import pandas as pd

import re

import os

from bs4 import BeautifulSoup

from sklearn.feature\_extraction.text import CountVectorizer

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

from sklearn.naive\_bayes import MultinomialNB

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

import nltk

from nltk.corpus import stopwords

# Download NLTK resources

# nltk.download('stopwords')

current\_directory = os.getcwd() # Get current working directory

file\_path = os.path.join(current\_directory, 'drug.csv') # Construct the file path

df = pd.read\_csv(file\_path)

# Read the drug review dataset (drug\_reviews.csv)

# df = pd.read\_csv('drug\_reviews.csv')

# Data preprocessing and cleaning

stop\_words = set(stopwords.words('english'))

def clean\_text(text):

text = BeautifulSoup(text, 'html.parser').get\_text() # Remove HTML tags

text = re.sub(r"[^a-zA-Z]", " ", text) # Remove non-alphabetic characters

words = text.lower().split() # Convert to lowercase and split into words

words = [w for w in words if w not in stop\_words] # Remove stopwords

return " ".join(words)

df['clean\_review'] = df['review'].apply(clean\_text)

# Convert ratings to sentiment labels: Positive (1), Neutral (0), Negative (-1)

df['sentiment'] = df['rating'].apply(lambda x: 1 if x > 3 else (-1 if x < 3 else 0))

# Split data into features (X) and labels (y)

X = df['clean\_review']

y = df['sentiment']

# Split data 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)

# Vectorize text data using CountVectorizer

vectorizer = CountVectorizer()

X\_train\_vectorized = vectorizer.fit\_transform(X\_train)

X\_test\_vectorized = vectorizer.transform(X\_test)

# Train a Naive Bayes classifier

classifier = MultinomialNB()

classifier.fit(X\_train\_vectorized, y\_train)

predictions = classifier.predict(X\_test\_vectorized)

# Evaluate the classifier

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

print("Accuracy:", accuracy)

# Print classification report

print("Classification Report:")

print(classification\_report(y\_test, predictions))

# Sample usage of the trained model with top 3 recommendations

def predict\_sentiment\_with\_recommendations(review\_text):

cleaned\_text = clean\_text(review\_text)

vectorized\_text = vectorizer.transform([cleaned\_text])

sentiment\_label = classifier.predict(vectorized\_text)[0]

if sentiment\_label == 1:

sentiment = "Positive"

elif sentiment\_label == 0:

sentiment = "Neutral"

else:

sentiment = "Negative"

# Extract top 3 recommendations based on the predicted sentiment

target\_condition = "Positive" if sentiment\_label == 1 else "Negative" if sentiment\_label == -1 else "Neutral"

top\_recommendations = df[(df['sentiment'] == sentiment\_label) & (df['rating'] > 3)].nlargest(3, 'rating')['drugName'].tolist()

return sentiment, top\_recommendations

# Example usage

sample\_review = "This medication worked wonders for my condition. I highly recommend it."

predicted\_sentiment, top\_recommendations = predict\_sentiment\_with\_recommendations(sample\_review)

print("Predicted Sentiment:", predicted\_sentiment)

print(sample\_review)

print("Top 3 Recommendations:")

print(top\_recommendations)

sample\_review = "I have only been on Tekturna for 9 days.The effect was immediate. I am also on a calcium channel blocker (Tiazac) and hydrochlorothiazide. I was put on Tekturna because of palpitations experienced with Diovan (ugly drug in my opinion, same company produces both however). The palpitations were pretty bad on Diovan, 24 hour monitor by EKG etc. After a few days of substituting Tekturna for Diovan, there are no more palpitations"

predicted\_sentiment, top\_recommendations = predict\_sentiment\_with\_recommendations(sample\_review)

print("Predicted Sentiment:", predicted\_sentiment)

print(sample\_review)

print("Top 3 Recommendations:")

print(top\_recommendations)