**Documentation: Hybrid Recommendation System**

**Introduction**

This project implements a **Hybrid Recommendation System** that combines **Collaborative Filtering** (user-item interactions) and **Content-Based Filtering** (story categories) to suggest relevant stories for users.

**Features**

* **Collaborative Filtering**: Finds similar users and recommends stories based on their preferences.
* **Content-Based Filtering**: Uses story categories to recommend similar stories.
* **Hybrid Approach**: Merges recommendations from both methods for more accurate results.
* **Optimized Performance**: Uses **Nearest Neighbors** instead of large similarity matrices for efficiency.

**Prerequisites**

* Python 3.x
* Required Libraries:
  + pandas
  + numpy
  + scipy
  + sklearn

Install dependencies using:

pip install pandas numpy scipy scikit-learn

**Dataset Details**

* **User Interaction Dataset** (User\_interaction.csv):
  + user\_id: Unique identifier for users.
  + pratilipi\_id: Unique identifier for stories.
  + read\_percent: Percentage of story read by the user.
  + updated\_at: Timestamp of the last interaction.
* **Metadata Dataset** (Metadata.csv):
  + pratilipi\_id: Unique identifier for stories.
  + author\_id: Unique identifier for authors.
  + category\_name: Genre/category of the story.
  + reading\_time: Estimated reading time.
  + updated\_at: Timestamp of the last update.
  + published\_at: Publication timestamp.

**Code Structure**

**1️ load\_and\_preprocess\_data()**

* Loads user interaction and metadata files.
* Merges datasets on pratilipi\_id.
* Converts read\_percent into binary (read\_binary > 50%).
* Computes read\_count per story.

**2️ collaborative\_filtering(merged\_data)**

* Maps user\_id and pratilipi\_id to unique indices.
* Creates a **User-Item Sparse Matrix**.
* Uses **Nearest Neighbors** for user similarity.

**3 content\_based\_filtering(meta\_data)**

* Converts category\_name into TF-IDF vectors.
* Uses **Nearest Neighbors** for story similarity.

**4️ generate\_recommendations(user\_id, ...)**

* Finds similar users using **Collaborative Filtering**.
* Maps recommended indices back to pratilipi\_id.
* Uses **Content-Based Filtering** for additional recommendations.

**5️ main()**

* Loads and preprocesses data.
* Initializes **Collaborative Filtering** and **Content-Based Filtering** models.
* Runs recommendations for sample users.

**Output Example**

Generating recommendations for user 5506791961876448...

Top 5 recommendations for user 5506791961876448:

📖 Pratilipi ID: 1377786228262109

📖 Pratilipi ID: 1377786228237359

📖 Pratilipi ID: 1377786226708728

📖 Pratilipi ID: 1377786220012297

📖 Pratilipi ID: 1377786219107242