Assignment Report: Recommendation System

# 1. Introduction

The objective of this assignment is to build a **recommendation system** for anime data. Recommendation systems are widely used in platforms such as Netflix, Amazon, and YouTube to suggest items to users based on their preferences or the behavior of similar users.

The dataset (anime.csv) contains details about various anime, including their IDs, names, genres, types, and ratings. The goal is to design a system that can suggest relevant anime to users by analyzing ratings and similarities.

# 2. Data Understanding

The dataset was loaded using Pandas:

import pandas as pd

df = pd.read\_csv("anime.csv")

# Dataset overview:

## **Number of records (rows):** Retrieved using df.shape

## Columns include:

* + anime\_id → Unique identifier for each anime
  + name → Name of the anime
  + genre → Genre(s) of the anime
  + type → Type of content (e.g., TV, Movie, OVA)
  + rating → User ratings for anime

# Data issues identified:

* Missing values were present in columns genre, type, and rating.
* These were handled using:
* df["genre"].fillna(0, inplace=True)
* df["type"].fillna(0, inplace=True)
* df["rating"].fillna(0, inplace=True)

# Distribution of Ratings:

A histogram was plotted (plt.hist(df["rating"])) to understand the distribution of ratings across anime.

# 3. Methodology

## Step 1: Creating the User-Item Matrix

A pivot table was generated:

user\_df = df.pivot(index='anime\_id', columns='name', values='rating')

This matrix has **anime IDs as rows** and **anime names as columns**, with ratings as the values. Missing values were filled with 0 to avoid null errors:

user\_df.fillna(0, inplace=True)

## Step 2: Computing User Similarity

To identify similar users, **Cosine Similarity** was computed using:

from sklearn.metrics import pairwise\_distances

user\_sim = 1 - pairwise\_distances(user\_df.values, metric='cosine')

* A similarity matrix (user\_sim) was generated.
* The diagonal was filled with 0 since a user is not compared with themselves:
* np.fill\_diagonal(user\_sim, 0)

## Step 3: Structuring the Similarity Matrix

The similarity matrix was converted into a DataFrame for readability:

user\_sim\_df = pd.DataFrame(user\_sim)

user\_sim\_df.index = df.anime\_id.unique()

user\_sim\_df.columns = df.anime\_id.unique()

## Step 4: Identifying Most Similar Users

The most similar users for each user were identified using:

user\_sim\_df.idxmax(axis=1)[0:10]

This provides the **closest matching anime IDs** for the first 10 entries.

## Step 5: Comparing Ratings of Similar Users

Example comparisons:

user\_6 = df[df['anime\_id']==6]

user\_168 = df[df['anime\_id']==168]

This allows direct comparison between anime with similar rating patterns.

# 4. Results & Analysis

1. **User similarity matrix** successfully identified the most similar users/anime based on cosine similarity.
2. **Top recommendations** can be extracted by finding highly similar users and suggesting anime they rated highly that the target user has not yet rated.
3. Data preprocessing (handling nulls) was crucial for building a stable model.

# 5. Conclusion

* The assignment implemented a **user-based collaborative filtering recommendation system** using anime ratings.
* Cosine similarity was effective for finding similar users and enabling recommendations.
* Key limitations:
  + Sparse rating matrix (many missing ratings).
  + Cold start problem for new users or anime with no ratings.

# Future Enhancements:

* Use **item-based collaborative filtering** (focusing on similarities between anime rather than users).
* Apply **matrix factorization techniques** (e.g., SVD) to handle sparsity better.

# Interview Questions:

## Can you explain the difference between user-based and item-based collaborative filtering?

### User-Based Collaborative Filtering (UBCF):

* Focus: Finds **similar users** based on their past ratings or behaviors.
* Process:
  + Identify users who are similar to the target user.
  + Recommend items that those similar users liked but the target user hasn’t consumed yet.
* Example: *If User A and User B like many of the same anime, and User B liked "Naruto," then "Naruto" can be recommended to User A.*
* Limitation: Struggles when the number of users is very large (scalability issue).

### Item-Based Collaborative Filtering (IBCF):

* Focus: Finds **similar items** instead of users.
* Process:
  + Look at items the user has rated.
  + Find items that are similar to those items.
  + Recommend these similar items to the user.
* Example: *If "Naruto" and "Bleach" are rated similarly by many users, then if a user liked "Naruto," they are likely to be recommended "Bleach."*
* Advantage: More stable and scalable because item relationships don’t change as frequently as user behavior.

### Key Difference:

* UBCF = recommends based on **similar users**.
* IBCF = recommends based on **similar items**.

## What is collaborative filtering, and how does it work?

**Definition:**  
Collaborative filtering is a technique used in recommendation systems that makes predictions about a user’s interests by **collecting preferences from many users**. The underlying assumption is that if users agreed in the past, they will agree again in the future.

### How it works:

1. **Data Collection** – Gather user interactions with items (ratings, likes, clicks, purchases).
2. **Similarity Calculation** – Measure similarity between users (user-based) or items (item-based) using metrics like **cosine similarity, Pearson correlation, or Jaccard index**.
3. **Prediction** – Predict ratings or preferences for items the user hasn’t interacted with.
4. **Recommendation** – Suggest top-N items with the highest predicted rating/likelihood.

### Types of Collaborative Filtering:

* **User-based CF** – Finds similar users.
* **Item-based CF** – Finds similar items.
* **Model-based CF** – Uses machine learning techniques (e.g., matrix factorization, SVD, deep learning) to learn latent features and make recommendations.