**Movie Recommendation System Using Collaborative Filtering**

**1. Introduction**

This project involves building a **Recommendation System** using **Collaborative Filtering** and **Matrix Factorization (SVD algorithm)**. The system will suggest items (movies or books) to users based on their past preferences.

**2. Dataset Overview**

* **Dataset Used:** MovieLens 100K dataset (or a book recommendation dataset)
* **Columns:**
  + user\_id: Unique identifier for each user
  + item\_id: Unique identifier for each movie/book
  + rating: User's rating (1-5 scale)

**3. Implementation Steps**

**Step 1: Import Required Libraries**

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

from surprise import Dataset, Reader, SVD

from surprise.model\_selection import train\_test\_split

from surprise.accuracy import rmse

**Step 2: Load and Explore Dataset**

# Load MovieLens dataset

url = "https://raw.githubusercontent.com/zygmuntz/goodbooks-10k/master/ratings.csv"

df = pd.read\_csv(url)

# Display dataset structure

print(df.head())

# Rename columns for clarity

df.rename(columns={'user\_id': 'UserID', 'book\_id': 'ItemID', 'rating': 'Rating'}, inplace=True)

**Step 3: Data Preprocessing**

# Define reader format for Surprise library

reader = Reader(rating\_scale=(1, 5))

# Load dataset into Surprise framework

data = Dataset.load\_from\_df(df[['UserID', 'ItemID', 'Rating']], reader)

trainset, testset = train\_test\_split(data, test\_size=0.2)

**Step 4: Train a Collaborative Filtering Model (SVD Algorithm)**

# Train an SVD model

model = SVD()

model.fit(trainset)

# Make predictions

predictions = model.test(testset)

# Calculate RMSE (Root Mean Squared Error)

print("RMSE:", rmse(predictions))

**Step 5: Generate Book Recommendations for a User**

def get\_recommendations(user\_id, model, df, num\_recommendations=5):

unique\_books = df['ItemID'].unique()

user\_read\_books = df[df['UserID'] == user\_id]['ItemID'].values

books\_to\_predict = np.setdiff1d(unique\_books, user\_read\_books)

predictions = [model.predict(user\_id, book) for book in books\_to\_predict]

predictions.sort(key=lambda x: x.est, reverse=True)

recommended\_books = [pred.iid for pred in predictions[:num\_recommendations]]

return recommended\_books

# Get recommendations for a sample user

sample\_user = 10

recommended\_books = get\_recommendations(sample\_user, model, df)

print(f"Recommended books for User {sample\_user}: {recommended\_books}")

**4. Results and Observations**

* **Model Performance:**
  + **RMSE Score:** Measures the error in rating predictions. Lower values are better.
  + The SVD model performs well for collaborative filtering.
* **Generated Recommendations:**
  + The system suggests **books/movies** that the user has not rated but might like.

**5. Conclusion**

* **Collaborative Filtering** is effective for recommendation systems.
* The **SVD model** improves accuracy by learning latent features.
* This approach is widely used in **Netflix, Amazon, and Spotify** recommendation engines.