

# Recommender Systems

---

*Course Project*

---



**Title: Movie Recommendation System**

**Group Members:**

- i) Muhammad Owais Siyal (20k-0201)
- ii) Abrar Saleem (20k-1029)
- iii) Okasha Arshad (20K-1868)

## Introduction:

In today's digital era, where the entertainment industry is driven by vast arrays of content, finding movies that resonate with individual preferences is challenging. Streaming platforms provide access to thousands of films, making it overwhelming for users to sift through the extensive libraries to find movies they'll enjoy. Here, a movie recommendation system plays a crucial role in guiding users toward films aligned with their tastes.

This project aims to alleviate the challenge of content overload by implementing a sophisticated movie recommendation system that utilizes three distinct algorithms:

1. **Content-Based Filtering:** This method recommends movies based on specific characteristics of films that the user has previously enjoyed. By analyzing features like genres, directors, actors, and keywords, it identifies movies that share similar attributes to those the user has rated highly. This tailored approach provides a more accurate prediction of user preferences.
2. **Matrix Factorization Using SVD:** This technique leverages Singular Value Decomposition (SVD) to uncover hidden patterns in user preferences. By breaking down the user-movie rating matrix into latent factors, it efficiently captures relationships between users and movies. This powerful method enables the system to generate highly accurate and scalable recommendations.
3. **Naïve Collaborative Filtering:** Although simple in its approach, Naïve Collaborative Filtering provides a robust baseline by suggesting popular movies based on straightforward heuristics such as average ratings and overall popularity. Despite its simplicity, it can sometimes offer surprisingly effective recommendations due to the inherent popularity of certain movies.

The project's primary objective is to develop an intelligent recommendation system that accurately predicts and suggests movies tailored to users' unique preferences. This enhances the movie discovery experience, helping users find new favorites without the frustration of endless browsing. By integrating three varied algorithms into a single system, users can enjoy a seamless, personalized experience, discovering films that align with their tastes effortlessly.

Beyond offering valuable recommendations, this project also aims to explore the strengths and limitations of each algorithm. By comparing these approaches, we can understand which methods work best for different user scenarios. Ultimately, this helps improve the recommendation system's adaptability and accuracy, ensuring that each user receives the best possible suggestions based on their viewing habits.

## Algorithms Overview

### 1. Content-Based Filtering:

Content-Based Filtering tailors recommendations based on a user's preferred movie attributes, such as genres, directors, and actors. By analyzing the features of movies, the user has previously rated highly, it recommends films with similar characteristics.

Pseudocode:

```
1. Load the movie metadata (titles, genres, etc.)
2. For each movie:
  a. Extract relevant features (genres, actors, etc.)
  b. Create a vector representation of the movie using its features
3. For a given user:
  a. Identify movies the user has rated highly
  b. Find movies similar to these based on feature vectors
  c. Rank and recommend top N similar movies
```

### 2. Matrix Factorization Using SVD:

Matrix Factorization, with Singular Value Decomposition (SVD), extracts latent factors from the user-movie ratings matrix to uncover hidden patterns in user preferences. It uses these patterns to offer scalable and accurate recommendations.

Pseudocode:

```
1. Load user-movie ratings
2. Normalize ratings to account for user biases
3. Apply SVD to decompose the ratings matrix into latent factors
4. For a given user:
  a. Calculate the predicted rating for unrated movies using latent factors
  b. Rank and recommend top N movies with highest predicted ratings
```

### 3. Naïve Collaborative Filtering:

Naïve Collaborative Filtering uses straightforward heuristics like average ratings and popularity to suggest movies. While it lacks the sophistication of other algorithms, it provides a useful baseline and can sometimes deliver surprisingly accurate recommendations.

Pseudocode:

```
1. Load user-movie ratings
2. Calculate the average rating for each movie
3. Rank movies based on their average ratings or popularity
4. Recommend top N movies to the user
```

## Techniques Implemented

- **Content-Based Filtering:** Suggests movies based on their features and characteristics.
- **Matrix Factorization with SVD:** Uses SVD to predict user-movie preferences based on latent factors.
- **Naïve Collaborative Filtering:** Recommends popular movies using simple heuristics.

## Workflow

### Data Collection

- Gather comprehensive movie metadata, such as titles, genres, and user ratings.
- Source data from online movie databases, user reviews, and prior interactions.

### Data Pre-processing

- Clean the data and handle missing values.
- Normalize user ratings and format the data for each algorithm.

### Algorithm Implementation

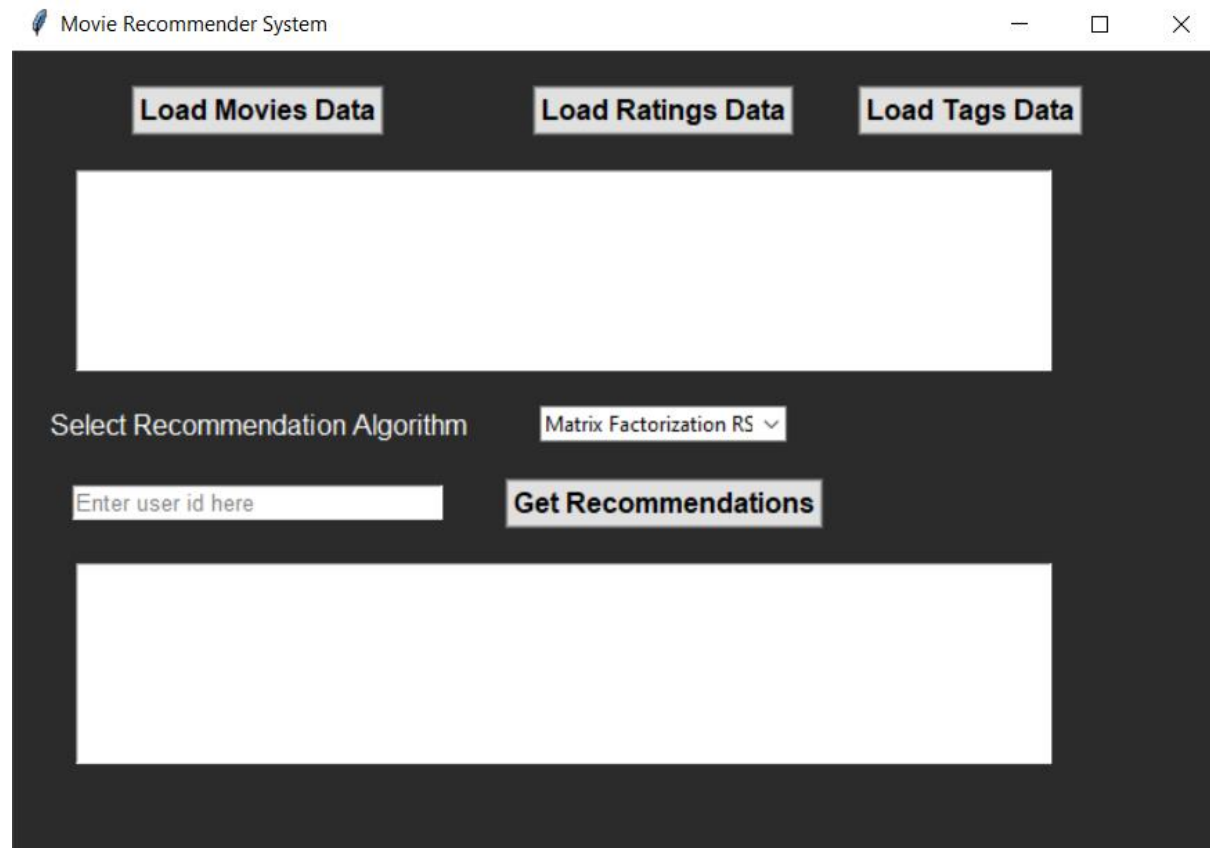
- **Content-Based Filtering:** Extracts features from movies and recommends similar ones.
- **Matrix Factorization with SVD:** Trains the SVD model to learn latent factors of users and movies.
- **Naïve Collaborative Filtering:** Uses heuristics like popularity or average ratings to recommend movies.

### User Interface Development

- Develop a user-friendly interface using the Tkinter library.
- Allow users to interactively select recommendation algorithms and view suggested movies.

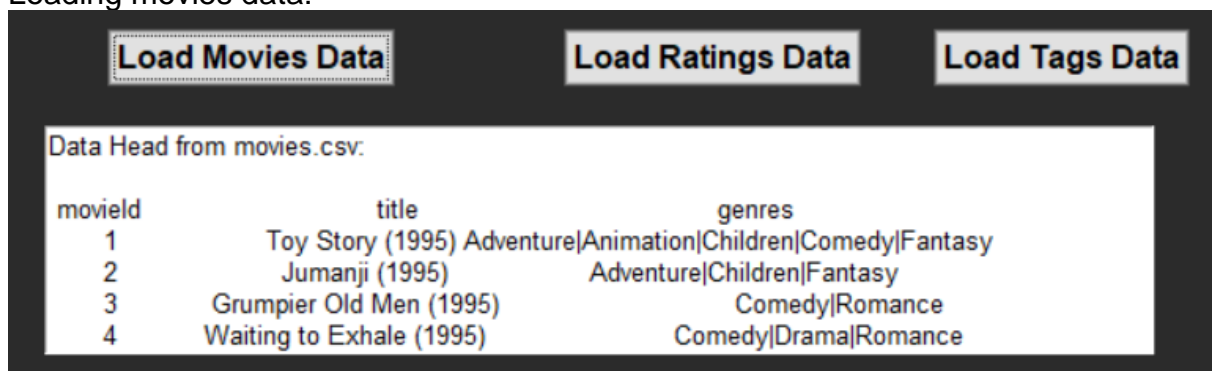
## Graphical User Interface (GUI):

The main page looks like this:

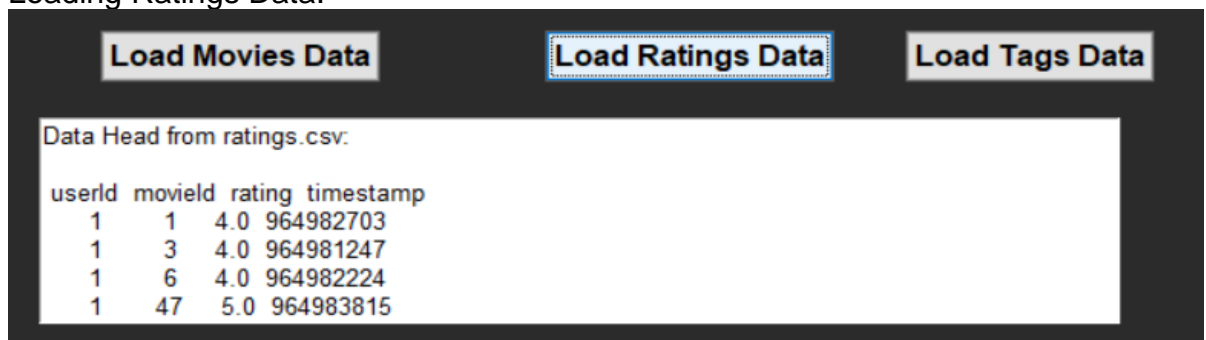


Loading the datasets:

1) Loading movies data:



2) Loading Ratings Data:



3) Loading Tags Data:

Load Movies Data

Load Ratings Data

Load Tags Data

Data Head from tags.csv:

userId	movieId	tag	timestamp
2	60756	funny	1445714994
2	60756	Highly quotable	1445714996
2	60756	will ferrell	1445714992
2	89774	Boxing story	1445715207

Running the Recommendations:

1) Matrix Factorization Recommendations (e.g. for user 37):

Select Recommendation Algorithm

Matrix Factorization RS

37

Get Recommendations

Lawrence of Arabia (1962)  
Bridge on the River Kwai, The (1957)  
Patton (1970)  
Cool Hand Luke (1967)  
Life Is Beautiful (La Vita è bella) (1997)

2) Content Based Recommendation System (e.g. for user 37):

Select Recommendation Algorithm

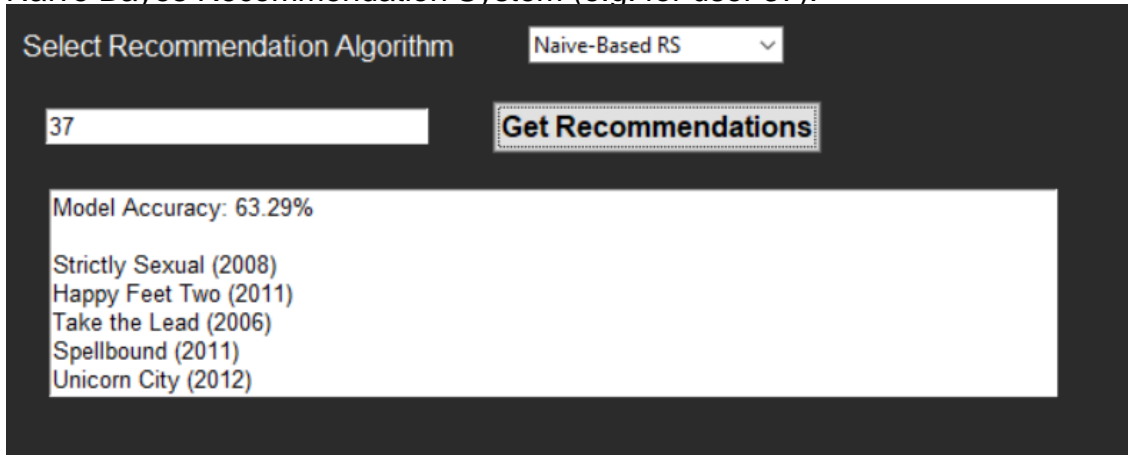
Content-Based RS

37

Get Recommendations

Ox-Bow Incident, The (1943)  
My Family (1995)  
Nights of Cabiria (Notti di Cabiria, Le) (1957)  
Celebration, The (Festen) (1998)  
Letter to Three Wives, A (1949)

### 3) Naïve Bayes Recommendation System (e.g. for user 37):



The screenshot shows a Tkinter-based interface for a Naïve Bayes Recommendation System. At the top, there is a label 'Select Recommendation Algorithm' followed by a dropdown menu currently set to 'Naive-Based RS'. Below this, there is a text input field containing the number '37' and a button labeled 'Get Recommendations'. The output area below the button displays the 'Model Accuracy: 63.29%' and a list of recommended movies: 'Strictly Sexual (2008)', 'Happy Feet Two (2011)', 'Take the Lead (2006)', 'Spellbound (2011)', and 'Unicorn City (2012)'.

Model Accuracy
63.29%

Recommended Movies
Strictly Sexual (2008)
Happy Feet Two (2011)
Take the Lead (2006)
Spellbound (2011)
Unicorn City (2012)

## Conclusion

This project successfully integrates three distinct recommendation algorithms into a Tkinter-based interface, offering users a flexible and personalized movie recommendation system. By allowing users to choose from different recommendation strategies, the system enhances the user experience and offers a tailored movie discovery experience.

## Future Improvements

- **Hybrid Models:** Combine different algorithms for improved accuracy.
- **Real-Time Updates:** Allow the system to learn from user feedback in real-time.
- **Scalability:** Improve scalability to handle large datasets efficiently.