# CSL2050-Pattern Recognition and Machine Learning

## Project Report

April 21, 2024

**Abstract**

With the vast amount of movies available across various platforms, users often face the challenge of selecting movies that align with their individual preferences and interests.Sometimes the case is like user watched some movie and liked that movie and he/she wants to watch some similar kind of movie but does not specifically know the name of the movie in this case movie recommender system can be helpful for them.For implementing this recommendation system we can make use of genre and some comments given by user which are incorporated as tags in our dataset for recommending movies.

# Contents

# 1 Introduction

## 1.1 Dataset Description

The provided dataset has three files- links.csv, movies.csv, ratings.csv and tags.csv.This dataset comprises 100,836 ratings and 3,683 tag applications across 9,742 movies. Movie ids are consistent between ratings.csv, tags.csv,movies.csv, and links.csv.
Dataset link:https://www. kaggle.com/datasets/shubhammehta21/movie-lens-small-latest-dataset

## 1.2 Dataset Components

Overview of Dataset Components:
**Movies.csv**

- The dataset contains information about 9,742 movies
- Each movie is identified by a unique movie ID.
- Movie titles are accompanied by the year of release in parentheses, providing temporal context.
- Genres are provided for each movie, representing its thematic categorization. Genres cover a wide range of categories, including Action, Adventure, Comedy, Drama, Fantasy, Horror, and more.
- Each line of this file has the following format: movieId,title,genres

**Ratings.csv**

- Ratings are made on a 5-star scale
- Each rating entry includes:
- User ID: Anonymized identifier for the user who provided the rating.
- Movie ID: Unique identifier for the movie being rated.
- Rating: The numerical rating given by the user, ranging from 0.5 stars to 5.0 stars.
- Timestamp: Indicates when the rating was provided, measured in seconds.
- Each line of this file represents one rating of one movie by one user, and has the following format: userId,movieId,rating,timestamp

**Tags.csv**

- Tags provide user-generated information about movies, often in the form of keywords or short phrases.
- Each tag entry comprises:
- User ID: Anonymized identifier for the user who applied the tag.
- Movie ID: Unique identifier for the movie associated with the tag.
- Tag: The descriptive keyword or phrase provided by the user.
- Timestamp: Indicates when the tag was applied, measured in seconds
- Tags serve as user-contributed annotations, enriching the dataset with contextual information.
- Each line of this file represents one tag applied to one movie by one user, and has the following format:userId,movieId,tag,timestamp

**Links.csv**

- The "links.csv" file provides identifiers that facilitate linking to external sources of movie data.
- Each link entry includes:
- movieId: An identifier for movies used by MovieLens. This ID corresponds to the movie's entry on the MovieLens website.
- imdbId: An identifier for movies used by the Internet Movie Database (IMDb). This ID enables linking to the movie's IMDb page.
- tmdbId: An identifier for movies used by The Movie Database (TMDb). This ID allows linking to the movie's TMDb page.
- Each line of this file represents one movie, and has the following format:movieId,imdbId,tmdbId

# 2 Data Preprocessing

## 2.1 Data Loading and Overview

### 2.1.1 Reading CSV Files

Using the `pd.read_csv()` function from the pandas library, the CSV files are read into pandas DataFrames. First few lines of all the loaded csv files into dataframes are printed so as to get an overview of the dataset.

### 2.1.2 Exploratory Data Analysis

Next,the code checks for missing values in each DataFrame, no missing values are found in movies dataframe,tags dataframe,ratings data frame,however for links dataframe tmbdId column has 8 missing values and we are neglecting these missing values as they do not affect our recommender system because our prediction model mostly predicts on the basis of tags given by user to the movies and genres of the movies.

## 2.2 Data Cleaning

### 2.2.1 Selecting relevant columns

Selecting only the relevant columns : 'userId', 'movieId', 'tag' from the tags DataFrame and assigning the resulting DataFrame back to tags.We have dropped 'timestamp' column from tags dataframe as when user gave tag to the movie does not affect our prediction model For the similar reason 'timestamp' column from ratings dataframe is also dropped.

### 2.2.2 Parsing Genres

In the movies dataframe each genre is separated by a `|` character eg: Action|Adventure|Sci-Fi ,our code uses .split(`|`) which splits the string into a list of genres.This operation converts a string like Action|Adventure|Sci-Fi into a list `[Action, Adventure, Sci-Fi]` for each movie.This allows for easier processing as it becomes easier to work with them in various data manipulation and analysis tasks.

### 2.2.3 Removing Whitespaces

Extraneous whitespace character from the tag column of the tags DataFrame is removed. This ensures that similar tags are treated identically. For example, `"action"` and `"action "` (with an extra space) would be considered the same tag after removing extra spaces and also words like "way too long ", "black hole" will become `"waytoolong"`, `"backhole"` and this will ensure that string matching operations, such as searching for specific tags in dataset, are more effective.

#### 2.2.4 Combining Tags

In the tags dataframe many users have given tags to the same movie. We have combined the tags for each movie, The result is stored in the `combined_tag` DataFrame.By consolidating all tags associated with a particular movie, we can capture a broader range of opinions, preferences, and descriptive information provided by different users,and this will help in recommending movies.

#### 2.2.5 Creating Description Column

A new dataframe `merged_df` is created by merging the movie information from the movies DataFrame with the combined tags from the `combined_tag` DataFrame based on the movieId.
A new column named description in the `merged_df` DataFrame by combining the genres and tags for each movie.and the genres and tag columns are dropped from the `merged_df` DataFrame since their information has been combined into the description column.
The TMDb IDs from the links DataFrame are added to the `merged_df`DataFrame under the column name tmdbId. Adding TMDb IDs to the merged DataFrame facilitates the retrieval of additional movie information, such as posters, from the TMDb API.
`merged_df` is our final dataframe after applying all the data cleaning which we will be using in our final recommender system and it looks like-

| | movieId | title | description | tmdbId |
|---|---|---|---|---|
| 0 | 1 | Toy Story (1995) | adventure animation children comedy fantasy pi... | 862.0 |
| 1 | 2 | Jumanji (1995) | adventure children fantasy fantasy magicboardg... | 8844.0 |
| 2 | 3 | Grumpier Old Men (1995) | comedy romance moldy old | 15602.0 |
| 3 | 4 | Waiting to Exhale (1995) | comedy drama romance | 31357.0 |
| 4 | 5 | Father of the Bride Part II (1995) | comedy pregnancy remake | 11862.0 |

Figure 1: `merged_df`

### 2.3 Preprocessing on `merged_df` DataFrame

#### 2.3.1 Lowercasing Text

All text in the `"description"` column of the DataFrame is converted to lowercase ,this ensures consistency and avoids case sensitivity in subsequent text processing steps.

#### 2.3.2 Removing Repetitive Words

`merged_df[description]`.apply( `remove_repetitive_words`): This line applies the `remove_repetitive_words` function to each description in the `"description"` column of the DataFrame. It removes repetitive words, ensuring that each word appears only once in the description

#### 2.3.3 Stemming with Porter Stemmer

Stemming is applied to each word of the description column using the Porter Stemmer algorithm from NLTK .Stemming reduces words to their root or base form, which helps in standardizing the text as this reduces the variations in word forms (e.g., plurals, verb tenses), making it easier to compare and match words during the recommendation process.Words with the same stem are likely related in meaning, even if they have different inflections. This allows the recommender system to identify movies with similar themes or topics more effectively.

## 3 Approaches Tried

### 3.1 Agglomerative Clustering

#### 3.1.1 Breakdown of steps

- By grouping together movies that have similar descriptions or characteristics. This grouping helps identify clusters of movies that share common themes, genres, or content.

- Agglomerative clustering builds a hierarchy of clusters by iteratively merging pairs of clusters that are closest to each other until all data points belong to a single cluster.
- Once the clustering is done, each movie belongs to a specific cluster. Recommendations are then made by suggesting other movies within the same cluster as the input movie

#### 3.1.2 Problems encountered

It is difficult to determine the right number of clusters in agglomerative clustering. While the clustering process produces clusters in a hierarchical manner, it is difficult to choose the granularity level to select the right number of clusters.
Agglomerative clustering is not suitable for classifying data with overlapping clusters. It doesn't take into account the relationship between the clusters.
There is no guarantee that the resulting clusters in agglomerative clustering are meaningful. The algorithm does not provide any probabilistic guarantees about the quality of the clusters.

### 3.2 DBSCAN

#### 3.2.1 Breakdown of steps

- The movie descriptions are vectorized using the CountVectorizer from scikit-learn. This step converts text descriptions into numerical feature vectors.
- DBSCAN is applied to cluster the movies based on their feature vectors
- For recommending movies, the model finds the cluster label of the input movie, retrieves other movies in the same cluster, and prints the titles of recommended movies

#### 3.2.2 Problems Encountered

The quality of DBSCAN depends on the distance measure.The most common distance metric used is Euclidean distance. Especially for high-dimensional data, this metric renderes almost useless due to the so-called "Curse of dimensionality".For making the movie recommendation the description column had very high dimensional data which led to problem in the recommendation.
DBSCAN is not entirely deterministic: border points that are reachable from more than one cluster can be part of either cluster, depending on the order the data are processed. When the vectors that were generated using the vectoriser were given to the model several points in the vector space were there which were a part of more than one cluster thus making some ambiguous predictions.

### 3.3 Using K-means Clustering

#### 3.3.1 Breakdown of steps

- Total number of unique genres which are there in the genre column of movies dataframe are found.
- The categorical genre information is converted into a binary matrix where each genre is represented as a binary feature.
- The number of clusters for k-means clustering is set to the total number of unique genres.
- The k-means model is fitted to the binary genre matrix
- For a given movie title first the cluster label of the given movie is found and then it retrieves other movies in the same cluster. Top similar movies are recommended based on genre similarity within the same cluster.
- Then the cosine similarity between the given movie and other movies in the cluster based on genre features is calculated.
- Finally, it recommends the top N similar movies based on similarity scores

### 3.3.2 Problems Encountered

Implementing K-Means clustering based solely on movie genres may lead to several limitations and differences in performance compared to the CountVectorizer method that incorporates both genres and user-provided tags.Excluding user tags neglects valuable information provided directly by users. Tags represent user-generated content and preferences, offering insights into specific aspects of movies that users find noteworthy or appealing.
The challenge with incorporating user tags directly into the K-Means clustering process lies in the categorical nature of tags. Unlike genres, which typically belong to a predefined set with a limited number of possible values (e.g., action, comedy, drama), user tags can be highly variable and may include a wide range of descriptors, keywords, or phrases that are not necessarily predefined or standardized.

## 3.4 Using CountVectorizer and Cosine Similarity

- Text data in the description column of `merged_df` is converted to numerical vectors using the countVectorizer class.
- CountVectorizer is then fitted to the `"description"` column of the `merged_df` DataFrame and it transforms the text data into a matrix of token counts represented as a NumPy array.
- Each row of the vectors array corresponds to a movie in the dataset.The value at vectors`[i,j]` represents the frequency of the `j-th` word in the vocabulary in the `i-th` movie description. In other words, it indicates how many times the `j-th` word appears in the `i-th` movie's description.
- The cosine similarity between all pairs of movie descriptions represented as vectors is calculated.
- For a given movie its cosine similarity score is sorted in descending order and top 5 movies are selected for recommendation.

Finally we took this approach to implement our recommender system as this approach solely focuses on the textual content of the movies. This means it can provide recommendations even for new or less-rated movies, making it more robust in scenarios with sparse or incomplete user data.

# 4 Website Development for Enhanced User Experience

For providing users with a seamless and intuitive experience, we've developed a dedicated website to serve as the gateway to our movie recommendation system. This platform offers users a user-friendly interface designed to enhance their interaction with our recommendation engine.
Users can browse through our extensive movie database and select their preferred movie. Upon selecting a movie, our recommendation engine analyzes its characteristics and generates a curated list of five recommended movies. These recommendations are carefully selected based on similarities in genre, theme, or other relevant factors.
Link to the website:https://movierecommender-prml.streamlit.app/

# 5 Summary

In this project, we developed a movie recommendation system to assist users in discovering movies that align with their preferences and interests. The system utilizes a dataset comprising movie information, including genres, user ratings, and user-generated tags. We explored various approaches to recommend movies, including agglomerative clustering, DBSCAN, and K-means clustering. After preprocessing the dataset and cleaning the data, we leveraged the descriptive information provided by users in the form of tags and combined it with movie genres to create a comprehensive description for each movie. We then applied different clustering techniques to group similar movies together and recommend them to users.
We encountered challenges with each clustering approach, such as determining the optimal number of clusters and handling high-dimensional data. But finally with the help of count vectorizer and cosine similarity, we were able to generate meaningful recommendations based on movie similarities.
To enhance the user experience, we developed a dedicated website where users can interact with the

recommendation system seamlessly. This platform allows users to explore a vast collection of movies and get recommendation.
Overall, our movie recommendation system provides users with a convenient and efficient way to discover new movies, enhancing their overall movie-watching experience.

# A Contribution of group members

1. Sahil Sharma: Analysed dataset,tried approaches to implement model ,helped in building website
2. Shruti Chaudhary: Data preprocessing,tried approaches for implementing model,project report,helped in building website.
3. Khyati Sisodia: Made project page,helped in implementing model,helped in making spotlight video
4. Kandrathi Sai Aiswarya: Made ppt for making spotlight video and helped in making spotlight video
5. Mohit Jingar:Helped in making ppt for the spotlight video.
6. Yesha Shah: Hepled in making report and project page.
7. Kamal Kumar: Helped in making spotlight video.

# B Links

Project Page link:https://shruti03052.github.io/

Link to the notebook:https://colab.research.google.com/drive/1iHfow9HRGKEW_41QfpSHgUTp8UxmUxSE?usp=sharing

Link to the github repository:https://github.com/Shruti03052/PRML_Project_Spring_2024