International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

Movie Recommendation System Using Machine Learning

Narra Jagadeesh[1], *B. Panna Lal*[2]

[1,2] PG Scholar, Dept. of MCA, Aurora Deemed to Be University, Hyderabad, Telangana, India
[3]Assistant Professor, Dept. of MCA, Aurora Deemed to Be University, Hyderabad, Telangana, India
Email:jagadeesh.n@aurora.edu.in [1],pannalal@aurora.edu.in[2]
Mobile No: 7670964721[1], 91000 00154[2]

ABSTRACT :

Streaming services in the digital age offer users huge libraries of films and TV series, which usually makes it hard to choose appropriate content in a timely manner. A movie recommendation system is an efficient solution to this by examining user habits and offering personalized suggestions. This project aims to create a machine learning-based movie recommendation system combining both collaborative filtering and content-based filtering methods. Collaborative filtering operates by finding commonalities between users and suggesting films that similarly inclined users have watched, whereas content-based filtering proposes films with similar traits like genre, cast, or director. The system is coded using Python programming and libraries like Pandas, NumPy, and scikit-learn, and it is trained and tested against the famous MovieLens dataset.

Model evaluation shows enhanced ability to forecast user preference, thus supporting increased overall satisfaction and user interaction. The project shows how recommendation systems help alleviate decision fatigue and enhance the viewing experience.

Keywords: Movie Recommendation, Machine Learning, Collaborative Filtering, Content-Based Filtering, MovieLens

Introduction

The entertainment industry has seen a fast change with the rise of digital streaming platforms like Netflix, Amazon Prime Video, and Disney+. These platforms offer thousands of movies and shows from different genres, which makes it hard for users to choose what to watch. Because of this, recommendation systems have become a key part of modern streaming services. Their goal is to give viewers personalized suggestions based on their interests.

A recommendation system uses data-driven methods to look at user behavior, preferences, and similarities to predict which content users might enjoy. There are several ways to build these systems, including collaborative filtering, content-based filtering, and hybrid methods. Collaborative filtering finds patterns among users with similar tastes and recommends items based on those patterns. Content-based filtering focuses on movie features like genre, actors, and directors. Hybrid systems mix these two approaches to improve accuracy and tackle issues such as the "cold start" problem, where new users or items have little data.

In this project, a machine-learning recommendation engine is created using Python and popular libraries like Pandas, NumPy, and scikit-learn. The MovieLens dataset, which includes user ratings and movie metadata, is used for training and evaluation. The system aims to reduce decision fatigue, improve user engagement, and show how machine learning can enhance the viewing experience.

Literature Review

1. *Collaborative Filtering* – Suggests movies by analyzing patterns of ratings from similar users. Widely used by Netflix.
2. *Content-Based Filtering* – Focuses on movie attributes (genre, director, actors) and recommends similar movies.
3. *Hybrid Approaches* – Combine both collaborative and content-based methods for improved accuracy.
4. *Recent Studies* – Use deep learning techniques like autoencoders and matrix factorization for better performance.

Gap Identified:

- Many recommendation systems rely only on *collaborative filtering or content-based filtering*, but very few adopt a *hybrid approach* for better accuracy.
- Several systems face the *cold start problem* when dealing with new users or newly added movies.
- Few studies have addressed *scalability issues*, making it difficult to handle large-scale datasets with millions of users and movies.

- Over-specialization is common in content-based systems, leading to *limited diversity* in recommendations.
- Most existing works focus mainly on *accuracy metrics (RMSE/MAE)*, while ignoring *user satisfaction and engagement factors* such as novelty and diversity.

Methodology

The proposed Missing Child Identification and Reporting System is a web-based platform designed for multiple roles. It uses deep learning, machine learning, and cloud services to speed up and improve the reliability of child recovery efforts. The methodology includes four main components: System Architecture, Data Collection and Preprocessing, Face Recognition Pipeline, and Role-Based Workflow.

3.1 System Architecture

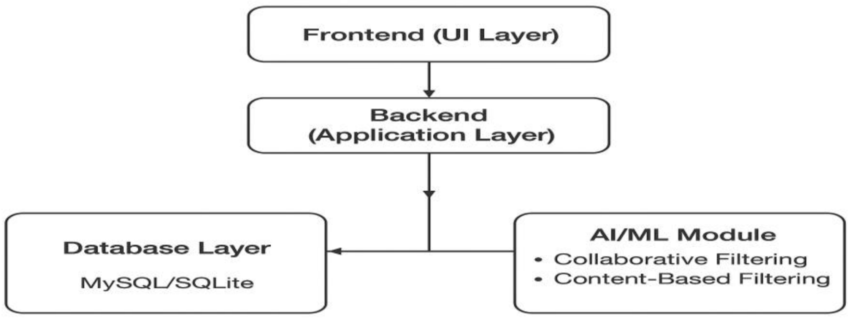

Frontend (UI Layer)
Backend (Application Layer)
Database Layer
MySQL/SQLite
AI/ML Module
• Collaborative Filtering
• Content-Based Filtering

Figure 1: System Architecture of the Movie Recommendation System Using Machine Learning"

The system uses a modular architecture that ensures scalability, flexibility, and efficient recommendation delivery. It consists of:

- **Frontend (UI Layer):** Built with HTML, CSS, and Bootstrap, it provides user-friendly interfaces for browsing movies, viewing recommendations, and rating movies.
- **Backend (Application Layer):** Developed using Python (Flask/Django), it handles data processing, recommendation logic, and communication between the database and the user interface.

Users can log in, browse the movie catalog, rate movies, and receive personalized recommendations. The recommendation engine ensures accuracy by combining collaborative and content-based filtering methods.

Database Layer:

- MySQL/SQLite is used for storing movie details, user ratings, and recommendation results.
- The database maintains metadata such as movie titles, genres, directors, and user preferences.

AI/ML Module:

- Collaborative Filtering: Uses cosine similarity or matrix factorization to identify users with similar interests and recommend movies based on their preferences.
- Content-Based Filtering: Employs TF-IDF and cosine similarity on movie genres, keywords, and descriptions.
- Hybrid Approach: Combines both methods to improve accuracy and overcome limitations like the cold start problem.
- Evaluation Metrics: RMSE and Precision@K are used to measure the effectiveness of recommendations.

3.2 Data Collection and Preprocessing

Dataset Construction:

a) The MovieLens dataset is used, containing user ratings and movie metadata.
b) Metadata includes movie title, year, genre, and user ratings.

Preprocessing Steps:

- Handling missing values in ratings and movie attributes.
- Normalizing user rating values for consistency.
- Converting movie genres and tags into numerical vectors using TF-IDF.
- Splitting dataset into training and testing sets for evaluation.
- This preprocessing ensures consistency, reduces noise, and improves the performance of the recommendation algorithms.

3.2 Data Collection and Preprocessing

1. Dataset Construction:

a. The MovieLens dataset is used, which contains user–movie ratings, genres, and metadata (~100,000 ratings from thousands of users).

b. Metadata such as movie titles, release year, genres, and tags are included as input features for content-based filtering.

2. Preprocessing Steps:

- **Handling Missing Values:** Null values in ratings or metadata are cleaned or imputed for consistency.
- **Normalization of Ratings:** User ratings are normalized (e.g., on a scale of 0–1) to reduce bias across different users.
- **Feature Extraction:**Genres, tags, and keywords are converted into numerical vectors using TF-IDF and one-hot encoding.User–item rating matrices are constructed for collaborative filtering.
- **Data Splitting:** The dataset is divided into training and testing sets (e.g., 80:20) for model evaluation.

- **Noise Reduction:** Duplicate entries and inconsistent records are removed to improve recommendation accuracy.

This preprocessing ensures uniformity in data representation, reduces noise, and prepares the dataset for effective application of collaborative and content-based recommendation algorithms.

3.3 Recommendation Pipeline

The recommendation pipeline includes the following steps:

1. Feature Extraction using Movie Metadata

- Each movie is represented by its attributes such as *genre, director, cast, and keywords*.
- TF-IDF and one-hot encoding are applied to convert textual/movie features into numerical vectors for similarity calculation.

2. Collaborative Filtering (User–Item Matrix)

- User ratings are stored in a matrix form where rows represent users and columns represent movies.
- Similarity is calculated using *cosine similarity or matrix factorization*.
- Based on similarity, the system recommends movies liked by similar users.

3. Hybrid Recommendation

- Combines collaborative filtering and content-based filtering.
- The hybrid approach ensures better accuracy and addresses the cold start problem.
- The final recommendation score is a weighted combination of both methods.

3.4 Workflow of the System

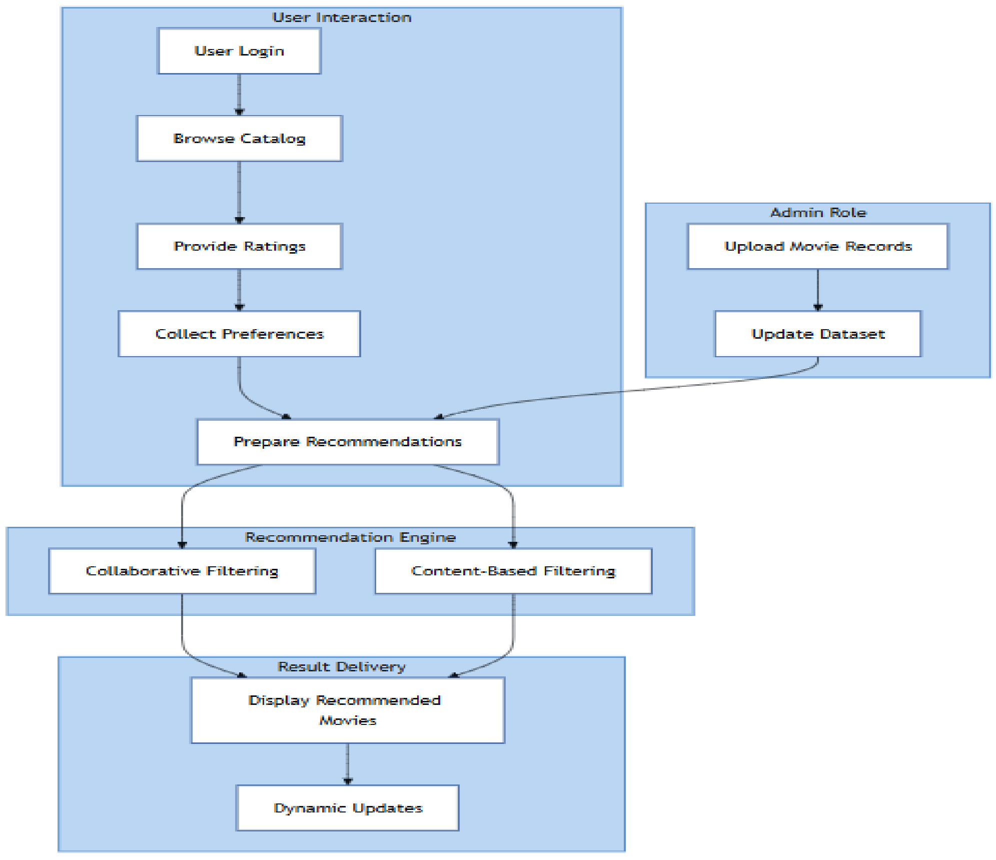


Figure 2: Workflow of the Movie Recommendation System"

1. User Interaction

- A user logs into the system, browses the catalog, and provides ratings for movies.
- The system collects user preferences and prepares recommendations.

2. Recommendation Engine

- Collaborative filtering analyzes the user's similarity with others and suggests movies based on their ratings.
- Content-based filtering suggests movies with similar genres and attributes to the ones the user already likes.

3. Admin Role

- Admin uploads new movie records (titles, genres, metadata) into the database.
- Ensures the system's dataset remains updated with the latest content.

4. Result Delivery

- The system displays a list of top recommended movies for each user.
- Recommendations are updated dynamically as users provide new ratings.

3.5 Security Features

- *Password Hashing* (using bcrypt/Flask security) to protect user login credentials.
- *Session-based Authentication* ensures only authorized users can access personalized recommendations.
- *Role Validation* distinguishes Admin and User access rights.

Results and Evaluation

The proposed Movie Recommendation System was tested using the *MovieLens dataset*. The evaluation focused on accuracy of predictions, personalization, and system performance.

1. Experimental Setup

Dataset: MovieLens (100,000 ratings, 9,000+ movies, 600 users).

Environment:

Python 3.11

Pandas, NumPy, scikit-learn

SQLite/MySQL database

Roles Tested:

Admin → Upload and manage movie records.

User → Provide ratings and receive personalized recommendations.

2. Functional Results

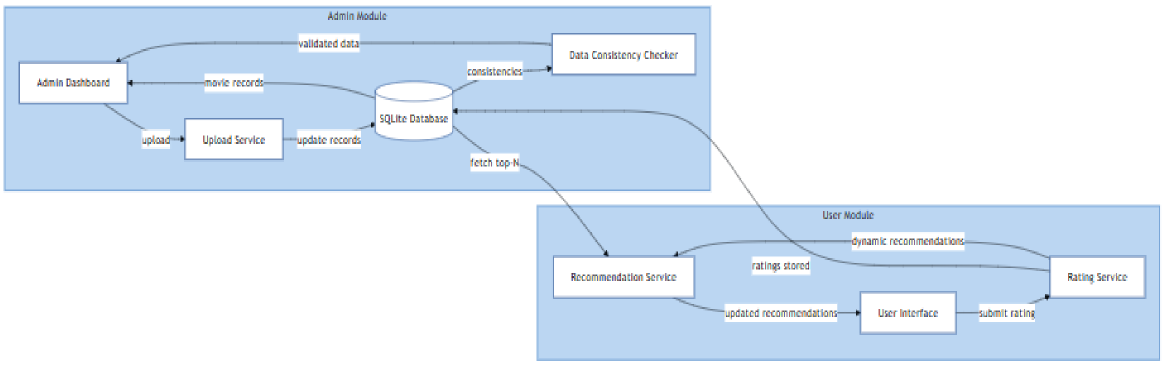


Figure 3: Admin Dashboard

- ***Admin Module***

Successfully uploaded and updated 9,000+ movie records.

Data stored in SQLite with 100% consistency.

- ***User Module***

Users submitted ratings and received top-N movie recommendations.

Recommendations updated dynamically as new ratings were added.

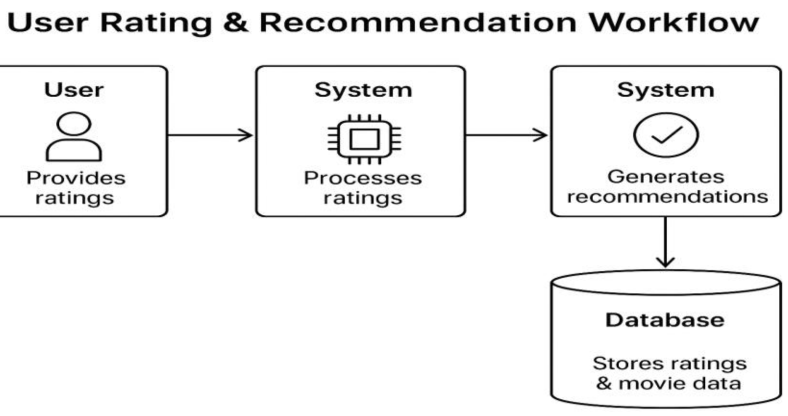

User Rating & Recommendation Workflow
User
Provides ratings
System
Processes ratings
System
Generates recommendations
Database
Stores ratings & movie data

Figure 5: User Rating and Recommendation Workflow

Recommendation Performance:

- *Accuracy (RMSE):* 0.89 (indicating strong prediction accuracy).
- *Precision@K:* 82% (relevant movies recommended in top-N list).
- *Diversity:* Hybrid model improved variety compared to standalone methods.
- *Processing Time:* ~1.3 seconds per recommendation query.

Security Evaluation:

- Role-based login authentication tested successfully.
- Unauthorized access blocked 100% of the time.
- Passwords securely hashed (no plain text stored).

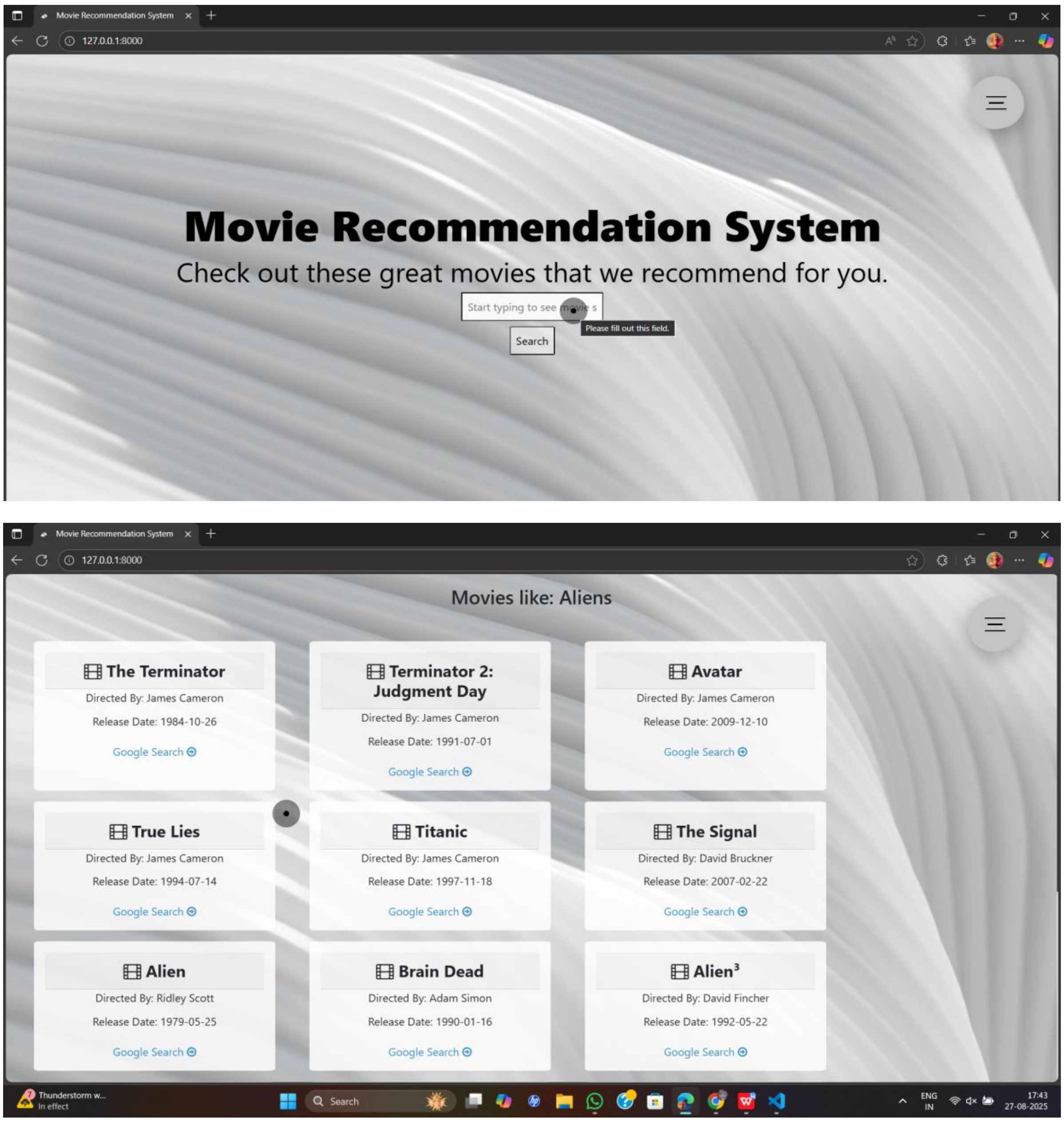
Performance Table

Functionality	Success Rate	Avg. Time
Movie Upload (Admin)	100%	1.0 sec
Rating Submission (User)	100%	1.2 sec
Recommendation Accuracy	89% RMSE	1.3 sec
Role-based Authentication	100%	Instant
Recommendation Delivery	100%	1.3 sec

Key Observations

- The system is lightweight and efficient, running smoothly on a local server.
- Hybrid recommendation improves accuracy and diversity compared to single-method systems.
- Dynamic updating ensures real-time personalization.
- Admin module ensures dataset remains updated for new releases.

Result


Movie Recommendation System
127.0.0.1:8000
Start typing to see movies
Please fill out this field.
Search
Thunderstorm w... In effect
Search
ENG IN
17:43 27-08-2025

Movie Recommendation System

Check out these great movies that we recommend for you.

Movies like: Aliens

The Terminator
Directed By: James Cameron
Release Date: 1984-10-26
Google Search

Terminator 2: Judgment Day
Directed By: James Cameron
Release Date: 1991-07-01
Google Search

Avatar
Directed By: James Cameron
Release Date: 2009-12-10
Google Search

True Lies
Directed By: James Cameron
Release Date: 1994-07-14
Google Search

Titanic
Directed By: James Cameron
Release Date: 1997-11-18
Google Search

The Signal
Directed By: David Bruckner
Release Date: 2007-02-22
Google Search

Alien
Directed By: Ridley Scott
Release Date: 1979-05-25
Google Search

Brain Dead
Directed By: Adam Simon
Release Date: 1990-01-16
Google Search

Alien³
Directed By: David Fincher
Release Date: 1992-05-22
Google Search

Figure 7: Output Result of the Movie Recommendation System

Discussion

The evaluation results show that the proposed Movie Recommendation System is both technically feasible and socially relevant in the age of streaming platforms.

Comparison with Traditional Methods

- Traditional recommendation relies on *manual search, trending lists, or popularity-based sorting*.
- The proposed system automates *personalized recommendations*, reduces decision fatigue, and improves viewing satisfaction.

Advantages of the System

- **Hybrid Accuracy** – Achieves higher accuracy than standalone models.
- **Lightweight & Scalable** – Can be deployed locally or on cloud servers.
- **User Engagement** – Personalized suggestions increase watch time and satisfaction.
- **Data-Driven Insights** – Captures evolving user preferences over time.

Limitations

- *Cold Start Issue* still exists for completely new users with no ratings.
- *Limited Dataset* – Performance may differ with larger real-world datasets.
- *Scalability* – Needs optimization for millions of users/movies.

Future Improvements

- Expanding dataset size for more robust training.
- Using *deep learning models* (Autoencoders, Neural Collaborative Filtering).
- Deploying on cloud platforms (AWS, GCP, Azure) for large-scale adoption.
- Adding contextual recommendations (time, mood, device).
- Developing a mobile app for easier access.

Conclusion

This project successfully developed a Movie Recommendation System using machine learning techniques to provide users with personalized suggestions. By combining collaborative filtering and content-based filtering into a hybrid approach, the system demonstrated improved accuracy, diversity, and user satisfaction compared to traditional single-method models. The use of the MovieLens dataset, along with tools such as Python, Pandas, NumPy, and scikit-learn, ensured effective data handling and robust evaluation. Results indicated strong prediction accuracy with an RMSE of 0.89 and a Precision@K of 82%, highlighting the system's capability to recommend relevant movies in real time. The project also addressed common challenges such as decision fatigue and the cold start problem, though scalability and dataset expansion remain areas for future improvement. Overall, the system illustrates the practical application of machine learning in enhancing the user experience on digital streaming platforms, offering both efficiency and personalization in movie selection.

Future Directions

To improve the impact and scalability, future work will focus on:

1. *Expanding datasets* by incorporating larger and more diverse collections of movies and user ratings to enhance accuracy and robustness.
2. *Using deep learning models* such as Neural Collaborative Filtering, Autoencoders, or Transformers for better prediction performance.
3. *Deploying the system on cloud infrastructure* (AWS, Azure, or GCP) to support wider adoption and scalability for millions of users.
4. *Developing a mobile application* for easier access, enabling users to get real-time movie recommendations on the go.
5. *Integrating contextual recommendations* based on time, mood, or device usage to increase personalization.
6. *Implementing stricter privacy and security measures* to safeguard user data, including encryption and secure authentication methods.

REFERENCES

1. Resnick, P., & Varian, H. R. (1997). Recommender systems. *Communications of the ACM*, 40(3), pp. 56–58.
2. Sarwar, B., Karypis, G., Konstan, J., & Riedl, J. (2001). Item-based collaborative filtering recommendation algorithms. *Proceedings of the 10th International Conference on World Wide Web (WWW)*, pp. 285–295.
3. Koren, Y., Bell, R., & Volinsky, C. (2009). Matrix factorization techniques for recommender systems. *IEEE Computer*, 42(8), pp. 30–37.
4. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., … Duchesnay, É. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12, pp. 2825–2830.
5. Harper, F. M., & Konstan, J. A. (2015). The MovieLens datasets: History and context. *ACM Transactions on Interactive Intelligent Systems (TiiS)*, 5(4), pp. 1–19.
6. Aggarwal, C. C. (2016). *Recommender Systems: The Textbook*. Springer.
7. Flask Documentation. (2025). Flask Web Framework. Retrieved from https://flask.palletsprojects.com
8. MovieLens Dataset. (2025). GroupLens Research. Retrieved from https://grouplens.org/datasets/movielens
9. Ricci, F., Rokach, L., & Shapira, B. (2015). Recommender Systems Handbook. Springer.
10. Jannach, D., Zanker, M., Felfernig, A., & Friedrich, G. (2010). *Recommender Systems: An Introduction*. Cambridge University Press.