Title: Early Detection of Mental Health Issues through Social Network Behavior Analysis using Machine Learning

import numpy as np

import pandas as pd

import random, re

import matplotlib.pyplot as plt

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

from sklearn.feature\_extraction.text import TfidfVectorizer

from sklearn.preprocessing import StandardScaler

from sklearn.ensemble import RandomForestClassifier

from sklearn.svm import SVC

from sklearn.metrics import classification\_report, confusion\_matrix, accuracy\_score

# -------------------------------

# 1. SYNTHETIC DATA GENERATION

# -------------------------------

random.seed(42)

np.random.seed(42)

# Create sample text templates

texts = {

"Normal": [

"Feeling great today! Had fun with friends.",

"Work went smoothly, staying productive and positive.",

"Enjoying some peaceful time with family."

],

"Mild": [

"Feeling a bit low today, trying to stay motivated.",

"Not sleeping well lately, too many thoughts.",

"Sometimes I feel anxious and overwhelmed."

],

"High": [

"I feel hopeless and empty, nothing makes sense anymore.",

"Everything feels pointless, losing interest in things I loved.",

"I can’t get out of bed, feeling completely drained."

]

}

data = []

for \_ in range(1000):

label = random.choices(["Normal", "Mild", "High"], weights=[0.6, 0.25, 0.15])[0]

text = random.choice(texts[label])

posting\_freq = np.random.poisson(3 if label == "Normal" else 2 if label == "Mild" else 1)

interaction\_rate = np.random.normal(10 if label == "Normal" else 5 if label == "Mild" else 2, 2)

late\_night\_ratio = np.random.beta(1, 9) if label == "Normal" else np.random.beta(2, 5) if label == "Mild" else np.random.beta(4, 3)

sentiment\_score = len(re.findall(r"\b(great|fun|positive|happy)\b", text)) - len(re.findall(r"\b(hopeless|anxious|sad|drained)\b", text))

data.append([text, posting\_freq, interaction\_rate, late\_night\_ratio, sentiment\_score, label])

df = pd.DataFrame(data, columns=["text", "posting\_freq", "interaction\_rate", "late\_night\_ratio", "sentiment\_score", "label"])

print("Sample data:\n", df.head())

# --------------------------------

# 2. TEXT PREPROCESSING & FEATURES

# --------------------------------

def clean\_text(text):

text = text.lower()

text = re.sub(r'[^a-z\s]', '', text)

return text

df["clean\_text"] = df["text"].apply(clean\_text)

# TF-IDF feature extraction

tfidf = TfidfVectorizer(max\_features=300)

X\_text = tfidf.fit\_transform(df["clean\_text"]).toarray()

# Combine text + behavioral features

behavior\_features = df[["posting\_freq", "interaction\_rate", "late\_night\_ratio", "sentiment\_score"]].values

scaler = StandardScaler()

behavior\_scaled = scaler.fit\_transform(behavior\_features)

X = np.hstack((X\_text, behavior\_scaled))

y = df["label"].map({"Normal": 0, "Mild": 1, "High": 2})

# --------------------------------

# 3. TRAIN-TEST SPLIT

# --------------------------------

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.25, stratify=y, random\_state=42)

# --------------------------------

# 4. MODEL TRAINING

# --------------------------------

svm\_model = SVC(kernel="rbf", random\_state=42)

rf\_model = RandomForestClassifier(n\_estimators=200, random\_state=42)

svm\_model.fit(X\_train, y\_train)

rf\_model.fit(X\_train, y\_train)

# --------------------------------

# 5. MODEL EVALUATION

# --------------------------------

svm\_pred = svm\_model.predict(X\_test)

rf\_pred = rf\_model.predict(X\_test)

print("\n===== SVM Model Results =====")

print(classification\_report(y\_test, svm\_pred, target\_names=["Normal", "Mild", "High"]))

print("Accuracy:", round(accuracy\_score(y\_test, svm\_pred)\*100, 2), "%")

print("\n===== Random Forest Results =====")

print(classification\_report(y\_test, rf\_pred, target\_names=["Normal", "Mild", "High"]))

print("Accuracy:", round(accuracy\_score(y\_test, rf\_pred)\*100, 2), "%")

# --------------------------------

# 6. CONFUSION MATRIX PLOT

# --------------------------------

def plot\_confusion(cm, labels, title):

fig, ax = plt.subplots(figsize=(5,4))

im = ax.imshow(cm, cmap="Blues")

plt.title(title)

plt.xticks(np.arange(len(labels)), labels, rotation=45)

plt.yticks(np.arange(len(labels)), labels)

thresh = cm.max() / 2

for i in range(cm.shape[0]):

for j in range(cm.shape[1]):

ax.text(j, i, str(cm[i, j]),

ha="center", va="center",

color="white" if cm[i, j] > thresh else "black")

plt.xlabel("Predicted Label")

plt.ylabel("True Label")

plt.tight\_layout()

plt.show()

# Plot confusion matrix for Random Forest

cm\_rf = confusion\_matrix(y\_test, rf\_pred)

plot\_confusion(cm\_rf, ["Normal", "Mild", "High"], "Random Forest Confusion Matrix")

# --------------------------------

# 7. SAVE MODELS (optional)

# --------------------------------

import joblib

joblib.dump(tfidf, "tfidf\_vectorizer.pkl")

joblib.dump(scaler, "scaler.pkl")

joblib.dump(rf\_model, "mental\_health\_rf\_model.pkl")

print("\nModel and preprocessing files saved successfully.")

print("Pipeline execution completed.")