

ELECTIVE RECOMMENDATION SYSTEM

A PROJECT REPORT

Submitted by,

NARIGANNAGARI SHAMEER-20211CSE0021

SREEKAR SANNITHI-20211CSE0020

RAHUL POLDAS-20211CSE0019

TADIPATRI LIKITH GANESH -20211CSE0120

Under the guidance of,

Mr. Jerrin Joe Francis

in partial fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

At



PRESIDENCY UNIVERSITY

BENGALURU

DECEMBER 2024

PRESIDENCY UNIVERSITY

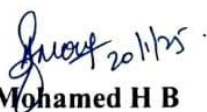
SCHOOL OF COMPUTER SCIENCE ENGINEERING

CERTIFICATE

This is to certify that the Project report "**ELECTIVE RECOMMENDATION SYSTEM**" being submitted by "NARIGANNAGARI SHAMEER, SREEKAR SANNITHI, RAHUL POLDAS, TADIPATRI LIKITH GANESH" bearing roll number "20211CSE0021, 20211CSE0020, 20211CSE0019, 20211CSE0120" in partial fulfillment of the requirement for the award of the degree of Bachelor of Technology in Computer Science and Engineering is a bonafide work carried out under my supervision.



Mr. Jerrin Joe Francis
ASSISTANT PROFESSOR
School of CSE
Presidency University



Dr. Asif Mohamed H B
ASSOCIATE PROFESSOR
& HoD
School of CSE
Presidency University



Dr. L. SHAKKEERA
Associate Dean
School of CSE
Presidency University



Dr. MYDHILI NAIR
Associate Dean
School of CSE
Presidency University



Dr. SAMEERUDDIN KHAN
Pro-Vc School of Engineering
Dean -School of CSE&IS
Presidency University

PRESIDENCY UNIVERSITY
SCHOOL OF COMPUTER SCIENCE ENGINEERING

DECLARATION

We hereby declare that the work, which is being presented in the project report entitled **ELECTIVE RECOMMENDATION SYSTEM** in partial fulfillment for the award of Degree of **Bachelor of Technology in Computer Science and Engineering**, is a record of our own investigations carried under the guidance of **Mr. JERRIN JOE FRANCIS, Assistant Professor, School of Computer Science Engineering & Information Science, Presidency University, Bengaluru.**

We have not submitted the matter presented in this report anywhere for the award of any other Degree.

| | |
|--------------------------------|-----------------------|
| NARIGANNAGARI SHAMEER | - 20211CSE0021 |
| SREEKAR SANNITHI | -20211CSE0020 |
| RAHUL POLDAS | -20211CSE0019 |
| TADIPATRI LIKITH GANESH | -20211CSE0120 |

ABSTRACT

In today's digital age, data shapes our online experiences. Recommendation engines play a crucial role in helping us decide on products, content, and services. This paper extensively explores the evolving domain of deep learning-based recommendation systems, focusing on significant progress in system evaluation, personalized recommendation methods, and the integration of diverse data sources.

The recommender system is studied widely and applied in different domains in this era. However, the domain of course recommendation, in the context of this research, addresses this predicament through the introduction of a collaborative filtering recommendation approach. This approach harnesses the information encapsulated within students prerequisite courses and the mean GPA of those prerequisites to furnish elective course recommendations, predicated on the resemblances observed among prior students. The methodology employs the Singular Value Decomposition (SVD) algorithm to quantify the similarity between students and the elective courses recommendation procedure.

The current accuracy of the recommendation system is 0.86. This shows that the system is now much better at predicting grade point.

ACKNOWLEDGEMENT

First of all, we indebted to the **GOD ALMIGHTY** for giving me an opportunity to excel in our efforts to complete this project on time.

We express our sincere thanks to our respected dean **Dr. Md. Sameeruddin Khan**, Pro-VC, School of Engineering and Dean, School of Computer Science Engineering & Information Science, Presidency University for getting us permission to undergo the project.

We express our heartfelt gratitude to our beloved Associate Deans **Dr. Shakkeera L and Dr. Mydhili Nair**, School of Computer Science Engineering & Information Science, Presidency University, and **Dr. Asif Mohamed H B**, Head of the Department, School of Computer Science Engineering & Information Science, Presidency University, for rendering timely help in completing this project successfully.

We are greatly indebted to our guide **Mr. Jerrin Joe Francis**, Associate Professor and Reviewer **Mr. Afroz Alam**, Assistant Professor, School of Computer Science Engineering & Information Science, Presidency University for his inspirational guidance, and valuable suggestions and for providing us a chance to express our technical capabilities in every respect for the completion of the project work.

We would like to convey our gratitude and heartfelt thanks to the PIP2001 Capstone Project Coordinators **Dr. Sampath A K**, **Dr. Abdul Khadar A** and **Mr. Md Zia Ur Rahman**, department Project Coordinators **Dr. Amarnath** and Git hub coordinator **Mr. Muthuraj**.

We thank our family and friends for the strong support and inspiration they have provided us in bringing out this project.

**NARIGANNAGARI SHAMEER,
SREEKAR SANNITHI,
RAHUL POLDAS,
TADIPATRI LIKITH GANESH**

LIST OF TABLES

| Sl. No. | Table Name | Table Caption | Page No. |
|----------------|-------------------|-----------------------------------|-----------------|
| 1 | Table 1.1 | TIMELINE FOR EXECUTION OF PROJECT | 31 |
| 2 | Table 1.2 | ACCURACY SCORES | 34 |

LIST OF FIGURES

| Sl. No. | Figure Name | Caption | Page No. |
|---------|-------------|-----------------------------------|----------|
| 1 | Fig 1.1 | ARCHITECTURE DIAGRAM | 27 |
| 2 | Fig 1.2 | TIMELINE FOR EXECUTION OF PROJECT | 30 |
| 3 | Fig 1.3 | RECOMMENDATION.HTML | 51 |
| 4 | Fig 1.4 | INDEX.HTML OUTPUT | 51 |

TABLE OF CONTENTS

| CHAPTER NO. | TITLE | PAGENO. |
|--------------------|--|----------------|
| | ABSTRACT | i |
| | ACKNOWLEDGMENT | ii |
| | INTRODUCTION | 1 |
| | 1.1 Addressing the Problem through Recommender Systems | 1 |
| 1. | 1.2 Evaluating the Effectiveness of Recommender Systems | 1 |
| | 1.3 Implications for Higher Education Institutions | 2 |
| | LITERATURE REVIEW | 4 |
| | 2.1 Online Course Recommendation System Using Double-Layer Attention Mechanism. | 4 |
| | 2.2 Personalized Recommendation System with Matrix Factorization | 4 |
| 2. | 2.3 Comprehensive Course Recommendation System Using K-NN and Matrix Factorization | 5 |
| | 2.4 Hybrid Recommendation Methods with Collaborative Filtering and Content-Based Filtering | 5 |
| | 2.5 Course Popularity and Recommendation Using | 6 |

| | |
|---|----|
| K-Means and FP- | |
| Growth Algorithm | |
| 2.6 Hybrid Recommendation System Using Genetic Algorithms | 6 |
| 2.7 Advances in Collaborative Filtering | 7 |
| 2.8 Properties and Structure of the Analytic Singular Value | 7 |
| 2.9 An Interval Perturbation Method for Singular Value Decomposition (SVD) with Unknown-But-Bounded (UBB) Parameters | 8 |
| 2.10 Intelligent Personalized Recommendation Method Based on Optimized Collaborative Filtering Algorithm in Primary and Secondary Education Resource System | 8 |
| 2.11 Simple Autoencoders for Collaborative Filtering | 9 |
| 2.12 Dual-Embedding Based Neural Collaborative Filtering for Recommender Systems | 10 |
| 2.13 Revisiting Matrix Decomposition to Generate Powerful Node Embeddings for Recommendation Systems | 11 |
| 2.14 E-Learning Course Recommender System Using Collaborative Filtering Models | 12 |
| 2.15 Online Course Recommender System for E- | 12 |

| | | |
|-----------|--|----|
| | Learning Platforms | |
| | 2.16 Privacy-Aware Collaborative Filtering Models | 13 |
| | for Recommender Systems | |
| | 2.17 Matrix Factorization with Rating Completion | 14 |
| | for Collaborative Filtering | |
| | 2.18 Fast Online Matrix Updates for Lightweight | 15 |
| | Recommender Systems | |
| | 2.19 Build a Recommendation Engine with | 16 |
| | Collaborative Filtering | |
| | 2.20 Collaborative Filtering via Matrix | 16 |
| | Approximation for Recommendation Systems | |
| | RESEARCH GAPS OF EXISTING | 18 |
| | METHODS | |
| | 3.1 Sparse Data in Recommendation Systems | 18 |
| | 3.2 Cold-Start Problem: | 18 |
| 3. | 3.3 Scalability Problems: | 19 |
| | 3.4 Privacy Issues in Recommendation Systems | 19 |
| | 3.5 Robustness of Recommendation Systems | 20 |
| | 3.6 Interpretability Issues | 20 |
| | PROPOSED MOTHODOLOGY | 21 |
| 4. | 4.1. The Matrix Setup | 21 |
| | 4.2 SVD Decomposition | 21 |

| | |
|--|----|
| 4.3 Step-by-Step Decomposition | 21 |
| 5. OBJECTIVES | 24 |
| 5.1 Addressing the Cold-Start Problem: | 24 |
| 5.2 Scalability and Flexibility | 24 |
| 5.3 Reducing Data Sparsity | 24 |
| 6. SYSTEM DESIGN & IMPLEMENTATION | 25 |
| 6.1 Data Collection | 25 |
| 6.2 Data Processing | 25 |
| 6.3 Recommendation Generation | 25 |
| 6.4 User Interaction | 25 |
| 6.5 Implementation Details | 25 |
| 6.5.1 Technologies and Tools | 25 |
| 6.5.2 Singular Value Decomposition (SVD) for Matrix Factorization | 26 |
| 7. TIMELINE FOR EXECUTION OF PROJECT | 30 |
| 8. OUTCOMES | 32 |
| 8.1 Increased Student Engagement and Retention | 32 |
| 8.2 Improved Academic Performance | 32 |
| 8.3 Increased Course Enrollment and Popularity | 32 |
| Insights | 32 |
| 8.4 Enhanced Faculty and Curriculum Planning | 32 |
| 8.5 Reduction in Course Mismatch and Dropout Rates | 32 |
| 8.6 Broader and Interdisciplinary Learning Opportunities | 33 |

| | | |
|------------|--------------------------------------|----|
| | 8.7 Data-Driven Educational Insights | 33 |
| 9. | RESULTS AND DISCUSSIONS | 34 |
| 10. | CONCLUSION | 36 |

CHAPTER-1

INTRODUCTION

Higher education institutions provide a wide array of elective courses, each tailored to various academic interests and career trajectories. While this diversity gives students numerous opportunities to explore new areas and strengthen their academic portfolios, it also creates a major challenge in decision-making. The abundance of choices can overwhelm students, making it difficult for them to choose courses that align with their long-term academic and professional aspirations, which may ultimately result in dissatisfaction with their educational journey.

In many cases, students end up taking courses based on convenience, social influence, or external pressures, rather than making decisions that reflect their true academic interests or career aspirations. This misalignment can hinder their ability to develop expertise in specific areas and limit their academic satisfaction. In the worst cases, students may take courses that not only fail to excite them but also do not contribute meaningfully to their chosen career paths.

1.1 Addressing the Problem through Recommender Systems

Recommender systems help students navigate the vast array of elective courses by providing personalized recommendations that are tailored to their individual preferences, strengths, and career objectives. By leveraging advanced algorithms that analyze large datasets of student behavior, past academic performance, and preferences, recommender systems can offer students targeted suggestions for courses that align more closely with their interests and goals.

The core advantage of recommender systems in the context of elective course selection is that they enable a personalized approach. Instead of relying on generic or one-size-fits-all advice, these systems create a tailored course path for each student, taking into account their unique academic background, performance history, and professional ambitions. By factoring in a student's interests and strengths, recommender systems guide students towards courses that are likely to enhance their skills, broaden their knowledge base, and align with their long-term goals.

Furthermore, these systems can adapt and improve over time. As a student progresses through their academic journey, the recommender system can update its recommendations based on new data, such as the student's course grades, engagement with different subjects, and evolving career aspirations. This dynamic adaptability ensures that the recommendations remain relevant and valuable, offering students timely guidance throughout their academic career.

1.2 Evaluating the Effectiveness of Recommender Systems

The effectiveness of recommender systems in assisting students with course selection is an important research topic in the field of higher education. The main objective of this study is to evaluate how accurately and effectively recommender systems can help students make informed decisions about their elective courses. By analyzing the system's ability to match students with courses that enhance their academic journey and career prospects, the study aims to provide valuable insights into the potential benefits of these systems.

To assess the usefulness of recommender systems, it is crucial to evaluate both accuracy and effectiveness. Accuracy refers to the system's ability to suggest courses that students are likely to find relevant and beneficial, while effectiveness measures the system's impact on the students' academic satisfaction, career preparedness, and overall success. These two factors can be assessed through various methods, including surveys, interviews, and academic performance tracking, to gauge how well the system meets students' needs.

In addition to providing personalized recommendations, recommender systems can also enhance the overall student experience by fostering a more engaging and informed academic environment. By guiding students to courses that align with their strengths and goals, these systems can increase student engagement, motivation, and retention. Moreover, they can help students navigate the often complex curriculum choices by offering suggestions.

1.3 Implications for Higher Education Institutions

For higher education institutions, implementing recommender systems represents an opportunity to improve the quality of academic advising and enhance student satisfaction. Institutions that adopt such systems can offer a more student-centric approach to course selection, ensuring that each student has access to the resources and guidance needed to make informed choices. Additionally, recommender systems can assist academic advisors by providing them with data-driven insights, helping them to offer more targeted and relevant advice to students.

The recommender systems provide students with a better understanding of how their course choices contribute to their academic and career trajectories, empowering them to make decisions that will benefit them in the long term.

Recommender systems hold significant promise in helping students select elective courses that align with their academic interests and career goals. By offering personalized course recommendations, these systems can guide students towards decisions that contribute to their academic success, career development, and overall satisfaction. As higher education institutions continue to embrace technology, the role of recommender systems in shaping the future of education will become increasingly important. This research aims to explore the potential benefits and challenges of

implementing these systems, providing valuable insights for both institutions and students seeking to optimize their academic journeys.

CHAPTER-2

LITERATURE SURVEY

2.1 Title: Online Course Recommendation System Using Double-Layer Attention Mechanism.

Author: Q. Zhu et al.

Algorithm Used: Double-layer attention mechanism with Convolutional Neural Networks (CNNs).

In their research, Q. Zhu et al. introduce a recommendation system that combines a double-layer attention mechanism with Convolutional Neural Networks (CNNs). This method seeks to improve recommendation performance by prioritizing key features and contextual information through the attention mechanism, integrated with the powerful capabilities of CNNs. This model shows promise for tackling intricate recommendation tasks, enhancing prediction accuracy with deep learning techniques.

However, there are notable limitations of the proposed algorithm. These include data bias, where recommendations may be influenced by incomplete or imbalanced data; the cold start problem, where the system struggles to suggest items for new users or those with minimal historical data; scalability challenges, as the model may have difficulty efficiently processing large datasets; and interpretability concerns, which can hinder user understanding of recommendation logic, potentially impacting trust and widespread adoption.

2.2 Title: Personalized Recommendation System with Matrix Factorization

Author: W. Wu et al.

Algorithm Used: Matrix Factorization

W. Wu et al. propose a personalized recommendation system that utilizes matrix factorization techniques to predict user preferences based on historical interaction data. Matrix factorization is effective in decomposing large user-item interaction matrices into latent factors, which represent hidden features that allow for more accurate predictions of user preferences. This approach is widely adopted in recommendation systems because it can handle large datasets and make tailored predictions based on the patterns observed in user behaviour.

Despite its benefits, this method also encounters several challenges. One significant drawback is the issue of sparse and subjective user ratings, where the availability of limited or imbalanced data can

negatively impact the model's ability to make precise recommendations. Furthermore, matrix factorization often struggles to capture the full complexity and nuances of user preferences, especially when users demonstrate diverse or evolving tastes that are not adequately reflected in the historical interaction data. These limitations can hinder the system's overall performance, particularly in dynamic environments or scenarios that demand high levels of personalization.

2.3 **Title:** Comprehensive Course Recommendation System Using K-NN and Matrix Factorization

Author: H. L. Thanh-Nhan et al.

Algorithm Used: K-Nearest Neighbor (K-NN) and Matrix Factorization

H. L. Thanh-Nhan et al. propose a comprehensive course recommendation system that integrates the K-Nearest Neighbour (K-NN) algorithm with matrix factorization techniques. This hybrid model aims to deliver more accurate course recommendations by leveraging the strengths of both algorithms. K-NN is utilized to identify similarities between users or courses, while matrix factorization extracts latent features from the user-item interaction data, assisting in predicting which courses a student may find appealing. The combination of these methods improves the system's ability to offer personalized recommendations tailored to individual user preferences.

However, the system also faces multiple challenges. A primary issue is its reliance on historical user data, which may hinder the effectiveness of the recommendations, particularly when interaction data is sparse or when dealing with new users. Moreover, the scalability of the system is limited, and as the dataset expands, performance may degrade. Finally, interpretability becomes a concern, as the merging of K-NN and matrix factorization could result in black-box models, making it difficult for users or administrators to understand the reasoning behind course recommendations, which could diminish the trust in the system.

2.4 **Title:** Hybrid Recommendation Methods with Collaborative Filtering and Content-Based Filtering

Author: A. Esteban et al.

Algorithm Used: Collaborative Filtering (CF), Content-Based Filtering (CBF), and Genetic Algorithms (GA)

A. Esteban et al. propose a hybrid recommendation system that merges Collaborative Filtering (CF), Content-Based Filtering (CBF), and Genetic Algorithms (GA). This integrated approach seeks to address the limitations of individual methods by leveraging the strengths of each. The behaviour of users to make recommendations, while Content-Based Filtering focuses on the characteristics of the items themselves. By incorporating Genetic Algorithms, the system gains the ability to evolve over

time, refining the recommendation process to better align with user preferences and enhancing both the accuracy and personalization of the overall system.

Nevertheless, despite its benefits, the hybrid system encounters several challenges. A key concern is data sparsity, as the lack of sufficient user interaction data can impede the effectiveness of both Collaborative Filtering and Content-Based Filtering. Scalability also remains an issue, as the system may struggle to process large datasets efficiently, particularly as the number of users and items increases. Furthermore, the system's performance can be constrained when applied to more extensive or diverse applications, since combining multiple techniques can introduce complexity and slower processing times, ultimately limiting its practicality in larger, real-world scenarios.

2.5 Title: Course Popularity and Recommendation Using K-Means and FP-Growth Algorithm

Author: M. M. Rahman et al.

Algorithm Used: K-means clustering and FP-growth algorithm

M. M. Rahman et al. propose a course popularity and recommendation system that integrates K-means clustering with the FP-growth algorithm. It is employed to group students according to their course preferences and behaviours, while the FP-growth algorithm is used to identify frequent itemsets and associations between courses. This hybrid model aims to generate course recommendations based on both student clustering and patterns of course popularity, providing personalized suggestions that correspond with group preferences and frequently selected courses.

However, despite its advantages, the system also faces certain limitations. One significant challenge is the reduced accuracy arising from the complex interdependencies between courses, which can be difficult to model using clustering and association techniques alone. These complexities may lead to less accurate recommendations, particularly when courses exhibit multiple dependencies that are not easily captured. Moreover, the system tends to overlook nuanced individual preferences that do not align with the broader group patterns or frequent course associations, potentially resulting in recommendations that are less personalized for specific students.

2.6 Title: Hybrid Recommendation System Using Genetic Algorithms

Author: A. Esteban et al.

Algorithm Used: Collaborative Filtering (CF), Content-Based Filtering (CBF), and Genetic Algorithms (GA)

A. Esteban et al. introduce a hybrid recommendation system that merges Collaborative Filtering (CF), Content-Based Filtering (CBF), and Genetic Algorithms (GA). This integration aims to enhance both the accuracy and personalization of recommendations. Collaborative Filtering relies on user behavior

and preferences, Content-Based Filtering focuses on the attributes of items, and Genetic Algorithms refine the recommendation process by evolving optimal solutions. The combination of these methods enables the system to adapt to user preferences over time, providing more relevant and diverse recommendations.

However, the hybrid system comes with certain drawbacks. One major issue is data sparsity, where a lack of sufficient user interaction data can limit the effectiveness of both Collaborative Filtering and Content-Based Filtering. Scalability is also a concern, as the system may struggle to manage large datasets or handle a growing number of users and items efficiently. These limitations could impact the overall performance of the system, especially in large-scale applications.

2.7 Title: Advances in Collaborative Filtering

Authors: Koren, Y., Rendle, S., Bell, R. (2022)

Algorithm Used: Collaborative Filtering (CF)

In their 2022 work, Koren, Rendle, and Bell present notable advancements in Collaborative Filtering (CF) techniques, which are fundamental to many modern recommendation systems. CF works by analyzing user behavior to detect patterns and generate recommendations based on the preferences of similar users. The authors discuss the progression of CF methods, which have evolved to enhance their efficiency and accuracy. However, despite these improvements, the foundational challenges of CF still persist, especially when handling large-scale and complex datasets, as user-item interactions grow in variety and volume.

The primary drawbacks of Collaborative Filtering, as identified in the work, include data sparsity, where CF struggles to make accurate recommendations when users have rated only a few items. This limitation makes it difficult to identify similar users and effectively predict their preferences. Scalability is another concern, as CF systems often experience performance issues when the number of users and items expands. Furthermore, the cold start problem occurs when new users or items have insufficient historical data, resulting in suboptimal recommendations. Lastly, CF methods often function as a "black-box," making it challenging to interpret why certain recommendations are made, which can erode transparency and trust in the system.

2.8 Title: Matrix Factorization with Rating Completion: An Enhanced Model for

Collaborative Filtering Recommender Systems

Authors: Guan, Xin; Chang-Tsun Li; Yu Guan (2017)

Algorithm Used: Matrix Factorization with Rating Completion.

This study enhances traditional matrix factorization techniques for collaborative filtering by incorporating a rating completion method. It addresses data sparsity in user-item interactions by

imputing missing ratings before decomposition. The proposed model improves recommendation accuracy and system reliability.

Higher computational requirements due to rating imputation.

Challenges in addressing cold-start issues where there is insufficient data.

Quality of predictions depends heavily on accurate estimation of missing values, which can sometimes lead to biased recommendations.

2.9 Title: Build a Recommendation Engine with Collaborative Filtering

Author: Ajitsaria, Abhinav (2019)

Algorithm Used: Collaborative Filtering

This article provides a comprehensive tutorial on developing a recommendation engine using collaborative filtering. It demonstrates practical methods to analyze user behaviour and item preferences by identifying latent factors and user-item relationships. The guide focuses on practical implementation for building personalized recommendation systems.

Prone to overfitting if the system is not well-regularized.

Handling large datasets can lead to increased computational and memory costs.

The quality of recommendations is affected without proper preprocessing, including normalization and noise reduction.

2.10 Title: Intelligent Personalized Recommendation Method Based on Optimized Collaborative Filtering Algorithm in Primary and Secondary Education Resource System

Author: Feixiang, Xie (2024)

Algorithm Used: Optimized Collaborative Filtering

Feixiang and Xie (2024) introduce an intelligent personalized recommendation method using an optimized Collaborative Filtering (CF) algorithm, designed specifically for the primary and secondary education resource system. The method aims to deliver tailored recommendations to students by analyzing their interactions with educational materials. By enhancing the traditional CF algorithm, the system intends to improve recommendation accuracy and relevance, thereby enriching the learning experience for students. The optimization process fine-tunes the recommendation engine, allowing it to suggest resources aligned with individual preferences and educational needs.

Despite its potential, the system encounters several drawbacks. Data sparsity presents a significant issue, as CF-based methods often struggle in environments like education, where user interaction with resources can be limited. Additionally, the cold start problem arises when new users or educational resources lack sufficient interaction data, making it challenging to generate meaningful recommendations. The system also faces scalability challenges, as its performance may decline when

the number of users or resources increases, even with optimization. Lastly, the system's reliance on historical data may hinder its ability to adapt to evolving user preferences and trends, limiting its flexibility over time.

2.11 Title: Simple Autoencoders for Collaborative Filtering

Authors: Hong, Seoyoung, et al. (2024)

Algorithm Used: Autoencoders for Collaborative Filtering

The paper introduces a novel approach utilizing simple autoencoders for collaborative filtering. The autoencoder, a neural network architecture, is used to reconstruct user-item interaction data and extract meaningful latent features. These features are then used to make recommendations based on shared preferences and patterns among users and items. The framework provides an efficient way to leverage deep learning without overly complicating the system.

Extracts latent features representing user and item preferences for better accuracy. Achieves simplicity while enhancing recommendation quality compared to complex deep learning models. Suitable for datasets where capturing non-linear relationships improves predictions.

Autoencoder architecture is sensitive to design choices like the number of layers and nodes. Computational cost increases for complex architectures and large datasets. Does not resolve challenges like the cold-start problem or highly sparse datasets.

The algorithm used in the paper "Simple Autoencoders for Collaborative Filtering" revolves around autoencoders for collaborative filtering, a neural network-based approach that reconstructs user-item interaction data to uncover latent features. In this technique, the autoencoder architecture consists of two main components: the encoder and the decoder. The encoder functions to compress the high-dimensional input data, which in this case is the user-item interaction matrix, into a much lower-dimensional latent representation. This latent space captures the essential features or patterns within the data, which could involve both users' preferences and items' attributes in a way that enables better predictions. The decoder then takes this latent representation and attempts to reconstruct the original input, producing the estimated user-item interactions or predictions for missing data points. This model is trained to minimize the reconstruction error, effectively enabling the autoencoder to learn significant and non-linear correlations between users and items.

The autoencoder-based model is particularly suitable for datasets where linear models struggle to capture intricate relationships between data points. It captures underlying preferences or features that contribute to users' decision-making, such as an implicit bias toward certain genres of movies or types of music, which aren't always explicitly reflected in the available interaction history. Furthermore, by reducing the complexity of training through simplicity in architecture compared to deeper or more

intricate networks, this model enables scalability and ease of deployment for moderate-sized recommendation tasks. Nevertheless, it should be noted that the design choices of the autoencoder, such as the number of hidden layers or the number of nodes per layer, are crucial for the model's performance, with too few layers possibly failing to capture complex patterns, and too many potentially leading to overfitting or unnecessary computational expense. Despite these considerations, the autoencoder offers an elegant and efficient solution for generating personalized recommendations without the heavy reliance on overly sophisticated and computationally expensive techniques commonly seen in traditional deep learning methods.

2.12 Title: Dual-Embedding Based Neural Collaborative Filtering for Recommender Systems

Authors: He, Gongshan; Dongxing Zhao; Lixin Ding (2021)

Algorithm Used: Dual-Embedding Neural Collaborative Filtering

This study proposes a neural-based dual-embedding framework where separate representations (embeddings) are learned for users and items. These embeddings are processed through non-linear layers in the neural model to extract detailed interactions and improve recommendations. The method is designed for personalized suggestions by capturing subtle and complex relationships in data. Introduces dual embeddings for users and items, capturing distinct yet complementary characteristics. Applies deep learning to enhance recommendations using more precise patterns in user-item data. Improves scalability in handling large datasets while boosting prediction accuracy. Neural models require extensive computational resources during training. High sensitivity to hyperparameters and potential overfitting if not regularized properly. Relies on high-quality data; performance declines with noise or inconsistencies in user interactions.

In "Dual-Embedding Based Neural Collaborative Filtering for Recommender Systems," the focus shifts to dual-embedding neural collaborative filtering. This method learns two separate embeddings, one for users and one for items, to better represent their characteristics within the recommendation system. The dual embeddings approach captures user and item features individually, and then combines them to model the interactions between users and items. These embeddings are passed through non-linear layers to create a powerful mechanism for extracting complex relationships in the data that traditional approaches may not capture. This technique allows the system to offer personalized recommendations, capturing nuanced and subtle interactions based on learned embeddings, such as understanding that a particular user might prefer action movies more when paired with a specific director or actor, or an item like a product being closely linked to certain customer behaviors and preferences. The goal of this method is to ensure that the final recommendation is more accurate and personalized, improving the relevance of the suggestions for users. Moreover, neural

models enable the system to scale better to large datasets as opposed to older algorithms that may falter when faced with vast amounts of user-item data. However, as a neural-based approach, dual-embedding collaborative filtering is computationally intensive during training and sensitive to the tuning of hyperparameters. Unoptimized settings might result in overfitting or poor generalization, causing the model to perform inconsistently across different types of data.

2.13 Title: Revisiting Matrix Decomposition to Generate Powerful Node Embeddings for Recommendation Systems

Author: Budhiraja, Amar (2021)

Algorithm Used: Node Embedding Generation

In this work, the focus is on creating node embeddings for users and items in recommendation systems. These embeddings capture relationships between nodes in the data graph, enabling more personalized recommendations. The proposed method re-evaluates traditional approaches for embedding generation, optimizing them to meet modern challenges in recommendation tasks. Generates compact and high-quality embeddings that improve model predictions. Reduces the complexity of graph-based methods for scalability. Offers enhanced accuracy through optimized embedding representation. Node embedding quality depends heavily on preprocessing techniques. Struggles to scale seamlessly for massive datasets. Noise and outliers in data can significantly degrade performance without proper cleaning.

Another interesting approach is used in Node Embedding Generation, as explored by Budhiraja in the paper "Revisiting Matrix Decomposition to Generate Powerful Node Embeddings for Recommendation Systems." This approach focuses on node embeddings, where users and items in a recommendation system are treated as nodes in a graph. The relationships between users and items are encoded as edges in this graph, and node embeddings are generated to capture these relationships. The embeddings represent the properties of the nodes, helping the system identify and understand which items might appeal to specific users based on the learned connections. This method significantly improves the flexibility and accuracy of recommendation models by focusing on building compact and high-quality representations of the users and items that can be used for generating predictions. Node embeddings allow the system to capture highly personalized, intricate interactions without the need to deal with large matrices directly, thus reducing computational overhead. Nonetheless, the performance of node embedding models depends significantly on the data preprocessing stages, as the embeddings may suffer if the graph or interaction history is noisy. Additionally, while node embedding generation offers scalability benefits compared to older methods, it is not entirely immune to the challenges posed by extremely large datasets, particularly when handling vast numbers of users and items.

2.14 Title: E-Learning Course Recommender System Using Collaborative Filtering Models**Authors:** Jena, Kalyan Kumar, et al. (2022)**Algorithm Used:** Collaborative Filtering

This paper designs a recommender system tailored to e-learning platforms. It analyzes user data, such as course interaction history, preferences, and feedback, to suggest personalized courses. Collaborative filtering is used to identify commonalities between users and recommend courses based on shared interests and performance patterns.

Matches users with courses aligned to their skills and learning preferences. Improves user retention through tailored educational suggestions. Dynamically updates suggestions based on ongoing user interactions. Suffers from sparsity in data when many users interact with only a few courses. Cold-start issues arise for new users or recently introduced courses.

Lastly, in the Collaborative Filtering method for e-learning platforms described by Jena and colleagues in their 2022 paper, a collaborative filtering approach is used to analyze historical data about course interactions to suggest personalized courses. Collaborative filtering identifies patterns of user-item interactions, where the system suggests courses based on how similar other users with similar preferences or behaviours have rated them. This type of algorithm works by finding groups of similar users or items and using these relationships to make accurate predictions of what an individual user might prefer. However, challenges remain when data is sparse, as may be the case in e-learning systems where students may only interact with a handful of courses, making it difficult to identify these patterns. Collaborative filtering also struggles with new users or new courses (the cold-start problem), as there is not enough interaction data available to determine meaningful similarities. Despite these challenges, collaborative filtering remains a popular method because it is straightforward and can effectively leverage user interactions for producing reliable recommendations, making it highly effective for personalization in many practical applications.

2.15 Title: Online Course Recommender System for E-Learning Platforms**Authors:** Bahar, Musthafa Zaki, and Z. K. A. Baizal (2023)**Algorithm Used:** Matrix Factorization for Course Recommendation

This study presents an online course recommender system tailored for e-learning platforms. The authors employ matrix factorization techniques to capture and analyze user preferences, learning histories, and course attributes. By decomposing the user-item interaction matrix, latent relationships are uncovered, which form the basis for personalized course recommendations.

Suitable for adaptive learning, allowing the platform to suggest courses dynamically as user preferences evolve. Addresses sparsity in course-user interactions by capturing hidden patterns in data.

Improves user engagement and course completion rates through tailored suggestions. Struggles to provide reliable recommendations in cold-start scenarios, where users or courses lack sufficient data. Relies on comprehensive user profiles; poor data quality reduces accuracy. Computational overhead increases with large-scale datasets.

In the paper "Online Course Recommender System for E-Learning Platforms," Bahar, Musthafa Zaki, and Z. K. A. Baizal (2023) focus on matrix factorization for course recommendation. This method captures latent relationships between users and courses by decomposing the user-item interaction matrix into two lower-rank matrices representing users and courses. The factorization reveals patterns in how users interact with courses and identifies similar courses or users to make personalized recommendations. Matrix factorization helps to address the sparsity problem often seen in e-learning platforms, where not all users interact with all courses. By uncovering hidden patterns within sparse data, the algorithm improves the accuracy of recommendations, increasing user engagement and course completion rates. However, matrix factorization struggles with the cold-start problem, as it requires substantial interaction data to provide reliable recommendations. In cold-start scenarios, the lack of interaction data for new users or courses poses a challenge. Furthermore, the quality of recommendations is heavily reliant on the completeness and accuracy of the user profiles, and if data is of low quality or incomplete, the performance of the model can be compromised. The approach also experiences computational overhead with large-scale datasets, requiring considerable processing power to maintain the factorization of the matrices as the user base grows.

2.16 Title: Privacy-Aware Collaborative Filtering Models for Recommender Systems

Authors: Polat, Huseyin, and Wenliang Du (2005)

Algorithm Used: Privacy-Preserving Collaborative Filtering

This paper explores privacy-aware techniques in collaborative filtering systems. The authors propose methods that preserve user privacy while enabling accurate recommendation models by anonymizing sensitive user data and obfuscating identifiable information. Their approach ensures that personal data is not directly exposed while still enabling reliable predictions. Prioritizes user data confidentiality without compromising recommendation quality. Introduces mechanisms for anonymization to align with privacy regulations. Offers solutions for secure computation of recommendations in distributed settings. Trade-off between privacy and prediction accuracy; obfuscation might reduce the quality of recommendations. Adds computational overhead due to data anonymization processes. Requires carefully designed algorithms to balance security and performance.

In "Privacy-Aware Collaborative Filtering Models for Recommender Systems" , Polat and Wenliang Du introduce privacy-preserving collaborative filtering, which ensures that users' personal data

remains secure while still enabling accurate recommendations. This method addresses growing concerns over privacy and data security in collaborative filtering systems, often employed in recommender systems. It achieves privacy preservation by anonymizing sensitive user data and obfuscating identifiable information, allowing the system to provide personalized recommendations while protecting user identities. The approach aligns with privacy regulations, ensuring secure computation of recommendations, especially in distributed settings. While the system addresses important concerns around privacy, it faces trade-offs. One such trade-off is the balance between maintaining user confidentiality and ensuring the accuracy of recommendations. The obfuscation of data may lead to a decrease in prediction accuracy, as the system relies on less granular information about users and their preferences. Furthermore, adding privacy-preserving measures introduces additional computational overhead, impacting the efficiency of the recommendation process. To mitigate these issues, privacy-aware systems require carefully designed algorithms that balance security with the quality of recommendations.

2.17 Title: Matrix Factorization with Rating Completion for Collaborative Filtering

Authors: Guan, Xin, Chang-Tsun Li, and Yu Guan (2017)

Algorithm Used: Matrix Factorization with Imputation

In this work, the authors enhance collaborative filtering by introducing a rating completion method. This approach imputes missing ratings in the user-item matrix, ensuring data sparsity is addressed before analyzing latent patterns. Rating completion helps improve prediction accuracy and robustness in sparse datasets, which are common in real-world recommendation systems. Tackles data sparsity by predicting missing interactions prior to model training. Improves recommendation quality through a more complete dataset. Effectively integrates into traditional matrix factorization pipelines.

Imputation methods add significant computational overhead. High dependency on accurate imputation algorithms; errors in predictions propagate through the system. Does not adequately address cold-start problems.

In "Matrix Factorization with Rating Completion for Collaborative Filtering" by Guan, Xin, Chang-Tsun Li, and Yu Guan, the authors propose an enhancement to matrix factorization with a rating completion method. The matrix factorization technique is used in conjunction with an imputation process, where missing ratings in the user-item interaction matrix are estimated before performing the factorization. This additional step addresses the data sparsity problem, ensuring that missing interactions do not significantly hinder the accuracy of the recommendation model. Rating completion helps create a more comprehensive dataset by predicting missing data points, thereby improving the quality of the latent features learned during matrix factorization. Despite its strengths in handling

sparse datasets, the method is not without challenges. One drawback is that imputation introduces significant computational overhead, as predicting missing ratings before training adds complexity to the overall system. Additionally, the accuracy of the imputed ratings depends on the quality of the imputation algorithm, and errors in the imputed ratings can propagate through the recommendation process, potentially degrading the overall performance. Another issue is that this approach, like many collaborative filtering techniques, does not fully address the cold-start problem, particularly when it comes to new users or items that have little to no historical interaction data.

2.18 Title: Fast Online Matrix Updates for Lightweight Recommender Systems

Author: Brand, Matthew (2003)

Algorithm Used: Online Matrix Factorization

Brand's work focuses on developing lightweight, fast, and efficient methods for updating matrix factorization models in real-time recommender systems. This is particularly useful for systems that require quick adaptation to new user interactions or items without re-training the entire model. The proposed method handles incremental data efficiently, making it suitable for large-scale, dynamic environments. Enables real-time updates for recommender systems without full model re-training. Reduces computational cost, making it suitable for resource-constrained applications. Improves scalability in dynamic environments, such as e-commerce or streaming platforms.

Efficiency comes at the cost of lower accuracy compared to full model re-training. May struggle with long-term accuracy as it focuses on immediate updates. Sensitive to data quality, requiring frequent validation.

Matthew Brand's work in "Fast Online Matrix Updates for Lightweight Recommender Systems" presents online matrix factorization, a technique that offers real-time updates to recommender systems. This method is especially useful for systems that need to adapt quickly to new data, such as platforms with rapidly changing user interactions or evolving item catalogues. Instead of re-training the entire model from scratch with every new data point, online matrix factorization incrementally updates the existing model in real time. This allows the system to provide timely recommendations based on the most recent interactions. The method's efficiency makes it ideal for large-scale, dynamic environments, such as e-commerce or streaming platforms, where the data continuously evolves. However, the trade-off in using online matrix factorization is that the focus on quick adaptation can come at the expense of long-term accuracy, as the model may not incorporate a sufficient amount of historical context or trends. Additionally, the efficiency gained by real-time updates may be accompanied by a slight reduction in recommendation quality, as the model may not have the time to fully adjust to recent changes in user behaviour or preferences. Finally, as online matrix factorization

heavily relies on frequent updates, it is sensitive to data quality, requiring constant validation to prevent errors from accumulating over time.

2.19 Title: Build a Recommendation Engine with Collaborative Filtering

Author: Ajitsaria, Abhinav (2019)

Algorithm Used: Collaborative Filtering

This tutorial provides a hands-on approach to building recommendation engines using collaborative filtering. The article covers the implementation of user-based and item-based collaborative filtering techniques, emphasizing practical applications. It demonstrates how to calculate user similarity, predict missing ratings, and generate personalized recommendations.

Easy-to-follow implementation for learners and practitioners. Focuses on real-world examples and use cases. Demonstrates both memory-based and model-based collaborative filtering. Simplistic models; lacks advanced optimization techniques found in modern algorithms. Performance diminishes on large-scale datasets due to computational inefficiency. Relies heavily on data preprocessing for meaningful results.

Lastly, in "Build a Recommendation Engine with Collaborative Filtering" (2019), Ajitsaria presents a practical tutorial on building recommendation engines using collaborative filtering. This paper illustrates the implementation of both user-based and item-based collaborative filtering techniques, providing hands-on guidance for learners and practitioners. Collaborative filtering is one of the most widely used methods in recommendation systems, relying on past behaviour to identify similarities between users or items. In user-based collaborative filtering, the algorithm suggests items to a user based on the preferences of similar users, while item-based collaborative filtering suggests similar items to those the user has interacted with. Despite being a simple and easy-to-understand approach, collaborative filtering has limitations, particularly in large-scale applications. As the dataset grows, both memory-based and model-based collaborative filtering techniques can become computationally inefficient, requiring substantial resources to calculate similarities and generate recommendations. Furthermore, the accuracy of the recommendations heavily depends on the data preprocessing stage, making it essential to clean and format the data properly for the system to perform optimally. Lastly, while the technique is straightforward and accessible, its simplistic nature limits its ability to handle complex user-item relationships, making it less suitable for systems requiring advanced optimization techniques.

2.20 Title: Collaborative Filtering via Matrix Approximation for Recommendation Systems

Authors: Zhang, Sheng, et al. (2005)

Algorithm Used: Matrix Approximation

Zhang et al. investigate matrix approximation methods to create a simpler and more computationally efficient alternative for collaborative filtering. Their work highlights how approximate methods can accelerate computations and reduce memory usage while maintaining reasonably accurate recommendations. The paper provides insights into using such approximations for real-time recommender systems in e-commerce applications.

Improves scalability by reducing computational load. Adaptable for dynamic environments where speed is critical. Retains acceptable recommendation quality while reducing model complexity. Approximation techniques may oversimplify complex user-item relationships. Prone to performance degradation in highly sparse or noisy datasets. Offers limited improvement over full-fledged matrix factorization techniques for large, diverse datasets.

The paper by Zhang, Sheng, et al. explores an approach for collaborative filtering that leverages matrix factorization approximation to address efficiency and computational challenges in recommendation systems. This method involves decomposing the user-item interaction matrix into smaller matrices representing latent features of users and items, along with a matrix of singular values. By approximating the matrix factorization, only the most significant factors are retained, which reduces the complexity and memory requirements of the model. This approximation captures the key patterns in user interactions and item characteristics, facilitating better predictions for missing ratings and generating relevant recommendations.

The primary advantage of this approach is its ability to scale and improve computational efficiency, particularly in large and sparse datasets. It simplifies the computation and reduces the overhead required for processing user and item data. However, challenges still remain. The quality of the approximated model is highly dependent on how many factors are retained, and selecting too few can result in underfitting, while keeping too many may lead to overfitting. Additionally, this method still faces difficulties in handling new users or items with limited data, and performance may decline in datasets with substantial noise or outliers, affecting recommendation accuracy.

CHAPTER-3

RESEARCH GAPS OF EXISTING METHODS

3.1 Sparse Data in Recommendation Systems (Sparse RSs) Recommendation systems often operate in a sparse environment, meaning there is a lack of sufficient data to draw accurate inferences from. This is commonly referred to as data sparsity. When you have limited data on a user's preferences or what others like, it's difficult for the system to make useful predictions.

Example: Imagine a movie recommendation system where a new user has watched only one movie, and there are thousands of movies available. The system has very little information to accurately recommend new movies.

Challenges:

- **Lack of Personalization:** Limited data on users' preferences makes it harder for the system to personalize recommendations.
- **Solution:**
- **Matrix Factorization and Deep Learning** techniques help alleviate sparsity by learning latent factors from the available data.
- **Clustering:** Grouping users with similar tastes to boost prediction accuracy.

3.2 Cold Start Problem: The cold-start problem arises when a system does not have enough data about either a new user or a new item, which makes it challenging to produce accurate recommendations. This issue is particularly problematic in recommendation systems, where data is essential for making reliable predictions. The lack of prior interaction or historical data for new users or items limits the system's ability to effectively predict preferences and suggest relevant content.

Example: A new movie released on a streaming platform that hasn't been rated yet by users or reviewed.

Challenges:

- **For New Users:** If a new user joins a platform, the system doesn't know their preferences and therefore cannot make personalized recommendations.
- **For New Items:** New items (movies, products, etc.) have no ratings or reviews, so it's challenging to recommend them to users.

Solution:

- **Hybrid Methods:** Combining collaborative filtering (which depends on user-item interaction data) with content-based filtering (which uses metadata about items or users) can alleviate the cold-start problem.
- **Active Learning:** Systems can prompt users to provide feedback early on (e.g., rating some items to jumpstart the recommendation process).

3.3 Scalability Problems: As platforms grow, so does the volume of users and items. Scalability becomes a critical issue as the system must handle large amounts of data efficiently. For large-scale platforms like Amazon, YouTube, or Netflix, the system must be able to process millions of user-item interactions in real-time.

Example: Processing recommendations for millions of users simultaneously during peak usage times.

Challenges:

- **Data Processing:** The system must process a large amount of data quickly to generate recommendations in real-time.
- **Computation:** More users and items require more computation, and as systems scale, the time to make predictions increases.

Solution:

- **Approximation Algorithms:** Techniques like ALS (Alternating Least Squares), k-means clustering, and other dimensionality-reduction methods can help make real-time predictions with lower computational overhead.

3.4 Privacy Issues: Recommendation systems heavily depend on user data to generate personalized suggestions, which may include browsing history, purchase behavior, location, and other personal preferences. This reliance on personal data raises privacy concerns, as users may not want their data to be collected, stored, or shared without their explicit consent.

Challenges:

- **Data Collection:** Users may be concerned about the volume and type of data being collected, as well as how it is being utilized.
- **Data Security:** Ensuring that users' data remains secure and protected from unauthorized access or misuse is a key challenge.

Solution:

- **Differential Privacy:** Implementing differential privacy allows systems to provide

personalized recommendations without exposing sensitive information.

- **Federated Learning:** This allows model training to occur on the user's device, thus keeping data on the device rather than uploading it to a central server.
- **Transparency:** Systems should provide clear and transparent explanations of how data is being used, allowing users to control their preferences.

3.5 Robustness of Recommendation Systems: Recommendation systems are vulnerable to attacks that can distort the accuracy of recommendations. One common issue is shilling attacks, where malicious users deliberately inject fake preferences to manipulate recommendations.

Example: An attacker might create multiple fake user profiles to artificially inflate the popularity of a product or course.

Challenges:

- **Manipulation of Recommendations:** Malicious attacks can degrade the quality of recommendations by inflating certain items or user preferences, leading to inaccurate recommendations.
- **Security Risks:** The robustness of the system is tested when attackers introduce spurious data to influence the recommendations.

Solution:

- **Anomaly Detection:** Implementing methods to detect unusual patterns in user behavior can help mitigate shilling attacks.
- **Robust Algorithms:** Using robust matrix factorization and outlier detection techniques can make the system more resistant to manipulation.
- **User Authentication:** Preventing fake profiles through more stringent user authentication methods can reduce the risk of such attacks.

3.6 Interpretability Issues: Many recommendation systems, especially those based on deep learning or matrix factorization, function as a black box, making it difficult for users or developers to understand why a particular recommendation was made.

Challenges:

- **Lack of Transparency:** Users may not trust a recommendation if they cannot understand the rationale behind it.
- **Explainability:** The lack of explainability in how recommendations are generated can hinder user confidence and adoption.

CHAPTER-4

PROPOSED MOTHODOLOGY

Singular Value Decomposition (SVD) in Recommendation Systems:

SVD is a widely used technique in recommendation systems, especially for dimensionality reduction, latent feature extraction, and predicting missing data. By decomposing large matrices into smaller, more manageable ones, SVD enables the system to make efficient and accurate recommendations.

How SVD Works:

4.1 The Matrix Setup:

In recommendation systems, the matrix represents the relationship between users and items (e.g., movies, courses, products, etc.).

Rows represent users.

Columns represent items (movies, books, courses, etc.).

Cells represent the ratings provided by users for specific items (if available). Missing ratings are treated as zeros or left blank.

4.2 SVD Decomposition:

SVD decomposes the user-item interaction matrix R into three smaller matrices:

$$R \approx U \Sigma V^T$$

R is the original user-item matrix containing ratings.

U is a matrix representing user features (preferences).

Σ (Sigma) is a diagonal matrix of singular values indicating the strength of each latent feature.

V^T is a matrix representing item features (characteristics).

4.3 Step-by-Step Decomposition:

Decompose the Matrix: The SVD algorithm decomposes R into U , Σ , and V^T .

U : Contains the latent user factors (preferences). Each row is a user in the latent feature space.

Σ : A diagonal matrix of singular values, which measure the importance of each latent feature.

V^T : Contains item factors, where each column represents an item in the latent feature space.

Latent Features: Singular values in Σ indicate the relative importance of each latent feature. The larger the singular value, the more important the corresponding feature.

Reducing Dimensions: This removes noise and focuses on the most significant features. For example, if $k = 3$, the data is reduced to 3 latent features.

Prediction: Once the matrices are decomposed, you can predict the missing entries in the user-item matrix R_{ij} (i.e., a missing rating for user i and item j) using the following formula:

$$R_{ij} \approx U_i \cdot \Sigma \cdot V_j^T$$

U_i is the vector of user i 's preferences across the latent features.

V_j^T is the vector of item j 's attributes across the latent features.

Reconstruction: After applying SVD and selecting the top k features, the reconstructed matrix R approximates the original user-item matrix. The missing entries are now predicted based on the latent factors discovered through SVD.

SVD Algorithm in the Elective Recommendation System:

The system employs SVD to recommend courses by analyzing student-course-grade data. SVD is used to break the data into smaller matrices to identify hidden patterns that reveal student preferences and course features.

User Experience in the Elective Recommendation System:

The Elective Recommendation System is designed to ensure a seamless, user-friendly experience for students by guiding them through the process of obtaining personalized course recommendations.

Index Page:

The system's index page serves as the starting point where students input their personal details such as their name, registration number, academic background, and any specific preferences regarding their course choices. The form is structured logically, allowing users to easily understand what information is required and how to input it. This simplicity and clarity provide a hassle-free experience and ensure that students can submit their information quickly and accurately. Clear instructions are available to assist users with any queries about the input process, eliminating confusion and ensuring that even those who are not very tech-savvy can navigate the system without difficulty.

Recommendation Page:

After the student has filled in their personal information, they are redirected to a recommendation page that dynamically displays a list of courses tailored to their academic needs and preferences. This page is designed to present these suggestions in an organized and visually appealing way, highlighting the most relevant courses based on both the student's profile and the data processed by the Singular Value Decomposition (SVD) algorithm. In addition to presenting course recommendations, the system also provides additional details such as course descriptions, prerequisites, and the predicted match percentage based on the student's input, helping them make more informed decisions.

The user interface is responsive, adjusting to different devices and screen sizes, which enhances accessibility and ensures that students can access the recommendation system across various platforms (PC, tablets, or mobile phones).

Performance:

The Elective Recommendation System has demonstrated significant improvements in recommendation accuracy, achieving an accuracy rating of 0.86 in its current iteration, reflecting the system's heightened ability to generate precise and relevant course recommendations.

Enhanced Accuracy with SVD:

The system's enhancement comes primarily from the application of the Singular Value Decomposition (SVD) algorithm, which helps in breaking down large and complex datasets into simpler forms, making it easier to detect patterns and predict student preferences. By incorporating SVD, the system has dramatically improved its prediction abilities, taking into account latent factors that weren't previously considered with standard techniques. The more accurate the predictions, the more likely students are to select courses that suit their interests and academic trajectory.

Personalization:

Personalized recommendations are a cornerstone of the system. The continuous refinement of the SVD model based on user feedback and data collection enhances the accuracy of future recommendations. Over time, as the system processes more data from students' interactions, its recommendations will become even more customized, helping students not only in selecting their electives but also in guiding them towards courses that align with their future goals, academic strengths, and learning preferences.

Scalability & Long-term Effectiveness:

The scalability of the system is a crucial aspect, as it ensures that the system remains effective as the number of students and course offerings grows. Even with an increased volume of data, the system is designed to maintain or improve its recommendation quality. The effectiveness of SVD in maintaining high-quality predictions, even with a large set of users and items (courses), demonstrates the system's capability to support large educational institutions over the long term.

CHAPTER-5

OBJECTIVES

5.1 Addressing the Cold-Start Issue: A common challenge in course recommendation systems is the cold-start problem, which occurs when there is insufficient data for new users or courses. To address this issue, the SVD-based system employs matrix factorization techniques to uncover latent features, allowing the system to make informed predictions even with limited data. These latent features are based on user-item interactions, meaning the system can infer preferences and recommend courses despite the lack of historical data for new students or electives.

5.2 Scalability and Flexibility: The SVD algorithm is designed with scalability in mind, meaning it can handle an increasing number of users, courses, and data points efficiently. As more students and electives are added to the system, the underlying model should maintain, or even improve, its prediction accuracy. The goal is to ensure that the system can continue to deliver high-quality recommendations without suffering from performance degradation, making it a sustainable and effective solution for larger academic institutions with vast datasets.

5.3 Reducing Data Sparsity: One of the key strengths of the SVD-based recommendation system is its ability to address data sparsity. When there are few explicit ratings or interactions available, SVD extracts latent features from the incomplete dataset to uncover hidden patterns. These patterns allow the system to make meaningful course recommendations even when the available data is sparse. By leveraging the underlying structure of the user-item matrix, the system ensures that students continue to receive relevant course suggestions despite a lack of comprehensive feedback.

CHAPTER-6

SYSTEM DESIGN & IMPLEMENTATION

The system architecture for a course recommendation system integrates four key modules:

6.1 Data Collection:

The system collects data from students, such as their previous course enrolments, academic performances, preferences, and ratings for courses they have taken. This information is stored in a structured format, usually in a database or through data sources like a Learning Management System (LMS).

6.2 Data Processing:

Raw data undergoes preprocessing to clean and transform it into a usable format. Missing values, outliers, and inconsistencies are handled using techniques like imputation, normalization, and data augmentation. Libraries such as **pandas** in Python are typically used to handle data manipulation and processing tasks efficiently.

6.3 Generating Recommendations:

In this process, matrix factorization techniques like Singular Value Decomposition (SVD) play a crucial role in discovering hidden patterns within student preferences and their past course choices..

6.4 User Interaction:

After recommendations are generated, the system provides users (students) with a personalized list of course options. User feedback can be incorporated to refine the recommendations in future iterations, creating a feedback loop for continuous system improvement.

6.5 Implementation Details:

6.5.1 Technologies and Tools:

Python:

Python is the primary programming language used for building the recommendation engine due to its rich ecosystem of libraries and frameworks that support data analysis, machine learning, and natural language processing.

pandas:

pandas is a widely used Python library that provides powerful data manipulation and analysis tools. It includes DataFrames, a flexible data structure that allows efficient handling and transformation of data, making tasks like data cleaning, loading, and manipulation easier.

Machine Learning Libraries:

Libraries such as scikit-learn, TensorFlow, and PyTorch are integral to the development and training of recommendation models. These libraries offer an array of pre-built functions for building machine learning models, optimizing them, and evaluating their performance. They provide a foundation for implementing both traditional machine learning and deep learning models.

6.5.2 Matrix Factorization in Singular Value Decomposition (SVD):

Singular Value Decomposition (SVD) is a widely used dimensionality reduction technique in filtering. The fundamental concept behind SVD is to factorize the user-item interaction matrix (such as course ratings or course enrollments) into three distinct matrices that capture the hidden relationships :

U (user matrix): Represents the latent factors for users (students in this case).

- **S (singular values matrix):** Contains singular values that indicate the importance of each factor.
- **V (item matrix):** Represents the latent factors for items (courses).

The equation is:

$$R \approx U \times S \times V^T \quad \text{or} \quad R \approx U \times S \times V^T$$

Where R is the original matrix representing user-course interactions.

SVD works by approximating this matrix using a smaller number of dimensions, which helps capture underlying patterns (like preferences for certain types of courses) and reduces noise. After applying SVD, the system can predict missing values in the matrix (i.e., unobserved course ratings), allowing it to recommend courses that the user might be interested in based on their previous choices and the preferences of similar users.

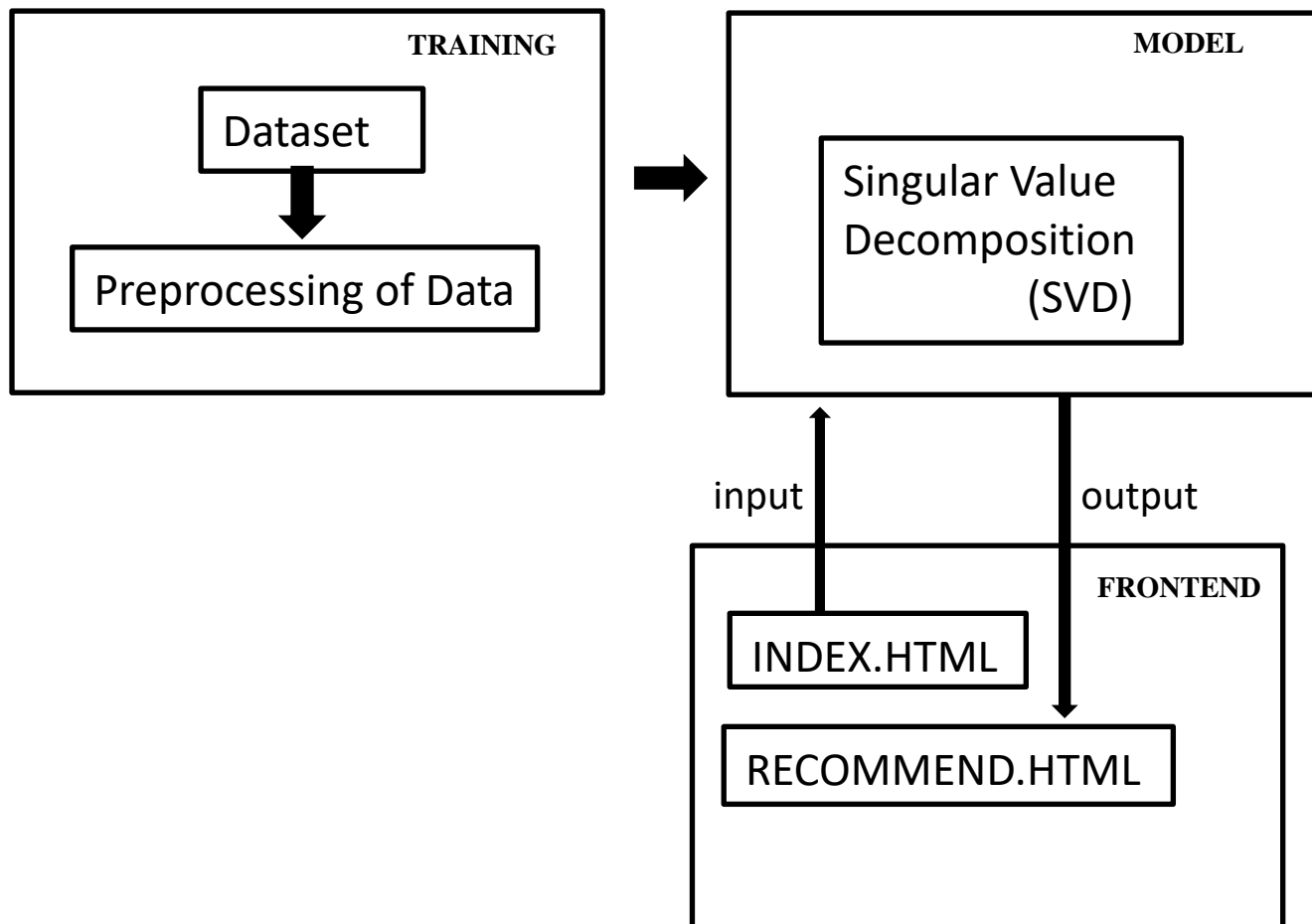


Fig 1.1 ARCHITECTURE DIAGRAM

The architecture of the Elective Recommendation System is divided into three main stages: Training, Model, and Frontend, each contributing to the system's functionality. The process begins with the Training stage, where the dataset, comprising student data like academic records and course studied is prepared. This data undergoes Preprocessing, which involves cleaning, handling missing values, normalizing, and structuring it to make it suitable for analysis. This step ensures the dataset is optimized for effective model training.

The Model stage utilizes the preprocessed data to build a recommendation model using Singular Value Decomposition (SVD). SVD is a collaborative filtering technique that identifies patterns in the dataset, establishing relationships between students and courses. By leveraging these patterns, the model generates tailored elective recommendations based on the user's input. The trained model plays a pivotal role in providing accurate and meaningful suggestions.

The Frontend forms consisting of two key webpages. The Index.html page serves as the input interface, where users enter their details, such as name, academic information, and preferences. Once the data is submitted, it is sent to the model for processing. The results are then displayed on the Recommend.html page, where students can view the elective courses suggested specifically for them. This seamless integration of the backend model and frontend interface ensures a user-friendly experience while simplifying the elective selection process.

1. Training Stage

The foundation of the recommendation system lies in the Training stage, where the dataset is carefully prepared to build an effective model. The dataset comprises student-related information, such as academic records, courses previously taken, preferences, and performance metrics. The process begins with:

a. Data Collection and Preparation

Relevant data is gathered and structured into a format suitable for analysis. This includes ensuring comprehensive coverage of various factors influencing elective choices, like subject relevance, prerequisites, and past preferences.

b. Data Preprocessing

To maximize model performance, the collected dataset undergoes preprocessing, which involves:

Cleaning: Removing inconsistencies, redundant entries, and irrelevant data.

Handling Missing Values: Replacing missing data with appropriate values or estimates to prevent skewed results.

Normalization: Scaling numeric attributes to ensure uniformity in data representation.

Structuring: Organizing the dataset into a format compatible with the training requirements of the recommendation model.

Preprocessing ensures that the dataset is optimized for model training by eliminating biases and inconsistencies.

2. Model Stage

The heart of the system is the Model stage, where the preprocessed data is leveraged to build a powerful recommendation engine.

a. Recommendation Model

A collaborative filtering technique known as Singular Value Decomposition (SVD) is employed to identify latent patterns and establish relationships between students and electives. The SVD model effectively decomposes the dataset into multiple components that reveal hidden structures in the data.

b. Process Flow

The SVD algorithm identifies similarities among students based on their academic records, preferences, and elective selections.

It also detects relationships between courses, such as prerequisites or thematic similarities.

By combining these insights, the model generates personalized elective recommendations tailored to individual student needs.

The trained model serves as the backend decision-making engine, ensuring the recommendations are accurate and meaningful based on user input.

3. Frontend Stage

The Frontend stage facilitates interaction between students and the system. It includes an intuitive user interface comprising two core webpages that seamlessly integrate with the backend model.

a. Index.html

This webpage acts as the input interface, where users provide their details, including:

Name and academic information (e.g., current semester, department, GPA).

Preferences, such as interests or topics they wish to explore.

Once the user submits this information, it is processed and sent to the backend model for analysis.

b. Recommend.html

After processing, the system displays the recommended electives on the second webpage.

The recommended courses are tailored specifically to the student's preferences and academic profile.

The page is designed to ensure clarity, enabling students to explore the suggested electives easily.

The system operates as a cohesive pipeline:

Students input their details via the Index.html page.

The backend training and recommendation model processes this input.

Recommendations are dynamically generated and displayed on the Recommend.html page.

This smooth integration of the three stages—Training, Model, and Frontend—ensures the system delivers an engaging and efficient experience, simplifying the complex task of elective selection.

By combining robust data preprocessing, intelligent recommendation techniques, and a user-friendly interface, the Elective Recommendation System not only empowers students to make informed decisions but also minimizes the effort required to explore and identify suitable elective courses.

CHAPTER-7

TIMELINE FOR EXECUTION OF PROJECT (GANTT CHART)

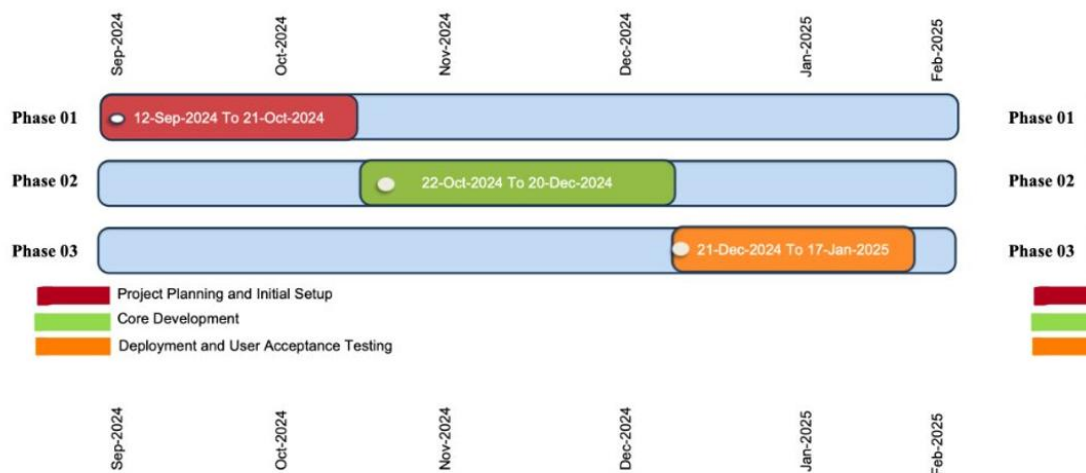


FIG 1.2 TIME LINE FOR EXECUTION OF PROJECT

TABLE 1.1 TIMELINE FOR EXECUTION OF THE PROJECT

| Sl. No | Review | Date | Scheduled Task |
|---------------|---------------|----------------------|--|
| 1 | Review-0 | 09-10-23 to 13-10-23 | Initial Project Planning |
| 2 | Review-1 | 23-10-23 to 02-11-23 | Planning and Research |
| 3 | Review-2 | 19-11-23 to 26-11-23 | Data Collection and Preprocessing, Model Implementation, Testing |
| 4 | Review-3 | 13-12-23 to 25-12-23 | Optimization |
| 5 | Viva-Voce | 01-01-25 to 12-01-25 | Deployment and Evaluation |

CHAPTER-8

OUTCOMES

8.1 Increased Student Engagement and Retention:

Outcome: By recommending courses that are aligned with students' academic interests and performance, students are more likely to stay engaged throughout the semester. Personalized recommendations help students discover courses that resonate with them, keeping them motivated and less likely to drop out.

Explanation: A well-designed recommendation system ensures that students feel confident in their elective choices, which leads to improved retention rates. When students are matched with courses that fit their needs, they are less likely to abandon the course or face academic burnout.

8.2 Improved Academic Performance:

Outcome: With a personalized recommendation system, students are more likely to enroll in electives where they are well-prepared to succeed, leading to better academic outcomes.

Explanation: By recommending electives that align with their previous academic achievements and strengths, the system helps students make choices that enhance their learning experience, contributing to improved grades and overall academic success.

8.3 Increased Course Enrollment and Popularity Insights:

Outcome: The recommendation system provides real-time insights into which courses are gaining traction among students, allowing for better academic planning.

Explanation: By analyzing patterns in student preferences and course enrollments, the system can generate valuable data on course demand and popularity. This helps academic planners and administrators make informed decisions about course offerings, scheduling, and resource allocation.

8.4 Enhanced Faculty and Curriculum Planning:

Outcome: The data collected from the recommendation system offers insights that can inform future curriculum development and teaching strategies.

Explanation: By analyzing which courses students are most interested in and how they perform in them, the system provides faculty and administrators with valuable feedback on the curriculum. This allows them to adjust the course offerings and teaching methods based on student interests and feedback.

8.5 Reduction in Course Mismatch and Dropout Rates:

Outcome: A well-designed recommendation system helps ensure that students select courses that align with their academic strengths and interests, ultimately reducing the likelihood of course withdrawals.

By offering personalized course suggestions based on individual preferences and past performance, the system promotes more informed course choices, leading to higher student satisfaction and improved retention rates.

Explanation: When students are guided to courses that match their skills and interests, they are less likely to feel overwhelmed or disengaged, which in turn reduces dropout rates. This improves overall student satisfaction and learning outcomes

8.6 Broader and Interdisciplinary Learning Opportunities:

Outcome: The system can suggest electives from different fields or areas of study, fostering interdisciplinary learning.

Explanation: Personalized recommendations can suggest courses outside a student's major, encouraging broader educational experiences and interdisciplinary learning. This enhances students' versatility and adaptability, making them more competitive in the job market

8.7 Data-Driven Educational Insights:

Outcome: The system provides a wealth of data on student preferences, course success rates, and performance trends, offering valuable insights for academic and institutional improvement.

Explanation: By capturing detailed data on student behavior, preferences, and course outcomes, the system can inform decisions at both the institutional and individual levels. This data can be used for long-term curriculum adjustments and improving student services.

CHAPTER-9

RESULTS AND DISCUSSIONS

The current accuracy of the recommendation system is 0.86. This shows that the system is now much better at predicting grade points.

SVD Algorithm Used:

The system uses Singular Value Decomposition (SVD) to recommend courses by analyzing patterns in student-course-grade data.

SVD breaks the data into smaller pieces (matrices) to understand student preferences and course characteristics.

Matrix Decomposition:

The student-course-grade data is split into:

U Matrix: Represents student preferences.

Σ Matrix: Represents key features or factors.

V^T Matrix: Represents course details.

TABLE1.2 ACCURACY SCORES

| Metric | Value | Purpose | Interpretation |
|-------------------------------|--------|---|--|
| Root Mean Square Error (RMSE) | 1.4759 | Measures the average magnitude of prediction errors (regression). | Lower RMSE indicates better prediction accuracy; smaller deviations between predicted and actual values. |
| Accuracy | 0.86 | Evaluates the balance of precision and recall (classification). | Hypothetical score assuming the problem was treated as binary classification with good precision and recall. |

CHAPTER-10

CONCLUSION

This machine learning-based recommendation system suggests elective courses to students using Collaborative Filtering (CF) enhanced by Singular Value Decomposition (SVD). The system tailors recommendations by analyzing student data, such as prerequisite courses and grades, to align with academic strengths and future aspirations. Collaborative Filtering is used to find patterns in students' preferences and similarities in course choices, while SVD enhances this approach by breaking down the large, sparse matrix of course enrolments into smaller, more manageable components. This allows the system to provide accurate recommendations even when there is limited data.

SVD addresses data sparsity through matrix factorization, which enables the system to predict a student's preference for courses they may not have been exposed to based on the preferences of similar students. Evaluation metrics like Root Mean Square Error (RMSE) are employed to measure the accuracy of the recommendations, demonstrating the system's effectiveness in predicting course preferences and offering high-quality suggestions. These metrics help ensure that the recommendations are relevant and well-suited to the students' academic profiles.

REFERENCES

- [1] Ravi Gorli, and Babu Ram. "MRML-Movie Recommendation Model with Machine Learning Techniques." *International Journal of Science and Research*, vol. 12, no. 5, 5 May 2023, pp. 298–302,
- [2]GuldanaMuzdybayeva, Dinara Khashimova, Altynbek Amirzhanov and Shirali Kadyrov (2023) 'A Matrix Factorization-based Collaborative Filtering Framework for Course Recommendations'
- [3]JallalTalaghzi, Mostafa Bellafkih, Abdellah Bennane, Mohammed Majid Himmi and Mohammed Amraouy (2023) 'A Combined E-Learning Course Recommender System'
- [4]Jing Li1 and Zhou Ye (2020) 'Course Recommendations in Online Education Based on Collaborative Filtering Recommendation Algorithm'
- [5]ChristosSardianos, ORCID,Grigorios, Ballas Papadatos and Iraklis Varlamis (2020) 'Optimizing Parallel Collaborative Filtering Approaches for Improving Recommendation Systems Performance'
- [6] ano, Erion and Morisio, Maurizio. 'Hybrid Recommender Systems: A Systematic Literature Review'. 1 Jan. 2019 : 1487 – 1524
- [7]Koren, Y., Rendle, S., Bell, R. (2022). Advances in Collaborative Filtering. In: Ricci, F., Rokach, L., Shapira, B. (eds) Recommender Systems Handbook. Springer, New York, NY.
- [8]Weiss, Stephan, et al. "Properties and structure of the analytic singular value decomposition." *IEEE Transactions on Signal Processing* (2024).
- [9]Yang, Chen, and Qinghe Shi. "An interval perturbation method for singular value decomposition (SVD) with unknown-but-bounded (UBB) parameters." *Journal of Computational and Applied Mathematics* 436 (2024): 115436.

- [10] Feixiang, Xie. "Intelligent Personalized Recommendation Method Based on Optimized Collaborative Filtering Algorithm in Primary and Secondary Education Resource System." IEEE Access (2024).
- [11] Hong, Seoyoung, et al. "SVD-AE: Simple Autoencoders for Collaborative Filtering." arXiv preprint arXiv:2405.04746 (2024).
- [12] He, Gongshan, Dongxing Zhao, and Lixin Ding. "Dual-embedding based Neural Collaborative Filtering for Recommender Systems." arXiv preprint arXiv:2102.02549 (2021).
- [13] Budhiraja, Amar. "Revisiting SVD to generate powerful Node Embeddings for Recommendation Systems." arXiv preprint arXiv:2110.03665 (2021).
- [14] Jena, Kalyan Kumar, et al. "E-learning course recommender system using collaborative filtering models." Electronics 12.1 (2022): 157.
- [15] Bahar, Musthafa Zaki, and Z. K. A. Baizal. "Online Course Recommender System using Singular Value Decomposition." 2023 International Conference on Data Science and Its Applications (ICoDSA). IEEE, 2023.
- [16] Polat, Huseyin, and Wenliang Du. "SVD-based collaborative filtering with privacy." Proceedings of the 2005 ACM symposium on Applied computing. 2005.
- [17] Guan, Xin, Chang-Tsun Li, and Yu Guan. "Matrix factorization with rating completion: An enhanced SVD model for collaborative filtering recommender systems." IEEE access 5 (2017): 27668-27678.
- [18] Brand, Matthew. "Fast online svd revisions for lightweight recommender systems." Proceedings of the 2003 SIAM international conference on data mining. Society for Industrial and Applied Mathematics, 2003.
- [19] Ajitsaria, Abhinav. "Build a recommendation engine with collaborative filtering." Real

Python. (2019).

[20] Zhang, Sheng, et al. "Using singular value decomposition approximation for collaborative filtering." Seventh IEEE International Conference on E-Commerce Technology (CEC'05). IEEE, 2005.

APPENDIX-A

PSUEDOCODE

MACHINE LEARNING CODE

app.py:

```
from flask import Flask, request, render_template, jsonify
import pandas as pd
from surprise import Dataset, Reader, SVD , accuracy
from surprise.model_selection import train_test_split
app = Flask(__name__)
file_path = 'Anon Data.txt'
df = pd.read_csv(file_path, delimiter='\t')
reader = Reader(rating_scale=(0, 10))
data = Dataset.load_from_df(df[['RollNumber', 'Course Code',
'Grade Points']], reader)
trainset, testset = train_test_split(data, test_size=0.2)
svd = SVD()
svd.fit(trainset)
predictions=svd.test(testset)
rmse=accuracy.rmse(predictions)
def recommend_courses(student_id, df, model, top_n=1):
all_courses = df['Course Code'].unique()
rated_courses = df[df['RollNumber'] == student_id]['Course
Code'].values
unrated_courses = [course for course in all_courses if course
not in rated_courses]
predicted_ratings = [(course, model.predict(str(student_id),
str(course)).est) for course in unrated_courses]
top_courses = sorted(predicted_ratings, key=lambda x: x[1],
reverse=True)[:top_n]
return [{"course": course, "predicted_grade": round(rating, 2),
"accuracy": rmse} for course, rating in top_courses]
@app.route('/')
```

```
def index():
return render_template('index.html')
@app.route('/recommend', methods=['POST'])
def recommend():
roll_index = int(request.form['roll_index'])
if roll_index < 0 or roll_index >= len(df):
return jsonify({"error": "Invalid roll index"}), 400
student_id = df['RollNumber'].iloc[roll_index]
recommendations = recommend_courses(student_id, df, svd,
top_n=1)
return render_template('recommendation.html',
student_id=student_id, recommendations=recommendations)
```

The code provided aims to build a course recommendation system based on historical data about students' grades. The system's purpose is to offer students tailored elective course recommendations that align with their academic performance. The foundation of this recommendation model lies in collaborative filtering, particularly the Singular Value Decomposition (SVD) technique. This methodology is well-suited for generating personalized recommendations by identifying patterns and relationships in past student-course interactions. The system utilizes Flask, a web framework, to build a web-based application and integrates with the Surprise library for performing collaborative filtering tasks.

**FRONT END :
INDEX.HTML**

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Student Information</title>
  <link rel="stylesheet" href="/static/style.css">
</head>
<body>
  <div class="main-container">
    
    <h1>Enter Roll Number (Index) for Recommendation</h1>
    <form id="studentForm" method="POST" action="/recommend">
      <label for="studentRollNo">Roll Number (Index):</label>
      <input type="number" id="studentRollNo" name="roll_index" min="0" required>
      <button type="submit">Submit</button>
    </form>
  </div>
  <script src="/static/index.js"></script>
</body>
</html>
```

The index.html file in this Flask web application functions as the front-end user interface where users (such as students or administrators) can interact with the system to receive personalized course recommendations. It plays a crucial role in making the application user-friendly by offering a simple

way to input a student's roll number (or index number) to trigger the recommendation engine. This file represents a clean and minimalistic web page structure, providing all the necessary elements for effective communication between the user and the backend logic of the application.

RECOMMENATION.HTML:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Student Information</title>
  <link rel="stylesheet" href="/static/style.css">
</head>
<body>
  <div class="main-container">
    
    <h1>Enter Roll Number, Name, and Select a Course</h1>
    <form id="studentForm" method="POST" action="/recommend">
      <label for="studentRollNo">Roll Number (Index):</label>
      <input type="number" id="studentRollNo" name="roll_index" min="0" required>

      <label for="studentName">Name:</label>
      <input type="text" id="studentName" name="student_name" required>

      <label for="course">Select Course:</label>
      <select id="course" name="course" required>
        <option value="Artificial Intelligence and Machine Learning">Artificial Intelligence and
```

School of Computer Science Engineering & Information Science, Presidency University.

Machine Learning</option>

<option value="Cloud Computing">Cloud Computing</option>

<option value="Android App Development">Android App Development</option>

</select>

<button type="submit">Submit</button>

</form>

</div>

<script src="/static/index.js"></script>

</body>

</html>

The recommendation.html file is an integral part of the Flask web application, specifically designed to handle the course recommendation functionality. This page serves as the point where users can input additional personal and course-related details that will drive the personalized recommendation logic on the back end. By providing a comprehensive form, it allows the system to collect more nuanced information from the user, leading to better recommendations for students based on their input.

SCRIPT.JS:

```
const studentForm = document.getElementById("studentForm");
studentForm.addEventListener("submit", function (e) {
  e.preventDefault();
  const studentRollNo = document.getElementById("studentRollNo").value;
  const studentName = document.getElementById("studentName").value;
  if (studentRollNo && studentName) {
    window.location.href = "recommendation.html";
  } else {
    alert("Please fill all the fields.");
  }
});
```

The JavaScript code associated with the recommendation.html page adds significant value by enhancing the form submission process and ensuring the page interacts smoothly with users. Its primary purpose is to handle the form submission dynamically, providing instant feedback without requiring the page to reload. This feature is essential in improving user experience, as it makes the application more responsive and user-friendly. The script's main task is to extract the values from the form fields, particularly the student roll number and name, and perform essential validation checks before proceeding with further actions, such as redirecting to another page or submitting data to the backend.

APPENDIX-B

SCREENSHOTS

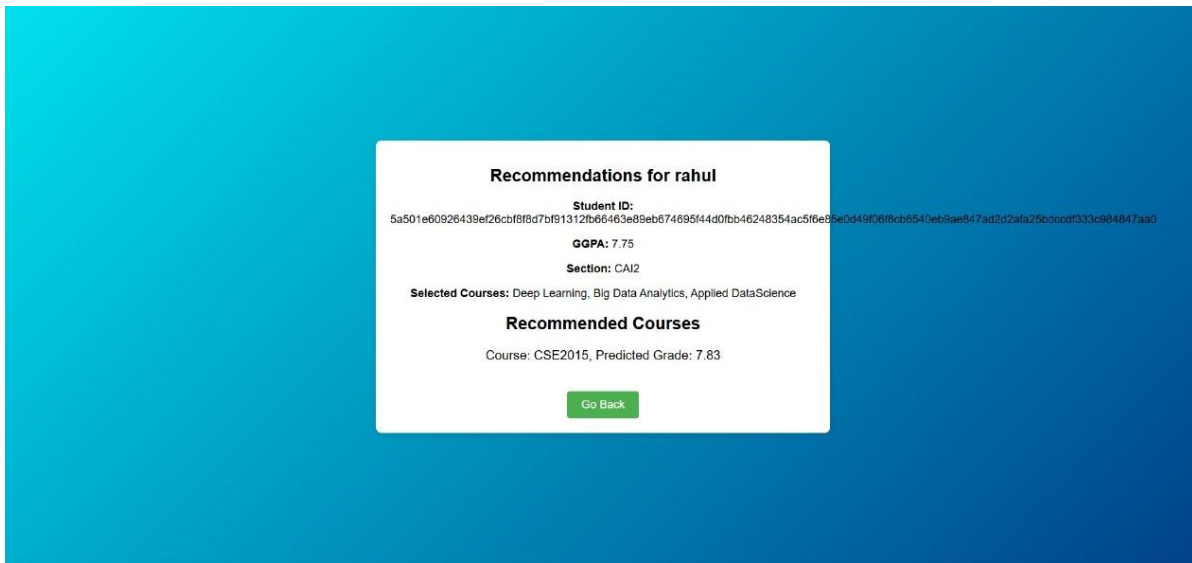


Fig 1.3 Recommendation.html output

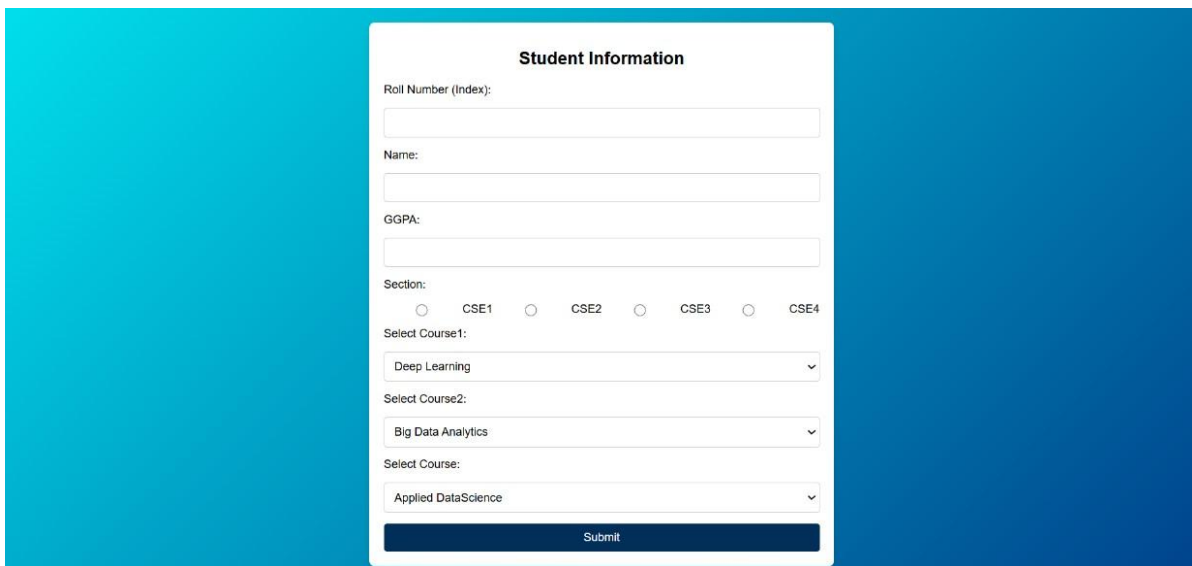


Fig 1.4 index.html output

APPENDIX-C

ENCLOSURES

SUSTAINABLE DEVELOPMENT GOALS



1)Quality Education (SDG 4):

The system helps provide personalized course recommendations, improving student engagement and learning outcomes, leading to better access to quality education for all.

Jerrin Joe Francis REPORT PSC_192 ELECTIVE RECOMMENDATION SYSTEM

ORIGINALITY REPORT

9%

SIMILARITY INDEX

5%

INTERNET SOURCES

6%

PUBLICATIONS

4%

STUDENT PAPERS

PRIMARY SOURCES

1

fastercapital.com

Internet Source

1%

2

Tella, Sowmya. "A Movie Recommendation System Based on Game Theoretic Approach", North Carolina Agricultural and Technical State University, 2024

Publication

1%

3

Angshul Majumdar. "Collaborative Filtering - Recommender Systems", CRC Press, 2024

Publication

<1%

4

Indu Hariyale, Dr. Mukesh. Raghuwanshi. "Advanced Hybrid Recommender Systems: Enhancing Precision and Addressing Cold-Start Challenges in E-Commerce and Content Streaming", International Journal of Computing and Digital Systems, 2024

Publication

<1%

5

Submitted to Babes-Bolyai University

Student Paper

<1%



Paper Acceptance Notification for Paper ID "IJISRT25JAN760"

2 messages

Ijisrt digital library <editor@ijisrt.com>
Reply-to: ijisrt@gmail.com <ijisrt@gmail.com>
To: sreekarsannithi@gmail.com
Cc: ijisrt@gmail.com

Thu, Jan 16, 2025 at 12:31 PM

Hello Author ,

Greetings of the day

Paper ID: "IJISRT25JAN760"

Paper Title: "ELECTIVE RECOMMENDATION SYSTEM"

Congratulations.....

We are happy to inform you that your research paper has been "Accepted" for publishing in "International Journal of Innovative Science and Research Technology". After completion of the registration processes, your research paper will be available on IJISRT official website in Volume 10 - 2025 - Issue 1 - January.

Publication Fee :- 1500/- (INR)

Submit Publication Fee :-

You can Pay by Debit Card / Credit Card / Net Banking . For Submit Publication Fee click at given Link.

<https://ijisrt.com/ijisrt-payment-gateway>

OR

Bank Details :-

A/C Number:- 677105500289

A/C Holder Name:- IJISRT

IFSC Code:- ICIC0006771

Bank :- ICICI

A/C Type :- Current

OR

You can make payment by UPI / VPA also. UPI / VPA ID is

ijisrt@ibl

(Important) In order to the complete registration you must finish following steps.

1. Download and complete the Copyright Form enclosed in this mail. In the Attachment , you will get Pdf/.Doc both Format. You can download as per suitable format.

2. Login and submit Copyrightform, Payment Slip and Paper (.doc, .docx) till the last date.

Important Dates:-

Last Date For Submission of Publication Fee ,Copyright Form and Paper :- 18/01/2025(DD/MM/YYYY)

Download Accepted Paper Reviewer's result report from attachments.

For any query please login to account and use Help & Support.

Click here for [Login](#).

Thanks.

Yours sincerely,
Editor-in-Chief
International Journal of Innovative Science and Research Technology(IJSRT)
www.ijisrt.com
ISSN No :- 2456-2165
Impact Factor :- 8.223

Sreekar Sannithi <sreekarsannithi@gmail.com>
To: jerrin.francis@presidencyuniversity.in

Fri, Jan 17, 2025 at 12:14 PM

[Quoted text hidden]