Name: Ayus Mukherjee (801417497)

Subject: Data Mining (ITCS-6162)

Title: Movie Recommendation System Report

1. Introduction Movie recommendation systems are essential tools in modern digital platforms that help users discover content based on their preferences. As the volume of available content grows, users face difficulty in selecting relevant movies, and recommendation systems provide a personalized experience to enhance engagement and satisfaction. There are several approaches to movie recommendation:

- **User-based collaborative filtering**, which identifies users with similar tastes and recommends what similar users liked.
- **Item-based collaborative filtering**, which recommends movies similar to the ones a user has previously enjoyed.
- **Random-walk-based algorithms** like the Pixie method, which explore a user-item interaction graph to find contextually relevant content through probabilistic traversal.

2. Dataset Description This project uses the **MovieLens 100K** dataset, a well-known benchmark for recommender system experiments. The dataset includes:

- **943 users**, **1,682 movies**, and **100,000 ratings**.
- Features used: `user_id`, `movie_id`, `rating`, and movie `title`.
- Preprocessing steps included:
    - Merging `ratings` and `movies` data to include movie titles.
    - Aggregating ratings to average per user-movie pair.
    - Normalizing ratings by subtracting the user's mean rating to reduce bias.
    - Constructing a user-movie bipartite graph for graph-based recommendation.

3. Methodology Three recommendation approaches were implemented:

- **User-based collaborative filtering**: A user-movie matrix was created, and cosine similarity was used to compute user-user similarities. For a given user, unrated movies were scored using weighted averages of ratings from similar users.
- **Item-based collaborative filtering**: The transposed user-movie matrix was used to compute item-item cosine similarity. Recommendations were based on items similar

to a given movie.

- **Random-walk-based Pixie algorithm**: A bipartite graph was built with users and movies as nodes. Random walks were performed starting from a user or movie node, and movies were ranked by how frequently they were visited. This method uncovers indirect and contextual relationships, offering personalized and sometimes serendipitous recommendations.

4. Implementation Details

- To implement **user-based collaborative filtering**, the dataset was transformed into a user-movie matrix using the `pivot()` function from pandas, with `user_id` as the index and `movie_id` as columns. Missing values, representing unrated movies, were filled with 0. Cosine similarity was calculated between user vectors using `scikit-learn`'s `cosine_similarity()` to create a user-user similarity matrix. The recommendation function ranks movies not rated by the user by computing a weighted average of ratings from similar users.

- In **item-based collaborative filtering**, the same matrix was transposed so that each row represented a movie and each column a user. Cosine similarity was again used to compute similarity between movies based on user ratings. For any given movie, the top-N most similar movies were retrieved by ranking those with the highest similarity scores. The function returned these as the recommended results.

- For the **Pixie-inspired graph-based recommendation**, an undirected bipartite graph was constructed using an adjacency list structure. Each user was connected to the movies they had rated, and each movie was connected back to all users who rated it. A random walk algorithm was implemented where a walk starts from a user or a movie node and randomly visits neighboring nodes for a specified number of steps. A dictionary was used to count how frequently each movie node was visited during the walk. These visit counts were then sorted, and the top-ranked movies were recommended to the user.

- The random-walk function supported both user-based and movie-based initiation. For movie-based walks, the input movie title was converted into its corresponding ID, and the walk started from that movie node. This allowed exploration of movies frequently co-rated by similar users. The method was particularly effective at identifying hidden or niche connections in the data that may not be obvious from direct similarity scores alone.

5. Results and Evaluation

- **Example Outputs:**
    - For **user-based collaborative filtering**, user 6 received recommendations such as "Jaws (1975)", "Apt Pupil (1998)", and "Star Trek III: The Search for

Spock (1984)", which were determined by analyzing similar users and their highly rated movies.

- For **item-based collaborative filtering**, when queried with "Top Gun (1986)", the system recommended related titles like "Empire Strikes Back (1980)", "Indiana Jones and the Last Crusade (1989)", and "Speed (1994)"—all action-packed or adventure-themed, matching the genre and viewer appeal.

- For the **Pixie random-walk method**, user 1 received recommendations such as "Desperate Measures (1998)", "Garden of Finzi-Contini, The (1970)", and "Santa with Muscles (1996)". Starting from "Jurassic Park (1993)", the system suggested diverse options like "Wedding Singer, The (1998)" and "Old Yeller (1957)", illustrating serendipitous discovery.

- **Accuracy and Usefulness Comparison:**

  - **User-based filtering** is particularly useful when users have a sufficient history of interactions, as it leverages the wisdom of similar users. However, its accuracy diminishes when user preferences are sparse.

  - **Item-based filtering** tends to perform better in sparse environments and offers more stable results, especially for popular movies, as item similarities change less frequently over time.

  - **Random-walk-based recommendations** provide more exploratory results and excel in discovering less obvious connections. They are useful for diversifying recommendations, especially in discovery-driven platforms, but may sometimes produce unpredictable or less interpretable results.

- **Limitations:**

  - The user- and item-based approaches are limited by **cold-start problems**, where new users or items lack enough data to compute similarities.

  - The **Pixie approach**, while capable of surfacing diverse content, is **non-deterministic** and can yield different results in each run. Additionally, the lack of edge weights or content/contextual metadata limits the depth of personalization.

  - None of the current models incorporate temporal dynamics, user demographics, or multi-criteria ratings, which could significantly enhance recommendation quality.

6. Conclusion This project demonstrated how different collaborative filtering and graph-based techniques can be applied to movie recommendations. User-based and item-based filtering offer strong baselines, while Pixie-inspired random walks provide scalable, dynamic recommendations. Potential improvements include combining methods in

a hybrid model, adding temporal/user-demographic features, and evaluating with metrics like precision and recall. These algorithms have strong real-world applications in platforms like Netflix, Spotify, and Pinterest for improving content discovery and user retention.