# Importing necessary libraries

import pandas as pd

import numpy as np

import re

import nltk

from nltk.corpus import stopwords

from nltk.tokenize import word\_tokenize

from sklearn.feature\_extraction.text import TfidfVectorizer

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

from sklearn.naive\_bayes import MultinomialNB

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

from sklearn.model\_selection import GridSearchCV

from flask import Flask, request, jsonify

# Downloading NLTK resources

nltk.download('punkt')

nltk.download('stopwords')

# Load the dataset

data = pd.read\_csv("IMDB Dataset.csv")

# Data Preprocessing

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

def preprocess\_text(text):

text = text.lower() # Convert to lowercase

text = re.sub(r'[^a-zA-Z\s]', '', text) # Remove special characters and digits

words = word\_tokenize(text) # Tokenization

words = [word for word in words if word not in stop\_words] # Removing stopwords

return ' '.join(words)

data['clean\_text'] = data['review'].apply(preprocess\_text)

# Feature Extraction

tfidf\_vectorizer = TfidfVectorizer()

X = tfidf\_vectorizer.fit\_transform(data['clean\_text'])

y = data['sentiment']

# Model Selection and Training

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

model = MultinomialNB()

model.fit(X\_train, y\_train)

# Hyperparameter Tuning

parameters = {'alpha': [0.1, 1, 10]}

grid\_search = GridSearchCV(model, parameters, cv=5, scoring='accuracy')

grid\_search.fit(X\_train, y\_train)

best\_params = grid\_search.best\_params\_

# Model Evaluation

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

print("Classification Report:")

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

print("Confusion Matrix:")

print(confusion\_matrix(y\_test, y\_pred))

# Deployment using Flask

app = Flask(name)

@app.route('/predict', methods=['POST'])

def predict\_sentiment():

data = request.get\_json()

review = data['review']

preprocessed\_review = preprocess\_text(review)

tfidf\_review = tfidf\_vectorizer.transform([preprocessed\_review])

prediction = model.predict(tfidf\_review)[0]

return jsonify({'sentiment': prediction})

if name == 'main':

app.run(debug=True)