import pandas as pd

import re

import nltk

from nltk.corpus import stopwords

from nltk.tokenize import word\_tokenize

from nltk.stem import PorterStemmer

from sklearn.feature\_extraction.text import TfidfVectorizer

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

from sklearn.linear\_model import LogisticRegression

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

# Download necessary NLTK data

nltk.download('stopwords')

nltk.download('punkt')

def preprocess\_text(text):

# Convert to lowercase

text = text.lower()

# Remove special characters, numbers, and punctuation

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

# Tokenization

tokens = word\_tokenize(text)

# Remove stopwords

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

tokens = [word for word in tokens if word not in stop\_words]

# Stemming

stemmer = PorterStemmer()

tokens = [stemmer.stem(word) for word in tokens]

# Join tokens back to string

return ' '.join(tokens)

# Load dataset (Assuming CSV file with 'review' and 'label' columns)

df = pd.read\_csv("dataset.csv")

# Apply preprocessing

df['cleaned\_review'] = df['review'].astype(str).apply(preprocess\_text)

# Vectorization using TF-IDF

vectorizer = TfidfVectorizer(max\_features=5000)

X = vectorizer.fit\_transform(df['cleaned\_review'])

y = df['label']

# 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)

# Train Logistic Regression model

model = LogisticRegression()

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

# Make predictions

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

# Evaluate model

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

print("Accuracy:", accuracy)

print("Classification Report:\n", classification\_report(y\_test, y\_pred))

# Save processed dataset

df.to\_csv("processed\_dataset.csv", index=False)

print("Preprocessing and model training complete. Processed dataset saved as 'processed\_dataset.csv'")

import pickle

# Save the trained model

with open("model.pkl", "wb") as model\_file:

pickle.dump(model, model\_file)

# Save the TF-IDF vectorizer

with open("vectorizer.pkl", "wb") as vectorizer\_file:

pickle.dump(vectorizer, vectorizer\_file)

@app.post("/predict/")

def predict\_review(data: ReviewRequest):

processed\_review = preprocess\_text(data.review)

transformed\_review = vectorizer.transform([processed\_review])

prediction = model.predict(transformed\_review)

return {

"review": data.review,

"fake\_review": bool(prediction[0]),

"message": "Fake review detected!" if prediction[0] else "Review is genuine."

}

if \_\_name\_\_ == "\_\_main\_\_":

import uvicorn

uvicorn.run(app, host="0.0.0.0", port=8000)