**Documentation: Pratilipi Recommendation System**

**1. Data Analysis**

The dataset used in this project consists of two main CSV files:

**1.1 user\_interaction.csv**

* **Description**: This file contains user interaction data with pratilipis (stories). It includes:
  + user\_id: Unique identifier for each user.
  + pratilipi\_id: Unique identifier for each pratilipi (story).
  + read\_percent: The percentage of the pratilipi that the user has read (0 to 100).
  + updated\_at: Timestamp when the interaction occurred.

**1.2 meta\_data.csv**

* **Description**: This file contains metadata about pratilipis, including:
  + author\_id: Unique identifier for the author of the pratilipi.
  + pratilipi\_id: Unique identifier for each pratilipi.
  + category\_name: Category of the pratilipi (e.g., Romance, Suspense).
  + reading\_time: Estimated time in seconds to read the pratilipi.
  + updated\_at: Timestamp of the last update to the metadata.
  + published\_at: Timestamp when the pratilipi was first published.

**1.3 Data Preprocessing**

Before training the model, the following steps were performed to preprocess the data:

* **Loading Data**: The datasets user\_interaction.csv and meta\_data.csv were loaded into Pandas DataFrames.
* **Handling Missing Data**:
  + We checked for missing values using isnull().sum().
  + Rows with missing published\_at in meta\_data.csv were dropped.
* **Feature Extraction**:
  + Only the columns user\_id, pratilipi\_id, and read\_percent were retained from the user\_interaction.csv for modeling.
  + We also extracted the pratilipi\_id and category\_name columns from meta\_data.csv, though these were not directly used in the recommendation system model but could be useful for future feature engineering.
* **Data Filtering**:
  + Filtered out pratilipis with reading time of zero from the meta\_data.csv, as these pratilipis are unlikely to be relevant for our recommendation system.

**1.4 Exploratory Data Analysis (EDA)**

* **Distribution of Read Percentage**:
  + The read\_percent column from user\_interaction.csv was analyzed to understand how users interact with pratilipis. The distribution showed that most users tend to read the entire pratilipi (close to 100%), with a few users reading only a portion (0 to 100%).
* **Unique Users and Pratilipis**:
  + The dataset contains millions of interactions with a large number of unique users and pratilipis, making it suitable for collaborative filtering techniques.
* **Data Imbalance**:
  + There is a potential imbalance in the dataset since some pratilipis have far more interactions than others. This is expected in real-world recommendation systems.

**2. Training Process**

The recommendation model is trained using **Collaborative Filtering** with the **Singular Value Decomposition (SVD)** technique, which is one of the most popular techniques for matrix factorization.

**2.1 Splitting Data**

* **Train-Test Split**: The dataset was split into training and testing sets, with 75% used for training and 25% for evaluation.
* We performed the split using the train\_test\_split function from the surprise library, ensuring that the data was randomly divided for model validation.

**2.2 Building the Model**

* **Model Type**: We chose **SVD (Singular Value Decomposition)** for collaborative filtering. SVD is particularly effective in recommending items based on users’ past interactions.
  + SVD factorizes the user-item interaction matrix into three matrices: user factors, item factors, and singular values.
  + It identifies hidden relationships between users and pratilipis by decomposing the interactions matrix, thereby predicting how much a user would like a pratilipi they haven’t interacted with yet.
* **Training**:
  + The model was trained using the SVD algorithm from the **surprise** library.
  + The model learns from the historical interactions between users and pratilipis. This helps in predicting user ratings (or interactions) for unseen pratilipis.

**2.3 Evaluating the Model**

* **RMSE (Root Mean Squared Error)**: After training the model, the **RMSE** score was calculated on the test set. RMSE gives an estimate of the prediction accuracy, where a lower RMSE indicates better performance.
  + Formula: RMSE=1n∑i=1n(ytrue,i−ypred,i)2RMSE = \sqrt{\frac{1}{n} \sum\_{i=1}^{n} (y\_{\text{true}, i} - y\_{\text{pred}, i})^2}RMSE=n1​i=1∑n​(ytrue,i​−ypred,i​)2​
  + Where y\_true are the actual values, and y\_pred are the predicted values.
* **Top-N Recommendations**:
  + For each user, the top 5 pratilipis that they are most likely to read next were predicted using the trained SVD model.
  + These predictions were based on the learned latent factors (user preferences and pratilipi characteristics) and the users' past interactions.

**3. Chosen Model: Singular Value Decomposition (SVD)**

**3.1 Why SVD?**

**SVD** was chosen because it is a widely used matrix factorization method for collaborative filtering and has several advantages:

* **Scalability**: SVD can handle large datasets effectively. Since the dataset contains millions of user-pratilipi interactions, SVD scales well to large-scale recommendation systems.
* **Latent Factor Discovery**: SVD can uncover hidden patterns in the data by learning latent factors that capture the relationship between users and pratilipis.
* **Flexibility**: SVD works well for both explicit feedback (ratings) and implicit feedback (interaction data like read percentage). In this case, we used read\_percent as the implicit feedback signal.

**3.2 Comparison with Other Models**

* **KNN (K-Nearest Neighbors)**:
  + KNN-based methods could also be used for collaborative filtering, where the nearest neighbors (users or pratilipis) are found. However, these methods tend to become computationally expensive for large datasets.
* **Matrix Factorization with Alternating Least Squares (ALS)**:
  + ALS is another matrix factorization technique, but it tends to be slower than SVD for large datasets. SVD is often preferred in terms of computational efficiency and performance for collaborative filtering tasks.
* **Neural Networks**:
  + Neural networks could be another alternative, but they require more computational resources and may be prone to overfitting without enough data. SVD provides a good balance between performance and computational cost for this project.

**3.3 SVD Parameters**

* **Factors**: The number of latent factors (user preferences and pratilipi characteristics) was set as 100.
* **Regularization**: Regularization was applied to avoid overfitting.
* **Learning Rate**: A learning rate of 0.005 was used during the training process.

**3.4 Model Performance**

* The model achieved an **RMSE of 22.37**, indicating the accuracy of predictions made by the SVD model on the test data.
* For example, the top 5 recommended pratilipis for some users were as follows:

User 5506791966280495: [(1377786219672726, 100)]

User 5506791968624247: [(1377786223936248, 100), (1377786222606858, 100), (1377786224655297, 100), (1377786228039628, 100), (1377786222825840, 100)]

User 5506791987341146: [(1377786227792278, 100), (1377786227820041, 100), (1377786221889554, 100), (1377786222685592, 100), (1377786222053149, 100)]

**3.5 Model Limitations and Future Work**

* **Cold Start Problem**: The model struggles with recommending pratilipis for new users or pratilipis that have very few interactions.
* **Data Sparsity**: Some pratilipis or users might not have enough historical data, leading to less accurate predictions.
* **Hybrid Models**: In future iterations, combining collaborative filtering (SVD) with content-based methods (such as category and author information) could improve recommendation accuracy.

**Conclusion**

The **SVD-based collaborative filtering** recommendation system provides effective predictions for users based on their historical reading behavior. The model can predict the pratilipis that users are likely to engage with, helping in personalized content delivery. Future improvements can be made by incorporating additional features or exploring more advanced hybrid models.