# Import libraries

import pandas as pd

import numpy as np

from sklearn.metrics.pairwise import cosine\_similarity

from sklearn.feature\_extraction.text import TfidfVectorizer

from surprise import Dataset, Reader, SVD

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

import pickle

# Load datasets

movies = pd.read\_csv('movies.csv') # columns: movieId, title, genres

ratings = pd.read\_csv('ratings.csv') # columns: userId, movieId, rating

# Merge datasets

df = pd.merge(movies, ratings, on='movieId')

# Content-based filtering: TF-IDF on genres

tfidf = TfidfVectorizer(stop\_words='english')

tfidf\_matrix = tfidf.fit\_transform(movies['genres'])

# Calculate cosine similarity

cosine\_sim = cosine\_similarity(tfidf\_matrix, tfidf\_matrix)

# Index mapping

indices = pd.Series(movies.index, index=movies['title'])

# Content-based recommender

def content\_recommender(title, n=5):

idx = indices[title]

sim\_scores = list(enumerate(cosine\_sim[idx]))

sim\_scores = sorted(sim\_scores, key=lambda x: x[1], reverse=True)[1:n+1]

movie\_indices = [i[0] for i in sim\_scores]

return movies['title'].iloc[movie\_indices]

# Collaborative filtering: Matrix Factorization (SVD)

reader = Reader(rating\_scale=(0.5, 5.0))

data = Dataset.load\_from\_df(ratings[['userId', 'movieId', 'rating']], reader)

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

model = SVD()

model.fit(trainset)

# Collaborative prediction

def predict\_rating(user\_id, movie\_id):

pred = model.predict(user\_id, movie\_id)

return pred.est

# Hybrid recommender

def hybrid\_recommender(user\_id, title, top\_n=5):

content\_titles = content\_recommender(title, 20)

content\_ids = movies[movies['title'].isin(content\_titles)].movieId

predictions = [(mid, predict\_rating(user\_id, mid)) for mid in content\_ids]

predictions.sort(key=lambda x: x[1], reverse=True)

top\_movie\_ids = [pid[0] for pid in predictions[:top\_n]]

return movies[movies['movieId'].isin(top\_movie\_ids)]['title'].values

# Example: Recommend movies for user 1 based on "Toy Story"

print(hybrid\_recommender(user\_id=1, title="Toy Story", top\_n=5))

# Save model

with open("svd\_model.pkl", "wb") as f:

pickle.dump(model, f)