Project title: Movie Recommendation using KNN

Team members:

1. Abhishek Kulkarni (A20516035)
2. Jhanavi Dave (A20515346)

Description of the problem:

With the evolution in consumption of digital media, consumers are presented with a huge range of choices for their entertainment. This has proved to be a privilege but has also led to a decision-making paradox, overwhelming the user with a variety of options. The quality and similarity between the choices differ greatly thus, challenging the ability to choose content based on their unique preferences and interests.

Orthodox methods of recommendations for movies are based only on the type of genre or referring to the periodical hits. The data generated from ratings from movies, keywords from movie descriptions, and cast and crew, if utilized appropriately, can notably and accurately perform the recommendations and improve the user's experience while enjoying one of their favorite forms of media - movies.

This project aims to build a movie recommendation system based on the K-Nearest Neighbors (KNN) algorithm, and using a comprehensive database of ~5000 movies, providing recommendations tailored to similar movie preferences, helping users discover new movies they are likely to enjoy, thereby improving content discovery and user satisfaction. The KNN algorithm is trained on the preprocessed dataset, to learn relationships between movies based on their features and user ratings. KNN algorithm is a popular choice for recommendation systems as it relies on the similarity between items (movies in our scenario) to make predictions. Additionally, this recommendation system can also enable businesses to gain valuable insights into user behavior and preferences, which can inform content offerings and business decisions, if deployed at an enterprise level with a significantly larger dataset (preferably containing almost all the movies).

Data Summary:

The data used in this project is sourced from Kaggle - TMDB 5000 Movie Dataset, which has expansive movies-related data. This data is two-fold:

The first file tmdb_5000_credits.csv contains the cast and crew of 4813 unique movies, identified with movie names and movie IDs.
Second, the tmdb_5000_movies.csv contains detailed information about the movies, from which vote_average, genres, titles, keywords, and other movie-related information.

Some key attributes present in the dataset:
Title: The title of the movie.
Genres: The genres associated with the movie.
Language: The primary language of the movie.
Cast: Information about the actors, directors, etc.
Crew: Information about crew members involved in the movie.
Keywords: Array of keywords from movie description.
Vote_average: average user rating for the movie.

The columns present inconsistencies with missing values and array values(for example, the column keyword contains information in the array format) which will be ironed out in data processing.

Progress so Far:

With the literature review of various existing papers and articles, the lack of parameters for recommendation of movies that could be liked by the user based on their previous interaction with media, and the lack of real-time data was discovered. The stated CSV files have various data parameters like the original language it was made in, the original title it was released with, an overview of the plot, popularity, production companies, countries it was produced in, the release date, revenue collected at the box office, runtime of the entire movie, the languages used in the dialogues, status of the movie release, tagline, title, average vote and total vote count, cast and crew for each aspect of the movie. In the current state of the system, the first step is data preprocessing.

The model processes the data, converts JSON strings to lists and strings, extracts relevant information, and cleans up the data. In the next step, the model engineers the features by extracting various genres, cast, director, keywords, average vote, etc. information from the data and converting these categorical features into binary representations for each movie, describing which parameter is related to the movie.

The model then generates visualizations to explore the data, such as bar plots for top genres, actors, and directors in the form of graphs and bar charts. A word cloud is then created based on movie keywords, providing a visual representation of frequently occurring words. A similarity calculator function is implemented to calculate the similarity between two movies based on their genres, cast, director, keywords, and vote average.

Future Steps:

Including other parameters like user preferences and behaviors, like ratings, actors & actresses, runtimes, popularity, and explicit feedback from the dataset, would make the recommendations more personalized and accurate. This can be achieved through collaborative filtering techniques.

The model will also rely on user preferences and ratings from other users along with other parameters, with a neighbor calculating function that finds the K-Nearest Neighbors of a given movie based on the similarity between the movies in the dataset. The movies with the smallest cosine distances as the nearest neighbors will then be selected and displayed as the recommended movies.

Using cosine distance which is commonly implemented in KNN the similarity between vectors will be calculated. Since the accuracy of the cosine angle and the equidistance of parameters remains almost the same for the cosine distance, this can prove as an optimized model. The model will then finally predict the name of the recommended movies similar to the given input, by matching the title to the movie name in the dataset, then finding the K-most similar movies using the similarity function, and then recommending the most similar movies with their details.

Combining content-based filtering with collaborative filtering or other recommendation techniques (hybrid systems) often leads to a robust and accurate recommendation model. This approach might leverage the strengths of multiple recommendation methods to mitigate their limitations. Recommendations using additional features, such as movie release dates, language, and user-generated tags, could also provide more diverse and informative attributes for recommendation.

Conducting thorough evaluation and testing of the recommendation system using techniques such as cross-validation and testing to measure its performance and effectiveness in real-world scenarios.