Name: Pavithra Selvan
ITIS 6162 – Data Mining

Data Mining Programming Assignment
Movie Recommendation System

Table of Contents

Github Repository:

https://github.com/PavithraSelvan169/datamining_recommendersystem

Objective

This assignment involves building a movie recommendation system using three major approaches:

- Data Exploration and Preprocessing
- Collaborative Filtering (User- and Item-Based)
- Pixie-Inspired Graph-Based Recommendation System
    - Unweighted model for learning purpose
    - Weighted model for final submission

Dataset Overview

u.data: Main interaction data with user_id, movie_id, rating, and timestamp

Name: Pavithra Selvan
ITIS 6162 – Data Mining

u.item: Metadata for movies including title, release year
u.user: User demographic information (age, gender, occupation)

Part 1: Data Exploration & Cleaning

Goals:
Understand data structure

Identify useful fields for recommendation tasks

Load and inspect the datasets using raw Python and pandas

Files and Formats:
u.data: Tab-separated, no header

u.item and u.user: Pipe-separated

Tasks Performed:
Opened and printed sample lines from each file
Used pandas.read_csv() with appropriate separators
Retained relevant columns only (e.g., titles, IDs, ratings)

Part 2: Collaborative Filtering

Method for similarity: sklearn.metrics.pairwise.cosine_similarity

User-Based Filtering:

It is a recommendation system technique that predicts the items a user might like by identifying the similar users with tastes and behaviours.
User similarity is calculated using methods like Pearson correlation which measures the correlation between their rating vectors. Users with high similarity scores are considered to be more alike in terms of their tastes.

Steps involved:

- Constructed a user-movie rating matrix
- Calculated cosine similarity between users
- Found top-N similar users for a given user
- Recommended movies that similar users liked but the target user hasn't seen

Name: Pavithra Selvan
ITIS 6162 – Data Mining

Output:

User Based:

Step 4: Return the Final Recommendation List

```
recommend_movies_for_user(7, num = 5)
```

Ranking	Movie Name
1	2001: A Space Odyssey (1968)
2	Apartment, The (1960)
3	Legends of the Fall (1994)
4	Third Man, The (1949)
5	Fly Away Home (1996)

Movie Based:

Step 3: Return the Final Recommendation List

```
recommend_movies("Jurassic Park (1993)", num=5)
```

Ranking	Movie Name
1	On Golden Pond (1981)
2	Wyatt Earp (1994)
3	Brazil (1985)
4	Princess Bride, The (1987)
5	M*A*S*H (1970)

Item-Based Filtering:

It is a recommendation system that uses the similarity between items to recommend products or content to users.
Item similarity is calculated using the similarity between all pairs of items in the dataset. It calculates the similarity using the cosine similarity, Pearson correlation based on user's ratings or interactions. It identifies the similar items and based on the similarity score it predicts how much the user might like this item and recommend the items with highest predicted scores or ratings.

Steps involved:

- Transposed the matrix to compute item-item similarity
- Recommended movies similar to ones the user already rated highly

Part 3: Pixie-Inspired Graph-Based Recommendation

The Pixie-Inspired system is one of the Graph-based recommendation systems. They are generally very effective for real-world recommendation tasks.

Its high effectiveness is because it allows for exploring the connection between items that might not be directly linked enabling the discovery of relevant recommendations.
The algorithm is more effective in its ability to model user behaviour and item relationship within a large graph leading to a more accurate, fast and personalized recommendations.

Name: Pavithra Selvan
ITIS 6162 – Data Mining

Graph Construction:

Steps:

1. Created a bipartite graph of users and movies
2. Edges represent user ratings (optionally weighted)
3. Random Walk (Unweighted & Weighted):
   Unweighted version:
   Each step picks a random neighbor
   Weighted version:
   Probability of choosing a neighbor is based on rating strength (edge weight)
4. User based recommendation

Step 3: Implement User-Based Recommendation

```
def weighted_pixie_recommend_userbased(user_id, walk_length=5000, num=10):


    if user_id in weighted_graph:
        visited_movies=weighted_graph[user_id]
        max_weight = max(dict(visited_movies).values())
        top_nodes = [node for node, weight in dict(visited_movies).items() if weight == max_weight]

        # randomly pick a movie to start the walk from:
        current_node=random.choice(top_nodes)
        visit_count={movie:0 for movie in (dict(visited_movies)).keys()}

        for _ in range(walk_length):
            connected_nodes=weighted_graph[current_node]
            if not connected_nodes:
                break

            max_weight = max(dict(connected_nodes).values())
            top_nodes = [node for node, weight in dict(connected_nodes).items() if weight == max_weight]
            next_node=random.choice(top_nodes)

            if next_node in visit_count:
                visit_count[next_node]+=1
            current_node=next_node
        ranked_movies=sorted(visit_count,key=visit_count.get,reverse=True)
        # ranked_movies = [movie for movie in ranked_movies if movie in u_item_df['movie_id'].values]

    else:
        return "User not found in the dataset."

    result_df=pd.DataFrame({'Ranking':range(1,num+1),'Movie Name':ranked_movies[:num]})
    result_df.set_index('Ranking',inplace=True)
    return result_df


# call the function for any user_id
weighted_pixie_recommend_userbased(1, walk_length=50, num=5)
```

Name: Pavithra Selvan
ITIS 6162 – Data Mining

5. Movie based recommendation

```
def weighted_pixie_recommend_moviebased(movie_name, walk_length=5000, num=5):

    if movie_name not in u_item_df['title'].values:
        return "Movie not found in the dataset."
    else:


        # randomly pick a movie to start the walk from:
        current_node=movie_name
        visit_count={movie:0 for movie in u_item_df['title']}

        for _ in range(walk_length):
            connected_nodes=weighted_graph[current_node]
            if not connected_nodes:
                break

            max_weight = max(dict(connected_nodes).values())
            top_nodes = [node for node, weight in dict(connected_nodes).items() if weight == max_weight]
            next_node=random.choice(top_nodes)

            if next_node in visit_count:
                visit_count[next_node]+=1
            current_node=next_node
        ranked_movies=sorted(visit_count,key=visit_count.get,reverse=True)
        # ranked_movies = [movie for movie in ranked_movies if movie in u_item_df['movie_id'].values]

        result_df=pd.DataFrame({'Ranking':range(1,num+1),'Movie Name':ranked_movies[:num]})
        result_df.set_index('Ranking',inplace=True)
        return result_df


# call the function for any user_id
weighted_pixie_recommend_moviebased('Young Frankenstein (1974)', walk_length=50, num=5)
```

Key Concepts:

Random walks simulate exploratory browsing behavior
Frequent visits to movies during walks indicate higher recommendation scores

Output:

Recommendation Results for Graph based recommendation

User-Based Pixie Recommendation:

User-Based Recommendation

```
# call the function for any user_id
weighted_pixie_recommend_userbased(1, walk_length=50, num=5)
```

Ranking	Movie Name
1	Star Wars (1977)
2	Terminator, The (1984)
3	Blues Brothers, The (1980)
4	Swingers (1996)
5	Usual Suspects, The (1995)

Movie-Based Recommendation:

Movie-Based Recommendation

```
# call the function for any user_id
weighted_pixie_recommend_moviebased('Back to the Future (1985)', walk_length = 50, num=5)
```

Ranking	Movie Name
1	Schindler's List (1993)
2	2001: A Space Odyssey (1968)
3	Friday (1995)
4	Ace Ventura: Pet Detective (1994)
5	Home Alone (1990)

Name: Pavithra Selvan
ITIS 6162 – Data Mining

Takeaways

1. Collaborative Filtering is simple and effective, but limited to users/movies with enough ratings
2. Pixie-style random walks offer personalization and diversity by leveraging graph structure
3. Weighted walks improve quality by emphasizing strong user preferences
4. Other findings:
    a. `walk_length` is a parameter that says how many steps the random walk happens. Higher the walk_length, the more likely it finds some hidden gems (rarely watched movies). Lower walk_length could lead to popular movies in the results, that most people watch
    b. `num` is another variable that is used to limit the number of recommendations