# Import necessary libraries

import numpy as np

import pandas as pd

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

from sklearn.feature\_extraction.text import CountVectorizer

from sklearn.naive\_bayes import MultinomialNB

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

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense, Embedding, LSTM

from tensorflow.keras.preprocessing.text import Tokenizer

from tensorflow.keras.preprocessing.sequence import pad\_sequences

# Load dataset (you need a labeled dataset for mental health detection)

# This dataset should have text samples and corresponding labels (e.g., 0 for not mental health, 1 for mental health)

# Replace 'your\_dataset.csv' with the actual filename and adjust the code accordingly.

dataset = pd.read\_csv('your\_dataset.csv')

# Preprocess the text data

X = dataset['text']

y = dataset['label']

# Split the dataset into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Tokenize and pad the text data for deep learning models

max\_words = 1000 # Adjust as needed

tokenizer = Tokenizer(num\_words=max\_words)

tokenizer.fit\_on\_texts(X\_train)

X\_train\_seq = tokenizer.texts\_to\_sequences(X\_train)

X\_test\_seq = tokenizer.texts\_to\_sequences(X\_test)

max\_len = 100 # Adjust as needed

X\_train\_seq = pad\_sequences(X\_train\_seq, maxlen=max\_len)

X\_test\_seq = pad\_sequences(X\_test\_seq, maxlen=max\_len)

# Build a simple LSTM model for text classification

model = Sequential()

model.add(Embedding(input\_dim=max\_words, output\_dim=32, input\_length=max\_len))

model.add(LSTM(100))

model.add(Dense(1, activation='sigmoid'))

model.compile(optimizer='adam', loss='binary\_crossentropy', metrics=['accuracy'])

# Train the model

model.fit(X\_train\_seq, y\_train, epochs=5, batch\_size=32, validation\_split=0.2)

# Evaluate the model

y\_pred = model.predict\_classes(X\_test\_seq)

accuracy = accuracy\_score(y\_test, y\_pred)

conf\_matrix = confusion\_matrix(y\_test, y\_pred)

print(f"Accuracy: {accuracy}")

print(f"Confusion Matrix:\n{conf\_matrix}")