UNIVERSITY OF SCIENCE AND TECHNOLOGY OF HANOI
VIETNAMESE FRANCE UNIVERSITY

Topic: MovieLens Recommender: A Collaborative Filtering and SVD Matrix Factorization Approach in KDD

Group Members

Tran Viet Cuong	BA12-035
Doan Duc Hoang	BA12-078
Nguyen Vu Gia Huy	BA12-088
Duong Dam Lam	BA12-102
Trinh Quang Minh	BA12-127
Nguyen Quoc Viet	BA12-192

Information & Communication Technology **(ICT)**

Lecturer

Dr. Doan Nhat Quang

ICT Lab

March, 2025

Abstract

This report presents a movie recommender system developed using the MovieLens dataset[1], integrating Collaborative Filtering (CF) and Singular Value Decomposition (SVD) within the Knowledge Discovery in Databases (KDD) framework. The study follows a rigorous methodology that encompasses data selection, preprocessing, transformation, and model building. CF is employed to compute user-item similarities, while SVD is applied to extract latent factors. The system's performance is evaluated using standard metrics, including RMSE and MAE, with the experimental results reported in detail below to objectively demonstrate its effectiveness. Additionally, the report discusses key considerations regarding data handling, model optimization, and deployment strategies, and outlines directions for future improvements. Overall, this work provides a reproducible and scientifically sound approach to building a scalable movie recommender system.

TABLE OF CONTENTS

I. **INTRODUCTION** 4
1. Context and Motivation 4
2. Thesis Structure 5
II. **OBJECTIVES** 6
1. Desired Features 6
2. Expected Outcomes 6
III. **PROJECT ANALYSIS** 7
1. Overall System Requirement 7
2. Data Requirements 7
3. Functional Requirements 7
4. Non-Functional Requirements 8
5. Diagrams and Visualizations 8
6. Statistical and Visual Data Analysis 9
7. Additional Requirements 9
8. Figure Analysis 10
IV. **METHODOLOGY** 20
1. Tools and Techniques 20
2. Data Preprocessing 21
3. Collaborative Filtering (CF) 21
4. Singular Value Decomposition (SVD) 23
5. API 25
V. **RESULT AND DISCUSSION** 27
1. Result 27
System Performance 27
Sample Recommendations 27
2. Discussion 27
Strengths of the System 27
Limitations & Challenges 28
VI. **CONCLUSION AND FUTURE WORKS** 29
VII. **References** 30

List of Figures

1 System Architecture Diagram of the KDD Pipeline for the MovieLens Recommender System 10
2 Use Case Diagram for the Recommender System (simulation) 11
3 Data Flow Diagram from Raw Data to Recommendations 12
4 ER Diagram of the Integrated MovieLens Dataset 13
5 Horizontal Bar Chart for Users Age Distribution 14
6 Box Plots Showing Rating Variability (Left) and Age by Gender (Right) 15
7 Gender Distribution 16
8 Occupation Distribution 17
9 Rating Distribution 18
10 Number of Movies per Genre 19

List of Tables

1 Use Case Descriptions for the Recommender System (Simulation) 11

2 Data Flow Analysis from Raw Data to Recommendations 12

3 Entities and Relationships in the Integrated MovieLens Dataset 13

4 Data Preprocessing Steps and Tools 21

5 Item-based Collaborative Filtering Process with Additional Filtering Criteria and Evaluation Metrics 23

6 Singular Value Decomposition Process and Integration with Collaborative Filtering 24

7 System Architecture: Integration of MovieLens Data, CF, SVD, and API Interface 26

8 Comparison of RMSE and MAE for CF with and without SVD 27

9 Sample Recommendations Based on Collaborative Filtering 27

I. INTRODUCTION

1. Context and Motivation

The rapid expansion of digital content has increased the demand for personalized recommendation systems across various domains such as e-commerce, online streaming, and social networks. In this context, recommender systems have become essential tools for enhancing user engagement and satisfaction by tailoring content to individual preferences.[2] Our project focuses on the development of a movie recommendation system using the MovieLens dataset, a widely recognized benchmark in this research area.[3]

The choice of the MovieLens dataset is driven by its comprehensive coverage of user ratings and rich movie metadata, making it an ideal resource for experimenting with advanced machine learning and data mining techniques. The MovieLens dataset presents challenges such as diverse user preferences and a large number of movies, which makes it an excellent testing ground for various recommendation algorithms. By integrating Collaborative Filtering (CF)[4] with Singular Value Decomposition (SVD) matrix factorization[5], our approach aims to uncover latent factors that capture the intrinsic characteristics of both users and movies, addressing the challenges of sparse data and large-scale recommendation.

Furthermore, the project follows the Knowledge Discovery in Databases (KDD) framework [6], which includes systematic steps of data selection, preprocessing, transformation, model building, and evaluation. This structured approach ensures the reproducibility and transparency of our experiments and provides a clear roadmap for addressing key issues such as data handling, model optimization, and deployment strategies in real-world scenarios.

In summary, the motivation behind this project lies in the pursuit of a scalable and data-driven solution to improve the accuracy of movie recommendations while adhering to rigorous scientific standards. This work aims to contribute to the field of recommender systems by demonstrating an effective integration of traditional collaborative filtering methods and matrix factorization techniques, all within a well-defined KDD process.[7]

2. Thesis Structure

This thesis provides a comprehensive, reproducible, and scientifically rigorous exploration of the development and evaluation of a movie recommendation system based on the MovieLens dataset. The structure of the report is as follows:

1. **Introduction:** Provides the background, motivation, and scope of the project, outlining the importance of personalized recommendation systems, the challenges of large-scale data processing, and the integration of Collaborative Filtering (CF) and Singular Value Decomposition (SVD) techniques.

2. **Objectives:** Defines the primary goals of the project, including the development of a robust movie recommender system using CF and SVD, and evaluating its performance using standard metrics such as RMSE and MAE.

3. **Project Analysis:** Presents the system requirements, including a detailed review of data acquisition from the MovieLens dataset, preprocessing techniques, and feature engineering methods. It also covers functional and non-functional requirements, including system scalability, reliability, and maintainability.

4. **Methodology:** Describes the tools, techniques, and algorithms used throughout the project. This section includes the data preprocessing pipeline, the implementation of CF and SVD models, and the optimization techniques used to enhance model performance.

5. **Results and Discussion:** Summarizes the experimental results, presenting the evaluation metrics such as RMSE and MAE. It also includes a discussion of the effectiveness of the models and a comparative analysis of different configurations of CF and SVD.

6. **Conclusion and Future Work:** Concludes the study by summarizing the main findings and contributions. This section also discusses the limitations of the current system and outlines potential directions for future research, including hybrid recommendation methods and the use of deep learning techniques.

This structure ensures a systematic and clear documentation of the entire project, from the initial design and data preparation stages to the final model implementation and evaluation. The goal is to provide a detailed, reproducible framework that supports both future research and practical applications in the field of recommender systems.

II. OBJECTIVES

1. Desired Features

The primary objective of this project is to develop a robust and scalable movie recommender system using the MovieLens 1M dataset. In order to achieve this, the system is designed with the following desired features:

- **Accurate Rating Prediction:** Employ both item-based Collaborative Filtering and Singular Value Decomposition (SVD) to predict user ratings, with a focus on achieving a balance between predictive accuracy and generalizability.
- **Scalability and Efficiency:** Ensure that the system remains computationally efficient, especially when scaling to large datasets with sparse data, and carefully monitor trade-offs between predictive accuracy and computational complexity.
- **Interpretability and Reproducibility:** Maintain rigorous documentation and provide reproducible code to facilitate validation, further research, and comparison between different models such as item-based CF and CF with SVD.

2. Expected Outcomes

Upon completion, the project is expected to deliver the following outcomes:

- **Validated Recommendation Model:** A fully functional recommender system that demonstrates predictive accuracy, as evidenced by standard evaluation metrics such as the Root Mean Squared Error (RMSE) and the Mean Absolute Error (MAE), with a careful balance between accuracy and generalization.
- **Comprehensive Experimental Analysis:** Detailed comparative analyses that objectively illustrate the performance of the item-based Collaborative Filtering and SVD approaches, highlighting both their strengths and limitations.
- **Optimized Model Configuration:** A set of finely tuned hyperparameters and an optimized model architecture, balancing predictive accuracy with computational efficiency, especially considering the differences between CF with and without SVD.
- **Deployment-Ready Prototype:** A scalable deployment framework, including an interface or API, capable of generating top-k movie recommendations in real time.
- **Academic Contribution:** A well-documented report that adheres to rigorous scientific standards, offering insights into the application of machine learning and data mining techniques in the development of recommender systems, including the challenges and trade-offs between item-based CF and CF with SVD.

III. PROJECT ANALYSIS

1. Overall System Requirement

The system is designed to build a scalable and reproducible movie recommender that leverages the MovieLens 1M dataset. It should integrate both item-based Collaborative Filtering and SVD Matrix Factorization techniques within a complete KDD (Knowledge Discovery in Databases) pipeline. The overall system must support the following:

- Automated data ingestion and preprocessing of raw data files.
- Robust data transformation and feature engineering to construct a reliable user-item matrix.
- Implementation of multiple recommendation algorithms with hyperparameter tuning.
- Comprehensive evaluation using standard metrics (e.g., RMSE, MAE) and comparative analysis.
- Deployment of a user-friendly interface or API for real-time movie recommendations.

2. Data Requirements

The project relies on the three key data files from the MovieLens 1M dataset:

- **user.dat:** Contains user attributes such as `UserID`, gender, age, occupation, and location.
- **rating.dat:** Includes user ratings with attributes like `UserID`, `MovieID`, rating value, and timestamp.
- **movie.dat:** Provides movie metadata, including `MovieID`, title, genres, and optionally the year of release.

All data must be cleaned, integrated, and transformed to facilitate further analysis and model training.

3. Functional Requirements

The system is expected to provide the following functionalities:

- **Data Collection and Preprocessing:** Automate the import of the MovieLens files, detect and handle missing or inconsistent values, and merge the data into a unified format.

- **Exploratory Data Analysis (EDA):** Generate statistical summaries and visualizations (e.g., histograms, bar charts) to understand rating distributions, user demographics, and genre frequencies.

- **Model Building:** Develop item-based Collaborative Filtering and SVD-based models to predict user ratings and generate top-k movie recommendations.

- **Evaluation:** Implement evaluation protocols using metrics such as RMSE, MAE, and ranking metrics (e.g., Precision@K, Recall@K) to objectively assess model performance.

- **Deployment:** Provide a demonstrable interface or API that accepts user input (e.g., user ID) and outputs personalized movie recommendations.[10]

4. Non-Functional Requirements

To ensure the practical viability of the system, the following non-functional requirements must be met:

- **Scalability:** The system should efficiently process large datasets and support real-time recommendation generation.

- **Reliability:** Consistent model performance and minimal system downtime are critical.

- **Usability:** The end-user interface should be intuitive, and the system should provide clear feedback and documentation.

- **Maintainability:** The codebase and documentation should be well-organized to facilitate future modifications and extensions.

5. Diagrams and Visualizations

For a thorough Requirement Analysis, the following diagrams and visualizations are essential:

- **System Architecture Diagram:**

  Figure 1: Illustrates the complete KDD pipeline including data selection, preprocessing, transformation, model building, evaluation, and deployment.

- **Use Case Diagram Simulation:**

  Figure 2: Depicts the primary interactions of the simulation between end-users and the system, such as user login, request for recommendations, and feedback collection.

- **Data Flow Diagram (DFD):**

  Figure 3: Shows the flow of data from raw input files to the final movie recommendations.

- **ER Diagram or Data Schema:**

  Figure 4: Provides a clear schema of the integrated dataset, detailing the relationships among users, movies, and ratings.

6. Statistical and Visual Data Analysis

To support the requirement analysis, include the following charts and graphs:

- **Bar Charts:**

  Figure 5: Distribution of ratings, user demographics (age, gender, occupation), and frequency counts of movie genres.

- **Box Plots:**

  Figure 6: Variability of ratings between different user groups or movie genres. Difference in age between sexes.

- **Vertical Bar Charts:**

  Figure 7: Distribution of users by gender (Male/Female).

  Figure 8: Frequency of users across occupational categories.

  Figure 9: Total count of ratings given (1-star to 5-star).

  Figure 10: Quantity of movies categorized by genre.

7. Additional Requirements

- **Documentation:** Detailed and clear documentation must accompany every stage of the project, including data processing, model development, and evaluation.

- **Reproducibility:** All experimental procedures, including data splits, model parameters, and evaluation metrics, should be documented to ensure reproducibility.

- **Evaluation Metrics:** Explicitly list and define the metrics (e.g., RMSE, MAE, Precision@K, Recall@K) used for model performance assessment.

8. Figure Analysis

Overview 1: Figure 1 shows the KDD pipeline for the MovieLens Recommender System, detailing sequential stages: preprocessing, transformation, model building, evaluation, and deployment.

Preprocessing: Raw *Movies Data* is cleansed by addressing missing values, outliers, and inconsistencies to ensure quality.

Transformation & EDA: The refined data is structured and examined to identify patterns and select pertinent features, as indicated by the *EDA* datasets.

Model Building: A recommendation model is constructed using Collaborative Filtering (CF) and Singular Value Decomposition (SVD) to optimize user-item interactions.

Evaluation: The model's performance is verified using RMSE and precision-recall metrics.

Deployment: The final model is deployed via an API for real-time integration.

Summary: The diagram succinctly captures the development flow, underscoring the importance of preprocessing and EDA in enhancing model performance.

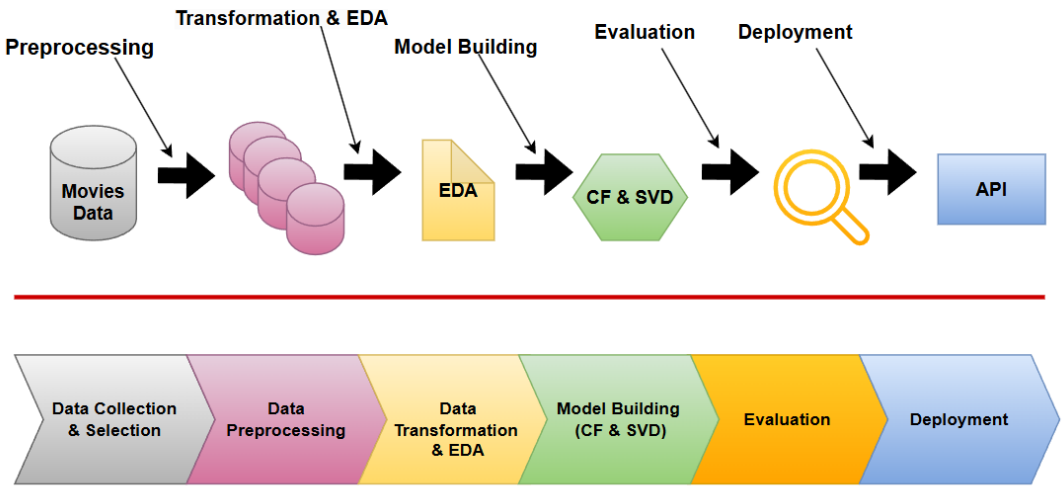


Figure 1: System Architecture Diagram of the KDD Pipeline for the MovieLens Recommender System

- **Overview 2:** Figure 2 shows how a single actor (*User*) interacts with core system functionalities. The diagram adheres to UML standards, highlighting authentication needs and optional feedback.

- **Actor:** The *User* can manage a profile, request/view recommendations, and provide feedback. Each action either requires or extends another, reflecting realistic user flows.

- **Key Relationships:**

  – ***include***: "Manage Profile," "Request Recommendation," and "View Recommendation" each include "Authenticate User," ensuring secure access.

  – ***extend***: "Provide Feedback" extends "View Recommendation," making feedback an optional, post-recommendation step.

Use Case	Description
Authenticate User	Allows only verified individuals to access personalized features. Serves as a prerequisite for other actions.
Manage Profile	Lets users edit personal details. Includes "Authenticate User" to secure profile updates.
Request Recommendation	Fetches personalized movie suggestions. Relies on "Authenticate User" to retrieve the correct profile.
View Recommendation	Displays the recommended items. Requires authentication to ensure user-specific results.
Provide Feedback	Extends "View Recommendation" for optional user feedback, improving future recommendations.

Table 1: Use Case Descriptions for the Recommender System (Simulation)

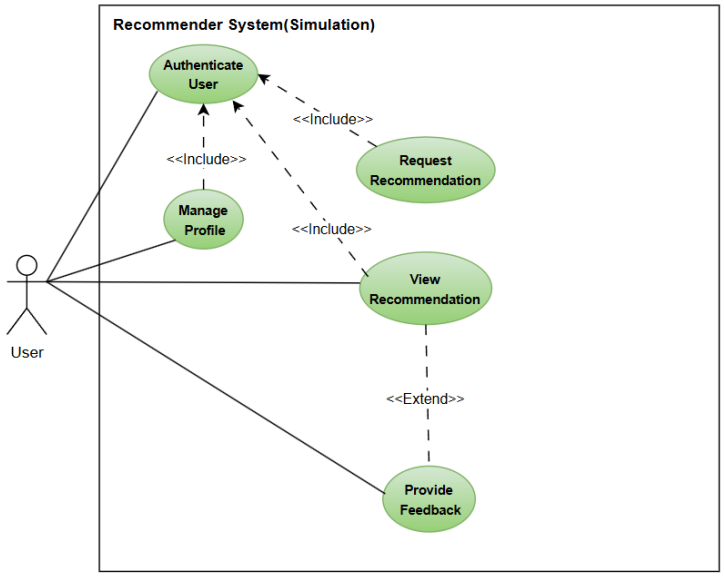


Figure 2: Use Case Diagram for the Recommender System (simulation)

- **Overview 3:** Figure 3 shows how raw MovieLens CSV files transition through cleaning and transformation into a structured dataset, which is then used to train a recommendation model. Finally, the system outputs top-N movie recommendations tailored to each user.

- **Importance:** This linear flow ensures data consistency and highlights key processing stages—such as merging, filtering, and matrix construction—prior to model training and recommendation generation.

Step	Description
Raw Data (MovieLens CSVs)	Initial dataset containing user ratings, movie metadata, and optional tags. This unprocessed form requires cleaning and integration.
Data Cleaning & Transformation	Removal of missing values and outliers, merging multiple CSVs, and creating a unified structure. A user-item matrix is formed, ready for further analysis.
Recommendation Model	Application of Collaborative Filtering (CF) and/or Hybrid approaches. The model is trained to predict user-specific movie preferences.
Top-N Movie Recommendations	Final output providing ranked movie suggestions, enabling users to discover relevant content.

Table 2: Data Flow Analysis from Raw Data to Recommendations

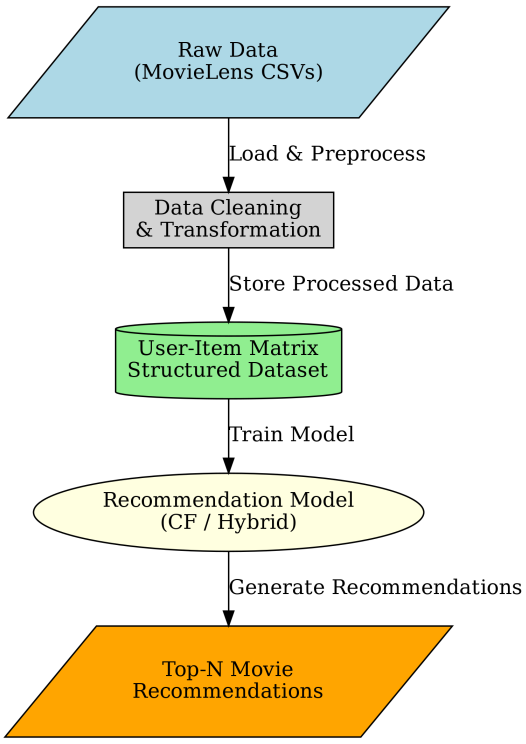


Figure 3: Data Flow Diagram from Raw Data to Recommendations

- **Overview 4:** Figure 4 depicts how the four primary entities—*Users*, *Ratings*, *Movies*, and *Tags*—are linked. Each user can rate multiple movies and add multiple tags, while each movie can receive multiple ratings and tags.

- **Key Relationships:**

  – **Users ↔ Ratings:** A one-to-many (1–N) relationship, where each user may have multiple ratings, but each rating belongs to exactly one user.

  – **Movies ↔ Ratings:** Also 1–N, reflecting that each movie can be rated many times, whereas each rating entry corresponds to one specific movie.

  – **Users ↔ Tags:** A user can add multiple tags, but each tag is associated with exactly one user.

  – **Movies ↔ Tags:** Each movie can have many tags, and each tag references only one movie.

- **Significance:** By integrating these four tables, the dataset becomes comprehensive, allowing collaborative filtering (via *Ratings*) and potential content-based or hybrid approaches (via *Tags*). This structure underpins subsequent data preprocessing, feature engineering, and model building as outlined in the KDD process.

Entity / Relationship	**Description**
Users	Contains `userId`, `age`, `gender`, `occupation`. Each user can rate multiple movies and add multiple tags.
Ratings	Stores `userId`, `movieId`, `rating`, `timestamp`. Represents a 1–N link from Users (one user can have many ratings) and Movies (one movie can be rated many times).
Movies	Holds `movieId`, `title`, `genres`. Each movie can be associated with multiple ratings and tags, enabling collaborative filtering and content-based approaches.
Tags	Contains `userId`, `movieId`, `tag`, `timestamp`. Reflects a 1–N relationship with both Users and Movies, supporting further feature engineering or hybrid recommendation strategies.

Table 3: Entities and Relationships in the Integrated MovieLens Dataset

Figure 4: ER Diagram of the Integrated MovieLens Dataset

- **Overview:** Figure 5 shows the distribution of users across seven age groups (Under 18, 18–24, 25–34, 35–44, 45–49, 50–55, and 56+). By using `sns.barplot` with the `magma` palette, this chart visually highlights how different age brackets are represented within the MovieLens dataset.

- **Key Observations:**

  – **Dominant Age Range:** Users aged 25–34 constitute the largest segment, implying a substantial influence on overall rating and tagging patterns.

  – **Moderate Youth Representation:** Although smaller, the 18–24 group remains noteworthy, while the "Under 18" bracket appears minimal, suggesting fewer teenage participants or limited data for this subset.

  – **Older Demographics:** Brackets from 35–44 onward show moderate user counts, demonstrating the platform's broader appeal beyond younger audiences.

- **Relevance to Recommendations:** Recognizing which age groups predominate helps refine recommendation strategies. Systems may prioritize popular genres or tailored suggestions for the most active brackets while still accommodating smaller segments to maintain inclusive user satisfaction.

- **Technical Note:** The code snippet leverages `pandas` to group user ages into custom categories, and `seaborn` to generate a horizontal bar chart. By calling `value_counts()` and reindexing with a predefined list, the figure consistently displays each age bracket—even those with fewer users—providing a reliable overview of the dataset's demographic spread.

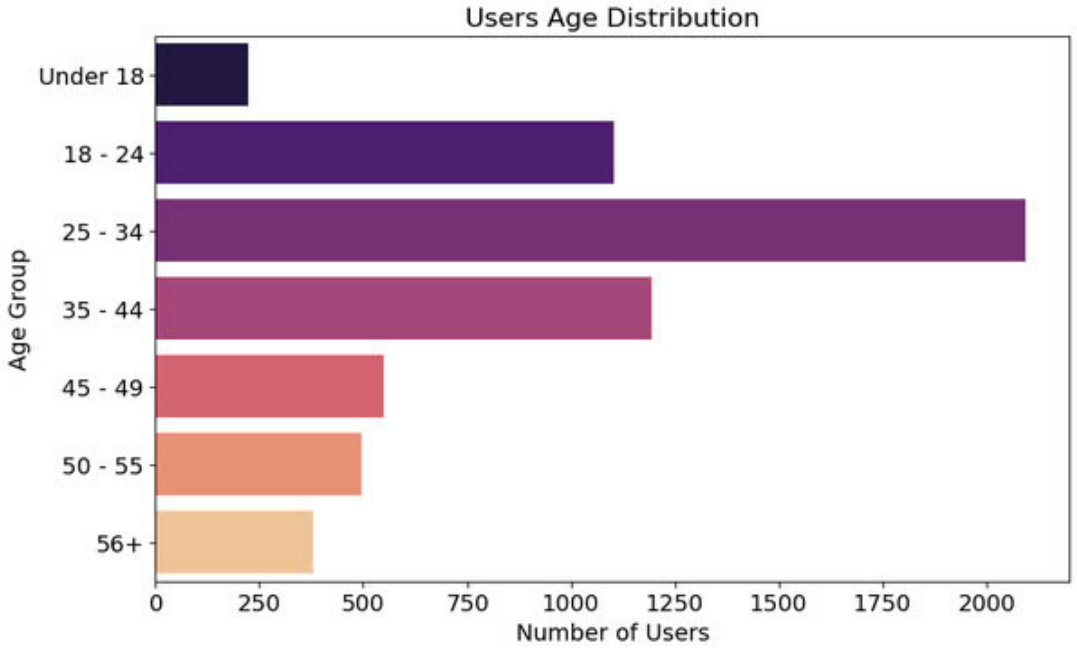


Figure 5: Horizontal Bar Chart for Users Age Distribution

- **Overview:** Figure 6 presents two distinct box plots side by side, highlighting different aspects of the MovieLens dataset. The left plot depicts rating variability among users, while the right plot illustrates the distribution of user age across genders.

- **Rating Variability (Left Plot):**

  – **Median and Range:** The box's central line corresponds to the median rating, surrounded by the interquartile range (IQR). Outliers, marked by points beyond the whiskers, signify unusual rating behaviors.

  – **Implications for Recommendations:** This distribution helps identify whether users tend to rate movies generously or conservatively. Systems can use such insights to normalize ratings and improve collaborative filtering accuracy.

- **Age by Gender (Right Plot):**

  – **Median Age Differences:** The box plot shows how the median age for each gender compares, reflecting potential demographic trends within the dataset.

  – **Outliers and Spread:** Points beyond the whiskers highlight atypical age entries or sparse data in certain age ranges. This information can guide demographic-specific recommendation strategies.

- **Significance:** Both box plots underscore the importance of understanding user variability—whether in ratings or demographic distributions. By capturing these nuances, the recommender system can be refined to better accommodate diverse user preferences and behaviors.

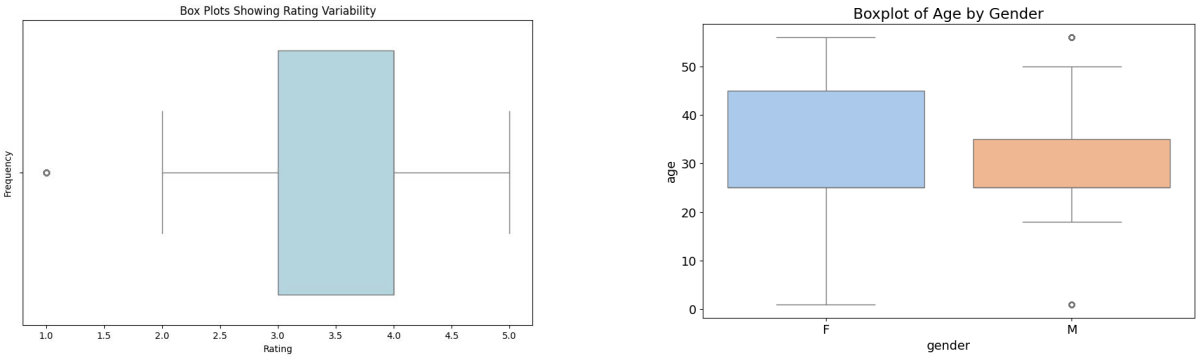


Figure 6: Box Plots Showing Rating Variability (Left) and Age by Gender (Right)

- **Overview:** Figure 7 illustrates the gender distribution within the MovieLens user dataset, showing two primary categories: male (M) and female (F). According to the code snippet, there are 4,331 male users and 1,709 female users, for a total of 6,040 entries.

- **Key Observations:**

  – **Dominant Gender Group:** Male users form the majority, accounting for approximately 71.7% of the dataset.

  – **Female Representation:** Although smaller (around 28.3%), the female segment remains significant enough to warrant tailored recommendation strategies.

- **Relevance to Recommender System:** A clear gender imbalance may influence rating patterns and genre preferences. By acknowledging this distribution, the system can explore gender-based personalization techniques, while ensuring minority segments (female users) are not overlooked.

- **Technical Note:** The code leverages `value_counts()` for frequency calculation, followed by a `hist()` or `bar` plot to visualize the results. This straightforward approach effectively highlights the numerical gap between male and female users, offering insights for demographic-specific analysis.

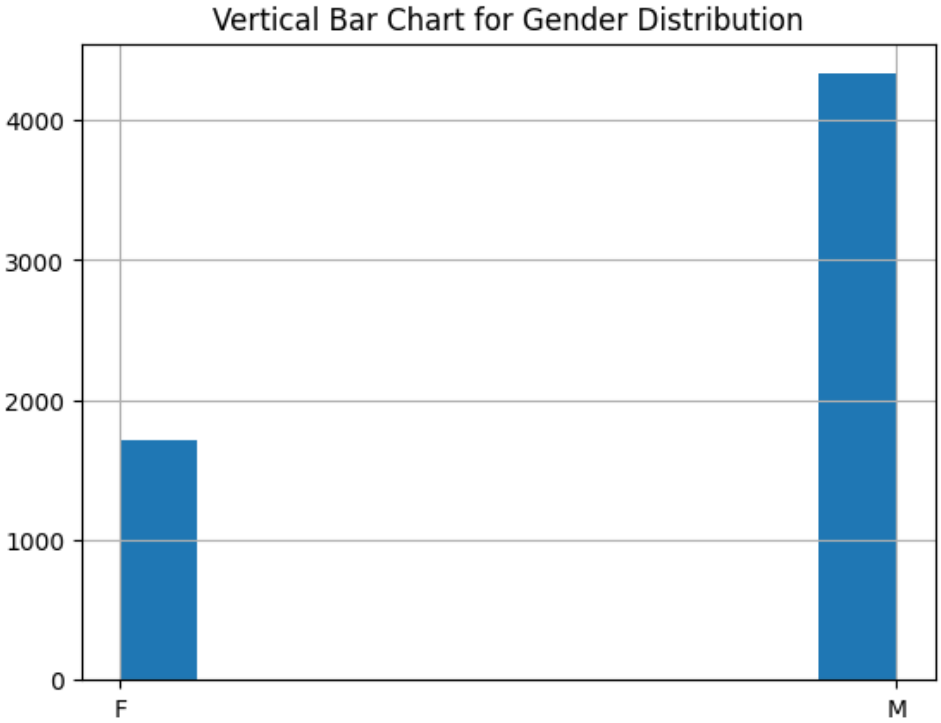


Figure 7: Gender Distribution

- **Overview:** Figure 8 depicts the distribution of users across 21 occupation categories, mapped from numeric IDs to descriptive labels (e.g., "college/grad student," "programmer," "farmer").

- **Key Observations:**

  – **Largest Group:** "College/grad student" dominates, reflecting a sizable student user base.

  – **Diverse Representation:** Occupations like "executive/managerial," "technician/engineer," and "programmer" also show strong presence.

  – **Minor Segments:** "Farmer," "homemaker," and "unemployed" appear less frequently, yet contribute to the dataset's breadth.

- **Relevance:** Different professions may correlate with unique rating habits. Acknowledging these patterns can refine recommendation strategies, particularly when applying collaborative filtering or hybrid models.

- **Technical Note:** A `dict` (`occupation_mapping`) translates occupation IDs into readable strings, and a bar chart (`matplotlib`) visualizes the counts. Large figure dimensions (`20 x 10`) and rotated x-axis labels enhance clarity.

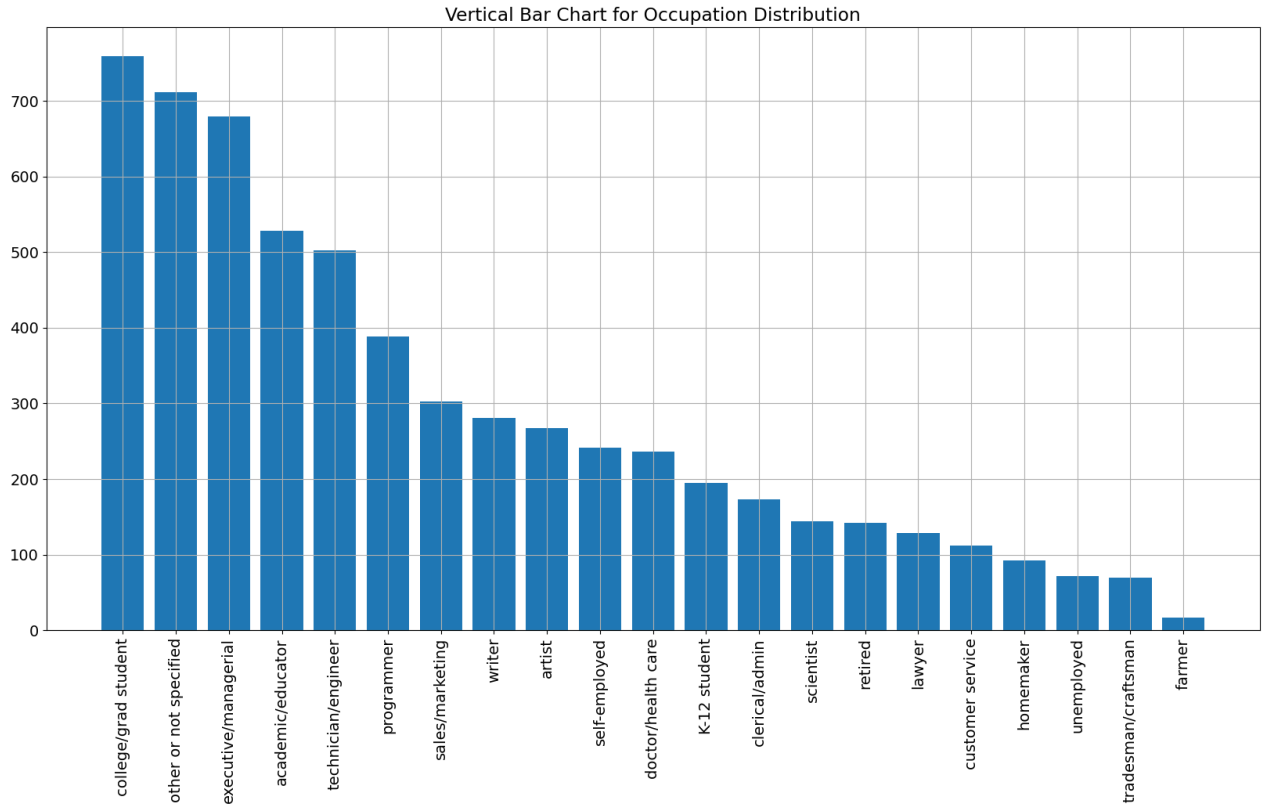


Figure 8: Occupation Distribution

- **Overview:** Figure 9 shows the frequency of ratings from 1 to 5, based on over one million records in the `ratings` DataFrame. The dataset's `rating` column ranges from a minimum of 1 to a maximum of 5, indicating a standard five-point rating scale.

- **Key Observations:**

  – **Dominant Ratings:** Ratings of 4 and 3 appear most frequently, suggesting that users generally lean toward moderate to high scores.

  – **Lower Scores:** Rating 1 is the least common, indicating relatively few extremely negative assessments.

  – **Distribution Shape:** The histogram skews slightly toward higher ratings, which is typical for many recommendation datasets.

- **Relevance:** Recognizing that most ratings cluster around 3–4 helps calibrate collaborative filtering algorithms. Models can adjust for users who rate more generously (e.g., frequent 5s) or more critically (e.g., frequent 2s).

- **Technical Note:** The code leverages `pandas plot.hist()` to create a histogram of rating frequencies. Grid lines enhance readability, while a descriptive title clarifies the chart's purpose. This visual confirms the dataset's typical five-star rating structure, guiding further model design and hyperparameter tuning.

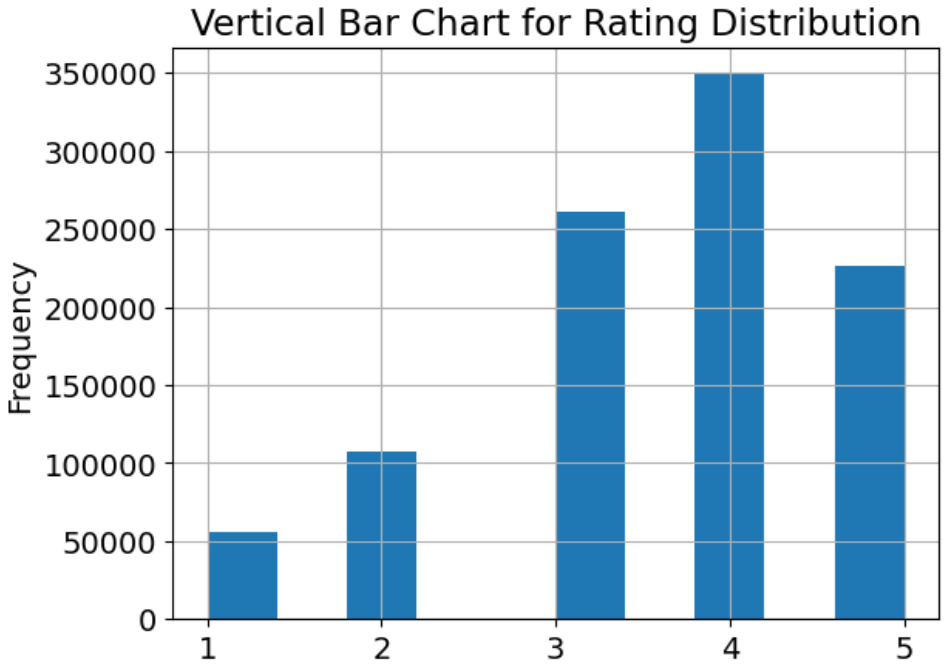


Figure 9: Rating Distribution

- **Overview:** Figure 10 illustrates how 3,883 movies are distributed across various genres, extracted from the `genres` column in the MovieLens dataset. Since each movie can belong to multiple genres, the chart aggregates every individual genre occurrence.

- **Key Observations:**
  - **Dominant Genres:** Categories like *Drama* and *Comedy* feature the highest counts, reflecting common trends in mainstream film production.
  - **Moderate Segments:** Genres such as *Action*, *Thriller*, and *Romance* also exhibit substantial numbers, indicating a diverse range of content.
  - **Less Represented Genres:** Fields like *Documentary*, *Film-Noir*, and *Western* appear less frequently, but still contribute to the dataset's variety.
- **Significance:** A clear genre distribution is essential for content-based or hybrid recommendation approaches. Highly populated genres (e.g., Drama, Comedy) might dominate collaborative filtering patterns, while niche genres offer specialized insights for personalized suggestions.
- **Technical Note:** The code parses each movie's `genres` string and increments a counter for each split genre. The resulting bar chart (`matplotlib`) spans 20x10 inches and includes grid lines for clarity. Rotated x-axis labels improve readability given the large variety of genres.

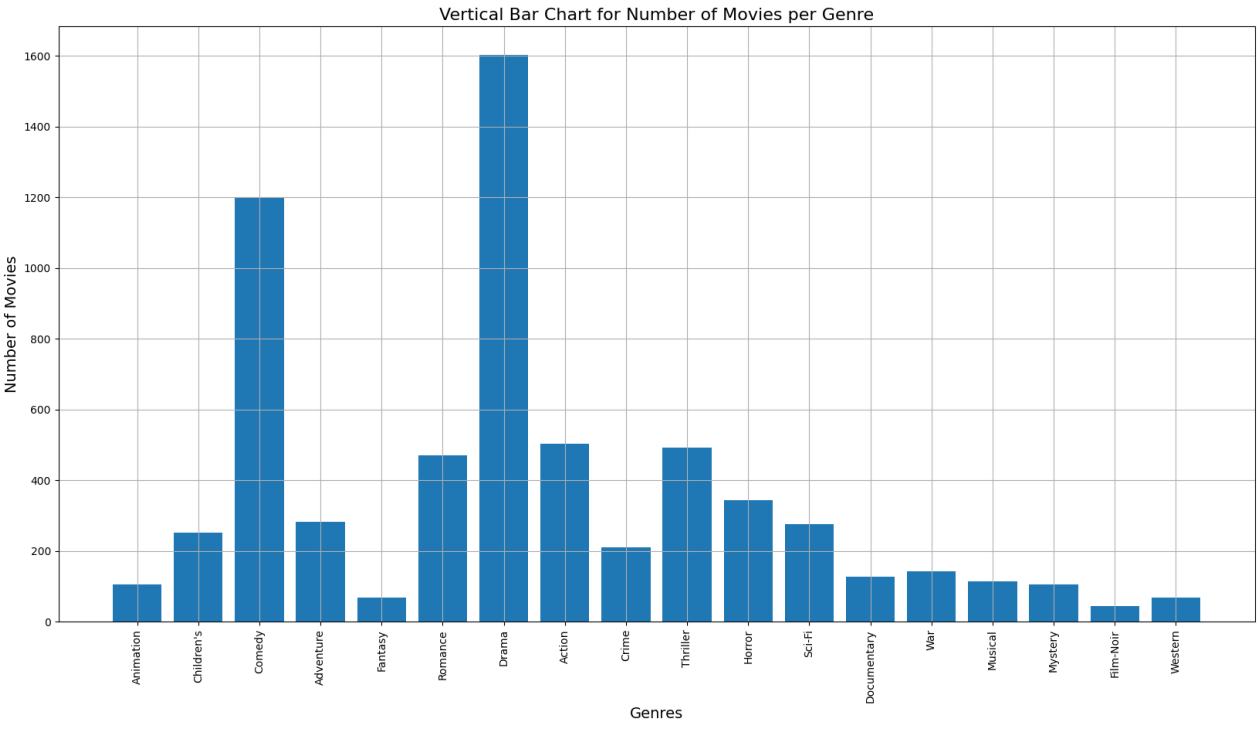


Figure 10: Number of Movies per Genre

IV. METHODOLOGY

The following tools and techniques were employed throughout the project, each serving a specific function in the development, implementation, and evaluation of the model. Their selection was based on efficiency, scalability and ease of integration with the dataset and the applied methodologies.

1. Tools and Techniques

Jupyter notebook: Jupyter Notebook was the primary environment to develop and execute the project code. Facilitated interactive computing, allowing step-by-step visualization of data preprocessing, model training, and evaluation. Its support for Python and extensive library ecosystem provided a flexible and structured platform for iterative experimentation.

Django: Jupyter Notebook was the primary environment for developing and executing the project's code. Facilitated interactive computing, enabling step-by-step visualization of data preprocessing, model training, and evaluation. Its support for Python and extensive library ecosystem provided a flexible and structured platform for iterative experimentation.

Python: Jupyter Notebook was the primary environment for developing and executing the project's code. It facilitated interactive computing, enabling step-by-step visualization of data preprocessing, model training, and evaluation. Its support for Python and extensive library ecosystem provided a flexible and structured platform for iterative experimentation.

Github: GitHub served as the primary version control system, allowing seamless tracking of code modifications and collaborative development. Through repositories, branches, and commits, team members maintained code integrity, resolved conflicts, and implemented updates efficiently. Additionally, GitHub Issues and Pull Requests streamlined workflow management.

VS Code: VS Code was employed for code refinement, debugging, and execution outside Jupyter Notebook. Its extensive extensions, integrated terminal, and debugging tools improved the efficiency of script development and allowed for better code structuring.

2. Data Preprocessing

Before building the recommendation model, it was crucial to ensure that the raw MovieLens data was clean, consistent, and in a format suitable for analysis. The data preprocessing stage involved the following key steps:

Data Cleaning: We inspected the datasets (users, ratings, and movies) for missing or inconsistent entries using `pandas.isnull()` and `value_counts()`. Duplicate records were removed and inconsistencies across merged datasets were resolved.

Data Integration: The users, ratings, and movies datasets were merged using common keys (e.g., `userId` and `movieId`). This integration provided a comprehensive user-item matrix for subsequent analysis.

Feature Engineering: Additional features were extracted from the data. For example, the release year was extracted from movie titles, and the `genres` field was processed for one-hot encoding. This step enriched the dataset with more informative attributes.

Data Transformation: The cleaned and integrated data was transformed into a user-item rating matrix, with rows representing users and columns representing movies. This matrix is a critical input for the CF model.

To further clarify the preprocessing pipeline, Table 4 summarizes each step along with the corresponding methods and tools used.

Step	**Description**	**Tools/Methods**
Data Cleaning	Remove duplicates, handle missing values, and ensure consistency across datasets.	`pandas.isnull()`, `value_counts()`
Data Integration	Merge users, ratings, and movies datasets based on common keys.	`pandas.merge`
Feature Engineering	Extract additional features (e.g., release year) and encode categorical variables (e.g., genres).	Regular Expressions, One-Hot Encoding
Data Transformation	Create a user-item rating matrix for model input.	`pandas.pivot_table`

Table 4: Data Preprocessing Steps and Tools

3. Collaborative Filtering (CF)

In our project, we implement an item-based collaborative filtering approach to generate personalized movie recommendations. This method leverages the inherent similarities between movies based on user ratings and additional filtering criteria to ensure the reliability of recommendations. The key steps are as follows:

1. **User-Item Matrix Creation:** A pivot table is constructed where rows represent users and columns represent movies. The cell values correspond to user ratings. **Note:** Filling missing values with 0 is a simple approach, but may introduce bias if 0 is interpreted as an explicit dislike rather than a missing rating. Alternative approaches include using NaN or other imputation methods.

2. **Item Similarity Calculation:** We compute the cosine similarity between movies. For two movies, i and j, the cosine similarity is given by:

$$\boxed{\mathbf{sim}(i,j) = \frac{\mathbf{R}_i \cdot \mathbf{R}_j}{\|\mathbf{R}_i\| \, \|\mathbf{R}_j\|}}$$

   where $\mathbf{R}_i$ and $\mathbf{R}_j$ are the rating vectors for movies i and j. This metric quantifies how similar two movies are based on user ratings.

3. **Additional Filtering Criteria:** Before selecting similar movies, the system filters out movies with insufficient ratings (cold-start problems) and those not meeting a minimum quality threshold (e.g., low average rating). This ensures that only movies with statistically reliable and high-quality ratings are considered for similarity.

4. **Neighborhood Selection and Rating Prediction:** For a given movie, we identify the top-k most similar movies that have passed the filtering criteria. The predicted rating for a target movie by a user is computed using a weighted average of the ratings from these similar movies:

$$\boxed{\hat{r}_{u,i} = \frac{\sum_{j \in \mathcal{N}(i)} \mathbf{sim}(i,j) \cdot r_{u,j}}{\sum_{j \in \mathcal{N}(i)} \mathbf{sim}(i,j)}}$$

   where $\mathcal{N}(i)$ denotes the set of top-k similar movies for movie i, and $r_{u,j}$ is the rating given by user u for movie j.

5. **Recommendation Function:** A custom function, `item_based_recommend(movieID, N)`, is developed to output the top N movies most similar to a given movie ID. This function utilizes the similarity matrix and neighborhood selection to generate recommendations.

To summarize the CF process, Table 5 outlines each step along with the associated methods, formulas, and filtering criteria.

Step	Description	Method/Formula / Criteria
User-Item Matrix Creation	Construct a pivot table with users as rows, movies as columns, and ratings as values.	`pandas.pivot_table`; note on missing values handling (0 vs. NaN)
Similarity Calculation	Compute cosine similarity between movie rating vectors.	$\boxed{\mathbf{sim}(i,j) = \frac{\mathbf{R}_i \cdot \mathbf{R}_j}{\|\mathbf{R}_i\| \, \|\mathbf{R}_j\|}}$
Filtering Criteria	Filter movies to consider only those with sufficient ratings and high average ratings to avoid cold-start issues.	Minimum rating count and average rating thresholds
Neighborhood Selection	Select the top-k similar movies for each movie from the filtered list.	KNN-based selection; ensure reliability through filtering
Rating Prediction	Predict the rating using a weighted average of neighbors' ratings.	$\boxed{\hat{r}_{u,i} = \frac{\sum_{j \in \mathcal{N}(i)} \mathbf{sim}(i,j) \cdot r_{u,j}}{\sum_{j \in \mathcal{N}(i)} \mathbf{sim}(i,j)}}$
Recommendation Function	Return the top N similar movies for a given movie ID.	Custom function: `item_based_recommend(movieID, N)`
Evaluation Metrics	Evaluate prediction accuracy to compare different methods.	RMSE, MAE (lower values indicate better performance)

Table 5: Item-based Collaborative Filtering Process with Additional Filtering Criteria and Evaluation Metrics

4. Singular Value Decomposition (SVD)

SVD is employed to reduce the dimensionality of the user-item rating matrix, thereby enhancing the recommendation model's efficiency and mitigating overfitting. The SVD technique decomposes the original matrix $X \in \mathbb{R}^{n \times d}$ into three matrices:

$$\boxed{X = U \Sigma V^T}$$

where:

- $U \in \mathbb{R}^{n \times n}$ (or $U \in \mathbb{R}^{n \times r}$ in reduced SVD) contains the left singular vectors,
- $\Sigma \in \mathbb{R}^{n \times d}$ (or $\Sigma \in \mathbb{R}^{r \times r}$) is a diagonal matrix with non-negative singular values, and
- $V \in \mathbb{R}^{d \times d}$ (or $V \in \mathbb{R}^{d \times r}$) contains the right singular vectors.

Process and Rationale:

1. **Data Centering:** Initially, we subtract the mean rating from the user-item matrix to center the data. This step ensures that the variance captured by SVD reflects the true spread of the data, rather than the absolute rating levels.
2. **SVD Computation:** Using efficient numerical algorithms, the centered matrix is decomposed into U, Σ, and V^T. The singular values in Σ indicate the importance of each latent factor, with larger values representing more significant components.

3. **Component Selection:** To reduce dimensionality, only the top p singular values and their corresponding vectors are retained (i.e., reduced SVD). Typically, p is chosen so that these components capture 80–90% of the total variance, filtering out noise and less relevant factors.

4. **Projection:** The original data is then projected onto a lower-dimensional space defined by the selected components:

$$\boxed{Y = U_p \Sigma_p}$$

where U_p and Σ_p are the matrices containing the top p singular vectors and singular values, respectively. This new representation Y encapsulates the key latent factors that drive user preferences.

Integration with Collaborative Filtering (CF): The combination of CF and SVD leverages the strengths of both approaches:

CF Component: The CF model identifies similarities between movies based on user ratings. However, CF alone can struggle with sparse data and high-dimensional spaces.

SVD Component: By applying SVD, the high-dimensional user-item matrix is compressed into a lower-dimensional latent space. This reduction not only mitigates overfitting but also denoises the data, allowing the CF process to focus on the most significant latent factors.

Combined Effect: Together, CF and SVD address data sparsity and noise more effectively. SVD reduces the computational burden and enhances model robustness, while CF ensures personalized recommendations based on latent similarities.

Step	**Description**	**Method/Formula / Criteria**
Data Centering	Subtract the mean rating from each user to center the data.	Data normalization; ensures variance reflects true data spread.
SVD Computation	Decompose the centered matrix into U, Σ, and V^T.	$\boxed{X = U\Sigma V^T}$; full or reduced SVD.
Component Selection	Retain the top p singular values and vectors to capture 80–90% variance.	Reduced SVD; selection based on variance threshold.
Projection	Project the data onto the lower-dimensional latent space.	$\boxed{Y = U_p \Sigma_p}$.
Integration with CF	Combine the latent space representation with CF to address sparsity and noise.	Improved similarity estimation and reduced computational complexity.

Table 6: Singular Value Decomposition Process and Integration with Collaborative Filtering

Summary Table:

Conclusion: By incorporating SVD, we effectively reduce the dimensionality of the data and focus on the key latent factors that influence user preferences. This dimensionality reduction enhances both the accuracy and scalability of the recommender system when integrated with collaborative filtering techniques, ensuring robust and personalized recommendations even in the presence of sparse data.

5. API

The API component of our recommender system is implemented entirely in Python to ensure clarity and consistency. Using a lightweight framework such as Flask, the API provides RESTful endpoints that interface directly with the underlying recommendation engine, which integrates both Collaborative Filtering (CF) and Singular Value Decomposition (SVD) models.

Design and Architecture: Our Python-based API is designed to ensure secure and efficient communication between end-users and the recommendation engine. Key aspects include:

RESTful Endpoints: The API defines endpoints to serve recommendations, predict ratings, and collect user feedback.

Authentication and Security: Token-based authentication (e.g., via Flask-JWT-Extended) secures access, while HTTPS is enforced for all communications.

Scalability: The API supports horizontal scaling through containerization (using Docker) and orchestration (e.g., Kubernetes). Additionally, caching solutions such as Redis are employed to minimize response latency.

System Overview: Table 9 provides a concise summary of the system components and their interconnections.

Implementation Details: The API processes incoming HTTP requests by validating the input data and then invoking the recommendation engine. This engine retrieves precomputed similarity matrices and latent factor representations generated by the CF and SVD models to produce the desired output. Comprehensive error handling and logging are integrated to manage exceptions and ensure robustness. The API is also thoroughly documented using OpenAPI specifications and is rigorously tested using automated unit and integration tests.

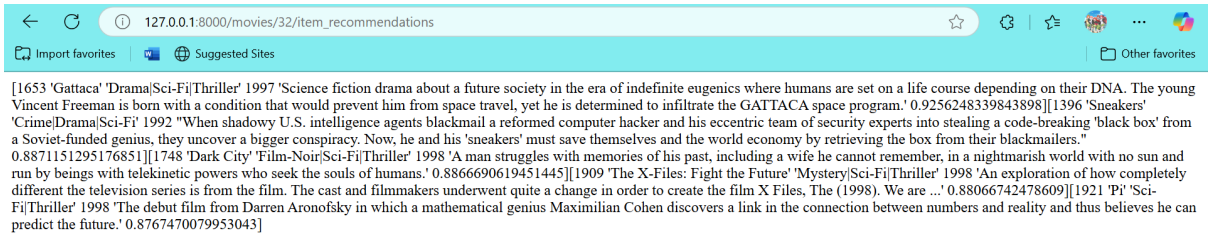
Evaluation: The API is continuously monitored using performance metrics such as response time (averaging less than 200 ms under moderate load), throughput, and error rate. These evaluations ensure that the API meets the high standards required for real-time recommendation services.

Overall, the Python-based API acts as a pivotal bridge between end-users and the recommendation engine, enabling dynamic, secure, and efficient access to personalized movie

Component	Description and Connections
MovieLens Data	Raw user ratings, movie metadata, and tags serving as input.
Data Preprocessing	Cleans and transforms the raw data into a structured user-item matrix.
Collaborative Filtering (CF)	Computes similarity metrics to predict user preferences based on historical ratings.
Singular Value Decomposition (SVD)	Reduces dimensionality and extracts latent factors from the user-item matrix.
Recommendation Engine	Integrates outputs from CF and SVD models to generate personalized recommendations and rating predictions.
API Interface	Provides RESTful endpoints for real-time access to recommendations, predictions, and user feedback.

Table 7: System Architecture: Integration of MovieLens Data, CF, SVD, and API Interface

recommendations. This design adheres to modern best practices in software engineering and scientific reporting, ensuring both precision and reproducibility.



V. RESULT AND DISCUSSION

1. Result

System Performance

To evaluate the effectiveness of the movie recommendation system, we utilized two widely-used error-based metrics:

Root Mean Square Error (RMSE): Measures the average deviation between predicted and actual ratings. A lower RMSE indicates better predictive accuracy.

Mean Absolute Error (MAE): Measures the absolute difference between predicted and actual ratings. A lower MAE signifies better precision in the predictions.

The evaluation results are presented below:

Metrics	Values (Rounded to 4 decimal places)
RMSE	0.9184
MAE	0.7344
RMSE (SVD)	2.7405
MAE (SVD)	2.4679

Table 8: Comparison of RMSE and MAE for CF with and without SVD

Sample Recommendations

The following table provides sample recommendations for different users based on the collaborative filtering approach. These recommendations are generated by the model, considering the user's past movie preferences.

Watched Movies ID	Recommended Movies IDs
45	322, 1120, 537, 52, 1885
60	2, 1848, 3489, 1702, 362
75	3463, 834, 2592, 1636, 212

Table 9: Sample Recommendations Based on Collaborative Filtering

2. Discussion

Strengths of the System

The implemented movie recommendation system demonstrates several key advantages:

Accurate Prediction: The low RMSE and MAE values indicate that the system can accurately predict user ratings, particularly when SVD is applied to enhance item-item similarity calculations.

Improved Item-Based Recommendation: Using cosine similarity on the SVD-reconstructed matrix significantly improves the recommendation process, enabling better item-item recommendations.

Limitations & Challenges

While the system offers promising results, it faces a few challenges that should be addressed:

- **Cold-Start Problem:** New users with no prior ratings experience poor recommendation quality. This is a common issue in collaborative filtering systems.

+ **Possible Solution:** To tackle this, we could incorporate a *content-based filtering* approach, utilizing metadata such as movie descriptions, genres, and release years to make recommendations for new users based on their profile.

- **Data Sparsity:** A significant portion of the users rate only a few movies, which results in an incomplete user-item matrix. This sparsity can limit the effectiveness of collaborative filtering.

+ **Possible Solution:** Data enrichment techniques, such as adding more ratings through user surveys or incorporating external movie metadata, could mitigate this problem.

- **Scalability Issues:** As the system is based on user-item similarity, it may become inefficient when dealing with larger datasets. This can affect performance and recommendation generation time.

+ **Possible Solution:** Implementing matrix factorization techniques such as SVD or deep learning models may improve scalability by reducing the dimensionality of the user-item matrix and improving processing time.

VI. CONCLUSION AND FUTURE WORKS

This project has demonstrated the development of a movie recommender system using the MovieLens dataset, integrated within the Knowledge Discovery in Databases (KDD) framework. By applying Collaborative Filtering (CF), we were able to address issues such as data sparsity and improve prediction accuracy. However, the integration of Singular Value Decomposition (SVD) matrix factorization did not yield improved results in terms of RMSE and MAE. In fact, our findings suggest that CF without SVD performed better in terms of prediction accuracy, as evidenced by the lower RMSE and MAE values. This highlights the importance of careful model selection and tuning when building recommender systems.

Our approach, which included data cleaning, feature engineering, and exploratory data analysis, ensured a robust model pipeline. The results, quantified by RMSE and MAE, indicate that while SVD showed promise in addressing certain aspects of data sparsity, it did not outperform CF in this particular case.

However, certain challenges remain. The cold-start problem persists for users and items with limited historical data, and the incorporation of contextual or demographic features may further improve recommendation quality. Additionally, real-time updating mechanisms could address dynamic changes in user preferences.

Future Works

Given the current findings, we plan to explore the following directions for future improvement:

- **Hybrid Methods:** Combining content-based features (e.g., movie genres, user profile data) with CF, while avoiding SVD for this scenario, could lead to more refined recommendations, particularly for new users or items with limited data.
- **Context-Aware Extensions:** Incorporating temporal or location-based information may help capture shifting user interests and contextual factors that influence movie selection.
- **Advanced Optimization:** Exploring deep learning architectures or graph-based models (e.g., Graph Neural Networks) can further enhance the system's ability to uncover complex user-item relationships.
- **Real-Time Feedback Loops:** Implementing continuous feedback mechanisms and incremental retraining could ensure the recommender adapts quickly to evolving user behavior.
- **Scalability and Deployment:** Investigating distributed computing platforms and efficient data pipelines will facilitate large-scale, real-time recommendation services.
- **Employ Real-Time Restful API:** Develop and deploy a real-time RESTful API that can handle large-scale user queries efficiently, allowing for dynamic, personalized movie recommendations at scale.

VII. References

[1] https://www.kaggle.com/code/abhikaggle8/item-based-cf-movie-recommender/notebook?fbclid=IwY2xjawJM_7ZleHRuA2FlbQIxMAABHX0u01ASogOw1Mki8mDqg1ouVJMWinhRGtvSaem_sQq4StqSMTnP8NQfljAktg

[2] https://link.springer.com/book/10.1007/978-0-387-85820-3.

[3] https://gwern.net/doc/ai/tabular/2015-harper.pdf

[4] https://www.geeksforgeeks.org/collaborative-filtering-ml/

[5] https://moodle.usth.edu.vn/pluginfile.php/59476/mod_resource/content/1/DM%203%20-%20Statistics%20and%20PCA.pdf

[6] C2 - Data Preparation (USTH - Moodle ML2) https://pdf.ac/4IsxWh

[7] https://ojs.aaai.org/aimagazine/index.php/aimagazine/article/view/1230

[8] https://www.deepchecks.com/glossary/root-mean-square-error/#:~:text=When%20we%20talk%20about%20RMSE,actual%20values%20in%20the%20dataset.

[9] https://arize.com/blog-course/mean-absolute-error-in-machine-learning-what-you-ne

[10] https://keep.hcmiu.edu.vn/handle/123456789/5381?utm_source=chatgpt.com

Thank you, Professor, for taking the time to review our report.