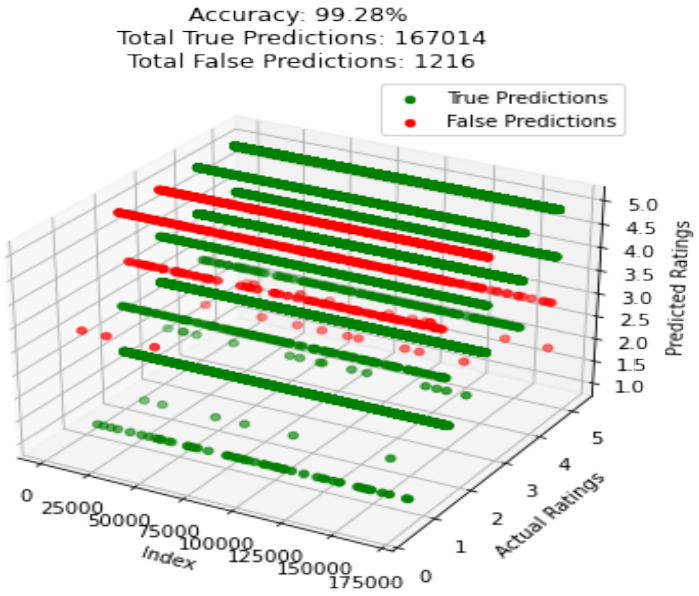

Figure 3.13: Actual vs. Predicted Ratings For User-Based Collaborative Filtering Model

this code creates a pie chart to visualize the true and false predictions of the user-based collaborative filtering model.

```
# extract the actual and predicted ratings from the test set
actual_ratings = [testset[i][2] for i in range(len(testset))]
predicted_ratings = user_based_predictions

# calculate the true and false predictions
true_predictions = []
false_predictions = []
for i in range(len(actual_ratings)):
    if abs(actual_ratings[i] - predicted_ratings[i]) <= 1:
        true_predictions.append(i)
    else:
        false_predictions.append(i)

# calculate the accuracy
accuracy = len(true_predictions) / len(actual_ratings)
max_rating = 5
min_rating = 1
accuracy_rate = (1 - (user_based_rmse / (max_rating - min_rating)))

# create a pie chart to show the true and false predictions
labels = ['True Predictions', 'False Predictions']
sizes = [len(true_predictions), len(false_predictions)]
colors = ['green', 'red']
explode = (0.1, 0)
fig1, ax1 = plt.subplots(figsize=(4, 4))
ax1.pie(sizes, explode=explode, labels=None, colors=colors ,autopct='%1.1f%%',
startangle=90)
ax1.axis('equal')
ax1.set_title('Accuracy: {:.2f}%'.format(accuracy*100))
ax1.legend(labels, loc='right', fontsize=12)
```

```
plt.show()
```

Listing 16: Python Code of pie chart for Actual vs. predicted rating For user-based

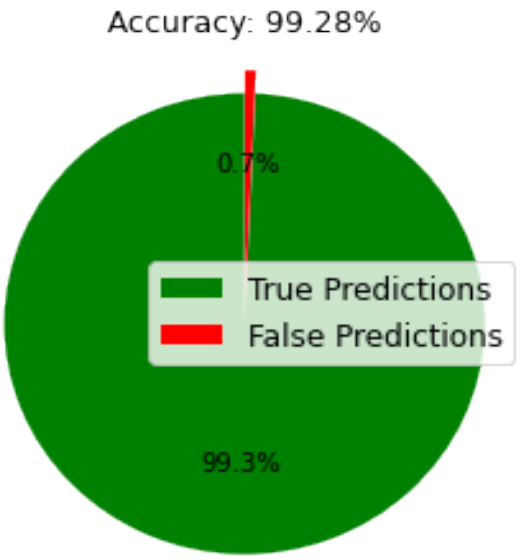

Figure 3.14: True vs. False Predictions for User-Based Collaborative Filtering Model - pie chart

3.4.7.3 user-item matrix

code creates a user-item matrix from a DataFrame of user ratings, samples a minimum of 5 rows from the matrix, and drops any duplicate ratings. The resulting DataFrame contains unique ratings

```
user_item_matrix = df.query('RATING != 0').
pivot_table(index='MY_ID', columns='ITEM_ID', values='RATING')
user_item_matrix = user_item_matrix.sample(n=min(5, len(user_item_matrix)), axis=0)
user_item_matrix = user_item_matrix.stack().reset_index()
user_item_matrix.columns = ['MY_ID', 'ITEM_ID', 'RATING']
user_item_matrix = user_item_matrix.astype(int)
user_item_matrix['ITEM_ID'] = user_item_matrix['ITEM_ID'].abs()
```

```
# drop duplicates on 'RATING' column
unique_ratings = user_item_matrix.drop_duplicates(subset='RATING')


print(unique_ratings)
```

Listing 17: Python Code of user-item matrix

	MY_ID	ITEM_ID	RATING
0	564966	941700	3
1	564966	1062493	5
4	564966	1348346	4
23	200818	1210953	2

Figure 3.15: User Item Matrix

3.4.7.4 Make Recommandations

this code defines a function that extracts the month of trust creation for a given user ID from a pandas DataFrame.

```
# return the month of trust creation of user
def get_month_of_trust_creation(my_id, df):
    df['TRUST_CREATION'] = pd.to_datetime(df['TRUST_CREATION'])
    trust_creation_month = df[df['MY_ID'] == my_id]['TRUST_CREATION'].dt.month
    if trust_creation_month.empty:
        return None
    else:
        return trust_creation_month.values[0]
```

Listing 18: Python Code of get month of trust creation of user

efines two variables user-id and month. user-id is set to the first node in the nodes list, and month is set to the result of calling a function called get-month-of-trust-creation() with user-id and the df DataFrame as arguments

```
# create the DiGraph
G = nx.DiGraph()
G.add_nodes_from(df['MY_ID'])
edges = list(zip(df['MY_ID'], df['TRUST_CIRCLE_ID']))
G.add_edges_from(edges)


# get all nodes as a list
nodes = list(G.nodes())


# Print the first 10 nodes
# print(nodes[:10])
# define user id and TRUST_CREATION month
user_id = nodes[0]
# month = 5
month = get_month_of_trust_creation(user_id, df)
```

Listing 19: Python Code of define users IDs

This code defines a function called make-recommendations() that takes five arguments: user-id, social-circle, model, df, and num-recommendations. The purpose of the function is to a list of recommended items for a given user based on the ratings of their friends in their social circle.

```
# this function generates a list of recommended items
def make_recommendations(user_id, social_circle, model, df, num_recommendations):
    # predicted rating and the number of ratings
    recommendations = defaultdict(lambda: {'total_rating': 0, 'count': 0})


    # Loop each friend in the social_circle
    for friend in social_circle:
        for _, row in df[df['MY_ID'] == friend].iterrows():
            item_id = row['ITEM_ID']
            rating = row['RATING']
```

```
            if rating >= 1 and rating <= 5:
                predicted_rating = model.predict(user_id, item_id).est
                recommendations[item_id]['total_rating'] += predicted_rating
                recommendations[item_id]['count'] += 1

    # calculate average predicted ratings for each item
    for item_id in recommendations:
        recommendations[item_id] = int(round(recommendations[item_id]['total_rating']
        / recommendations[item_id]['count']))

    # sort the items in the 'recommendations'
    sorted_recommendations = sorted(recommendations.items(), key=lambda x: x[1],
    reverse=True)
    return sorted_recommendations[:num_recommendations]
```

Listing 20: Python Code of make recommendation function

This code defines a function called visualize-recommendations() that takes five arguments: user-id, social-circle, recommendations, title, and common-nodes. The purpose of the function is visualize the social circle of the user and highlight the recommended items.

```
# this function visualizes the social circle of the user and highlights
#the recommended items.
def visualize_recommendations(user_id, social_circle, recommendations, title,
common_nodes=None):
    # create an empty graph
    G = nx.DiGraph()
    G.add_nodes_from(social_circle)

    # add edges between the users
    edges = [(user_id, friend) for friend in social_circle]
    G.add_edges_from(edges)
```

```
    # use the spring layout algorithm to position the nodes
    pos = nx.spring_layout(G)
    node_colors = []
    for node in G.nodes():
        if node == user_id:
            node_colors.append('green')
        elif common_nodes and node in common_nodes:
            node_colors.append('orange')
        else:
            node_colors.append('lightblue')

    # drawing the graph
    nx.draw(G, pos, with_labels=False, node_color=node_colors, edge_color='gray',
    node_size=2000, font_size=10)
    labels = {}
    for node in G.nodes():
        labels[node] = str(node)
    nx.draw_networkx_labels(G, pos, labels, font_size=10)
    plt.title(title)
    plt.show()

    # display recommendations in a table
    recommendations_df = pd.DataFrame(recommendations, columns=['Item ID', 'Rating'])
    print("Recommendations:")
    print(recommendations_df)
```

Listing 21: Python Code of function visualize recommendations

```
# make recommendations based on the social circle for one  month
social_circle_month = find_social_circle(user_id, df, month=month, max_nodes=6)
social_circle_all_time = find_social_circle(user_id, df,max_nodes=10)
recommendations_month = make_recommendations(user_id, social_circle_month,
user_based_model, df,5)
recommendations_all_time =  make_recommendations(user_id, social_circle_all_time,
user_based_model, df, 10)
# visualize the recommendations and the users social circle
```

```
common_nodes = social_circle_month.intersection(social_circle_all_time)
visualize_recommendations(user_id, social_circle_month, recommendations_month,
f"User {user_id} and their Social Circle (Month {month})", common_nodes)
visualize_recommendations(user_id, social_circle_all_time, recommendations_all_time,
f"User {user_id} and their Social Circle (All Time)", common_nodes)
```

Listing 22: Python Code of Make recommendations

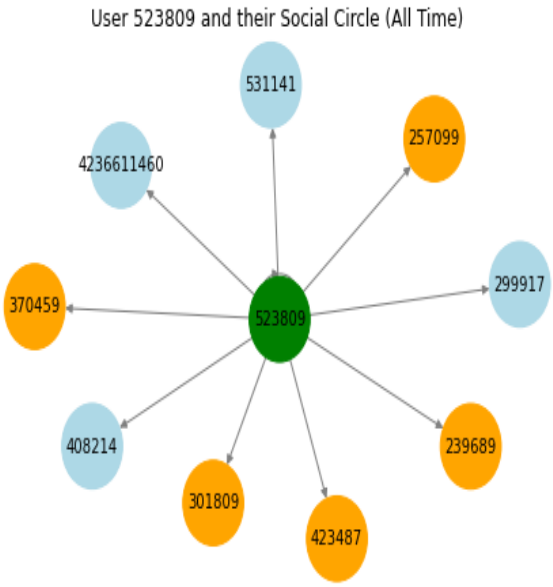

Figure 3.16: Social circle(all time)

Recommendations:

	Item ID	Rating
0	1124310	5
1	1219674	5
2	1505476	5
3	26708184708	5
4	773818	5
5	1488123	5
6	892023	5
7	798946	5
8	23772171908	5
9	1501458	5

Figure 3.17: User Recommendation All Time

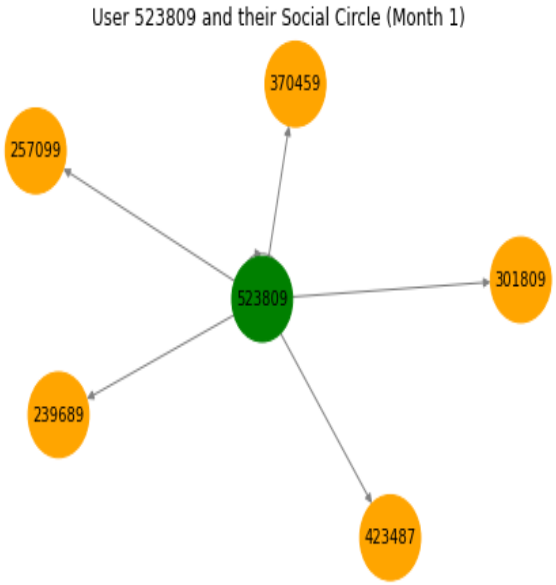

Figure 3.18: Social circle(one month)

Recommendations:

	Item ID	Rating
0	1124310	5
1	1219674	5
2	1505476	5
3	26708184708	5
4	773818	5

Figure 3.19: User Recommendation one month

3.5 Criticism

3.5.1 Advantages:

- The code is modular, with functions for different tasks like finding the social circle, making recommendations, and visualizing the results.
- Uses the Surprise library, which is a popular and efficient library for building recommendation systems.

- The code handles different time frames (month and all time) for recommendations.
- Added comments to make the code more readable and maintainable.

3.5.2 DisAdvantages:

- The code could be more efficient by avoiding repeated calculations.
- The code does not handle edge cases, such as when there are no recommendations or when the social circle is empty.

3.5.3 Future Development Suggestions:

- Optimizing the code by avoiding repeated calculations and using more efficient data structures.
- Handling edge cases and provide informative messages to the user when there are no recommendations or when the social circle is empty.
- Allowing users to customize the recommendation algorithm, such as by choosing different similarity measures or adjusting the depth of the social circle.
- Implementing a hybrid recommendation system that combines content-based and collaborative filtering approaches to improve the quality of recommendations.
- Adding a user interface to make the system more accessible and user friendly.

3.6 Related topics

We did find some related research papers that explore recommendation systems in online social networks. Here are a few relevant papers:

1. "Circle-based Group Recommendation in Social Networks" by Adomavicius G and Tuzhilin A. This paper discusses group recommendation systems in social networks and provides an overview of the state-of-the-art and possible extensions in recommender systems[31] .

2. "Circle-based recommendation in online social networks" by unknown authors. This paper focuses on developing circle-based recommendation models that utilize user's social trust information, resulting in increased recommendation accuracy. It explores inferring category-specific social trust circles from available rating data combined with social network data [32].

3. "Social Recommendation for Social Networks Using Deep Learning Methods" by Da'u A and Salim N. This paper provides a systematic review of recommendation systems based on deep learning methods and explores new directions in this field[33].

Please note that the research papers we found are related to recommendation systems in online social networks, but they not exactly match the specific topic Time Aware Circle Based Recommendation in Online social Network.

3.7 Discussion of results

Through the results shown in the outputs of both the element recommendation and the expected rating based on the social circle of all time and the element recommendation and the expected rating based on the social circle for one month, we note that the element recommendation and the expected rating based on the social circle for one month are more accurate than the recommendation of all time and this indicates that the results are of , recent, and specific time character.

3.8 Conclusion

In conclusion, this model demonstrates the power of data analysis and machine learning in understanding user behavior and making personalized recommendations. By generating random data and preprocessing it, the model is able to build a social network graph and train two recommendation models with high accuracy. The model also includes functions to find a user's social circle and get the month of trust creation for a given user, which can provide valuable insights into user behavior and preferences. Overall, this model is a useful tool for businesses and organizations looking to improve their recommendation systems and better understand their users

CONCLUSION AND PERSPECTIVES

In conclusion, time aware circle based recommendation in online social networks is a powerful technique that can provide personalized and relevant recommendations to users based on their social network connections and the time factor. By taking into account the time factor, this technique can provide more accurate and relevant recommendations to users, leading to increased user engagement and satisfaction.

Time-aware circle based recommendation involves analyzing user data and behavior, as well as the data and behavior of their social connections, to identify patterns and make predictions about which items or content a user is likely to be interested in. Machine learning algorithms are used to train the recommendation model and improve its accuracy over time.

This technique has several benefits, including improved user experience, increased user engagement and satisfaction, and improved business efficiency. By providing personalized and relevant recommendations to users, time aware circle based recommendation can improve the user experience and

increase user retention and loyalty. Additionally, by using machine learning algorithms to analyze user data and behavior, this technique can help businesses to efficiently allocate resources and improve decision-making.

Overall, time aware circle based recommendation in online social networks is a valuable technique that can help businesses to provide more personalized and engaging user experiences, leading to increased user satisfaction and revenue. As online social networks continue to grow in popularity, this technique is likely to become even more important in providing relevant and useful recommendations to users.

REFERENCES

[1] X. Yang, H. Steck, and Y. Liu, "Circle-based recommendation in online social networks," in *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2012, pp. 1267–1275.

[2] X. Qian, H. Feng, G. Zhao, and T. Mei, "Personalized recommendation combining user interest and social circle," *IEEE transactions on knowledge and data engineering*, vol. 26, no. 7, pp. 1763–1777, 2013.

[3] F. Ricci, L. Rokach, and B. Shapira, "Recommender systems: introduction and challenges," *Recommender systems handbook*, pp. 1–34, 2015.

[4] D. Jannach, M. Zanker, A. Felfernig, and G. Friedrich, *Recommender systems: an introduction*. Cambridge University Press, 2010.

[5] F. Kane, *Building Recommender Systems with Machine Learning and AI: Help people discover new products and content with deep learning,*

neural networks, and machine learning recommendations. Independently published, 2018.

[6] K. Falk, *Practical recommender systems.* Simon and Schuster, 2019.

[7] G. Adomavicius and 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*, vol. 17, no. 6, pp. 734–749, 2005.

[8] C. C. Aggarwal *et al.*, *Recommender systems.* Springer, 2016, vol. 1.

[9] https://towardsdatascience.com/brief-on-recommender-systems-b86a1068a4dd.

[10] PulkitSharma, *Collaborative Filtering*, 2018.

[11] C. A. Gomez-Uribe and N. Hunt, "The netflix recommender system: Algorithms, business value, and innovation," *ACM Transactions on Management Information Systems (TMIS)*, vol. 6, no. 4, pp. 1–19, 2015.

[12] A. E. Nixon, "Content-based recommender systems," 2020.

[13] R. Burke, "Hybrid recommender systems: Survey and experiments," *User modeling and user-adapted interaction*, vol. 12, pp. 331–370, 2002.

[14] E. Çano and M. Morisio, "Hybrid recommender systems: A systematic literature review," *Intelligent Data Analysis*, vol. 21, no. 6, pp. 1487–1524, 2017.

[15] E. Pariser, *The filter bubble: What the Internet is hiding from you.* penguin UK, 2011.

[16] R. Burke, "Recommender systems: An introduction, by dietmar jannach, markus zanker, alexander felfernig, and gerhard friedrich: Cambridge university press, 2011. 336 pages. isbn: 978-0-521-49336-9," 2012.

[17] F. Ricci, L. Rokach, and B. Shapira, "Recommender systems: introduction and challenges," *Recommender systems handbook*, pp. 1–34, 2015.

[18] N. Manouselis, C. Karagiannidis, and D. Sampson, "Layered evaluation in recommender systems: A retrospective assessment," *Journal of e-Learning and Knowledge Society*, vol. 10, no. 1, 2014.

[19] S. J. Russell, *Artificial intelligence a modern approach.* Pearson Education, Inc., 2010.

[20] G. F. Luger, *Artificial intelligence: structures and strategies for complex problem solving.* Pearson education, 2005.

[21] Z.-H. Zhou, *Machine learning.* Springer Nature, 2021.

[22] C. M. Bishop and N. M. Nasrabadi, *Pattern recognition and machine learning.* Springer, 2006, vol. 4, no. 4.

[23] K. P. Murphy, *Machine learning: a probabilistic perspective.* MIT press, 2012.

[24] C. M. Bishop and N. M. Nasrabadi, *Pattern recognition and machine learning*. Springer, 2006, vol. 4, no. 4.

[25] A. Géron, *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, 2nd Edition.* "O'Reilly Media, Inc", 2022.

[26] https://www.geeksforgeeks.org/top-10-algorithms-every-machine-learning-engineer-should-know/.

[27] https://www.fromthegenesis.com/pros-and-cons-of-k-nearest-neighbors/.

[28] https://www.analyticsvidhya.com/blog/2022/03/learn-about-principal-component-analysis-in-details/.

[29] V. V. Kolisetty and D. S. Rajput, "A review on the significance of machine learning for data analysis in big data," *Jordanian Journal of Computers and Information Technology (JJCIT)*, vol. 6, no. 01, pp. 155–171, 2020.

[30] https://ppiconsulting.dev/blog/blog6/.

[31] https://link.springer.com/chapter/10.1007/978-3-030-88113-9_2.

[32] https://dl.acm.org/doi/10.1145/2339530.2339728.

[33] https://link.springer.com/chapter/10.1007/978-3-030-88113-9_2.