from flask import Flask, render_template, request import pandas as pd import numpy as np import networkx as nx from surprise import Dataset, Reader, KNNBasic

Initialize Flask App

app = Flask(**name)**

Load Datasets

ratings_df = pd.read_csv('ratings.csv') trust_df = pd.read_csv('trust.csv')

Build trust graph

trust_graph = nx.DiGraph() for idx, row in trust_df.iterrows(): trust_graph.add_edge(row['UserID'], row['TrustedUserID'], weight=1)

Collaborative Filtering Model

reader = Reader(rating_scale=(1,5)) data = Dataset.load_from_df(ratings_df[['UserID','ItemID','Rating']], reader) trainset = data.build_full_trainset()

sim_options = {'name': 'pearson', 'user_based': True} cf_model = KNNBasic(sim_options=sim_options) cf_model.fit(trainset)

Trust Propagation Function

def propagate_trust(trust_graph, user_id, max_depth=2): trust_scores = {} for target in trust_graph.nodes(): if target == user_id: continue paths = list(nx.all_simple_paths(trust_graph, source=user_id, target=target, cutoff=max_depth)) score = 0 for path in paths: path_weight = 1 for i in range(len(path)-1): path_weight *= trust_graph[path[i]][path[i+1]]['weight'] score += path_weight if score > 0:

trust_scores[target] = score if trust_scores: max_score = max(trust_scores.values()) for k in trust_scores: trust_scores[k] /= max_score return trust_scores

Hybrid Rating Prediction

def predict_rating(user_id, item_id, cf_model, trust_scores, alpha=0.7): neighbors = cf_model.get_neighbors(user_id, k=20) cf_pred = [] trust_weighted_pred = []

```
for neighbor in neighbors:
    try:
        pred = cf_model.predict(neighbor, item_id).est
        cf_pred.append(pred)
        trust = trust_scores.get(neighbor, 0)
        trust_weighted_pred.append(pred * trust)
    except:
        continue

if cf_pred:
    cf_mean = np.mean(cf_pred)
    trust_mean = np.mean(trust_weighted_pred) if trust_weighted_pred
else 0
    hybrid_rating = alpha * cf_mean + (1 - alpha) * trust_mean
    return hybrid_rating
else:
    return ratings_df['Rating'].mean()
```

Generate Top-N Recommendation

def top_n_recommendations(user_id, cf_model, trust_graph, N=10): trust_scores = propagate_trust(trust_graph, user_id) items = ratings_df['ItemID'].unique() predictions = [] for item in items: pred = predict_rating(user_id, item, cf_model, trust_scores) predictions.append((item, pred)) predictions.sort(key=lambda x: x[1], reverse=True) return predictions[:N]

Flask Routes

@app.route('/', methods=['GET', 'POST']) def index(): recommendations = [] if request.method == 'POST': user_id = int(request.form['user_id']) recommendations = top_n_recommendations(user_id, cf_model, trust_graph) return render_template('index.html', recommendations=recommendations

Run App

if **name** == '**main**': app.run(debug=True)