# Item-Based Collaborative Filtering Recommender System for Anime Streaming

Role: Data Analyst – Recommender Systems

Project Type: Personal / Freelance Simulation

Duration: 1 week

Tools Used: Python (Pandas, Scikit-learn), Jupyter Notebook, Matplotlib

## Background

Anime streaming platforms depend heavily on personalized content discovery to drive user engagement and retention. In this project, I simulated the role of a Data Analyst on a recommender systems team to build an item-based collaborative filtering (CF) engine using real-world anime consumption data.  
  
This case study was structured to mirror a Shopify-style product analytics task, focusing not only on technical model implementation but also on product thinking, performance tradeoffs, and business impact.

## Data Overview

The project used three primary datasets:

1. animelist.csv: 109M rows of user-anime interactions including user\_id, anime\_id, score, watching\_status, and watched\_episodes.

2. rating\_complete.csv: Subset of animelist.csv with 57M rows where users fully completed and rated anime (stronger signal for preferences).

3. anime.csv: Metadata for each anime: Name, Genres, Type, Episodes, Source, Rating, Popularity, etc.

## Problem Statement

Goal:  
Build an item-based CF model that accepts a user’s list of anime ratings (no user\_id, cold-start) and returns a list of at least 5 recommended anime titles.  
  
Requirements:  
- Output includes: MAL\_ID, Name, Score, Type, Source, and Synopsis.  
- Recommendation logic must rely on item similarity, not popularity.  
- Should be explainable and adaptable for integration into a product interface (e.g., onboarding screen or homepage).

## Approach

1. Data Preprocessing  
Filtered rating\_complete.csv to keep only rows where score > 0. Created a user-item rating matrix with anime titles as columns. Merged anime metadata from anime.csv for final output enrichment.

2. Item Similarity Matrix  
Computed item-item cosine similarity using the rating matrix. Normalized scores to avoid popularity bias. Used Scikit-learn’s cosine\_similarity with sparse matrices for performance.

3. Recommendation Logic  
Input: A user’s list of rated anime (no user\_id). For each anime in the input list, find top-N similar anime using similarity matrix. Weight similarity by user score. Aggregate and rank candidates by cumulative similarity scores.

4. Final Output  
Returned top 5 anime not present in the input list. For each, enriched with: MAL\_ID, Name, Score, Type, Source, and placeholder Synopsis.

## Sample Input (User Ratings)

[ {'anime\_id': 1, 'score': 9}, {'anime\_id': 5114, 'score': 8}, {'anime\_id': 32281, 'score': 10} ]

## Sample Output (Top 5 Recommendations)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| MAL\_ID | Name | Score | Type | Source | Synopsis |
| 9253 | Steins;Gate | 9.1 | TV | Visual Novel | ... (mocked) |
| 11061 | Hunter x Hunter (2011) | 9.0 | TV | Manga | ... (mocked) |
| 19815 | No Game No Life | 8.3 | TV | Light Novel | ... (mocked) |
| 1535 | Death Note | 8.7 | TV | Manga | ... (mocked) |
| 30276 | One Punch Man | 8.6 | TV | Web Manga | ... (mocked) |

## Results & Learnings

Explainability: The model can justify why a recommendation was made (e.g., users who loved Code Geass also rated Steins;Gate highly).  
Cold Start Workaround: Accepting a pseudo-user input with no historical ID allows cold-start onboarding recommendations.  
Data Volume: Scaling this system beyond prototype would require approximate nearest neighbor (ANN) methods or embedding-based models.  
Filtering Strategy: Including only completed and rated anime improved recommendation precision by removing weak/noise interactions.

## Next Steps

1. A/B Test Deployment Options: Test if users prefer seeing recommendations during onboarding or after first 3 watches.  
2. Incorporate Genre Diversity: Penalize over-recommendation of similar genres to increase discovery.  
3. Hybrid Model: Blend item-based CF with popularity-based fallback for cold users with very little input.  
4. Build Explanation Layer: “Because you liked Death Note, we think you’ll enjoy Parasyte.”

## Skills Demonstrated

Recommender Systems (Item-Based CF)  
Python Data Wrangling  
Similarity Metrics (Cosine)  
Product Thinking for Cold Start Problems  
Dataset Enrichment and Output Structuring