EBPL – DS-Exposing the Truth with Advanced Fake News Detection Powered by Natural Language Processing

Source code

```
import pandas as pd
import numpy as np
import nltk
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
import re
# Download stopwords and wordnet
nltk.download('stopwords')
nltk.download('wordnet')

# Initialize lemmatizer and stopwords
lemmatizer = WordNetLemmatizer()
stop_words = set(stopwords.words('english'))
# Load the dataset
data = pd.read_csv('fake_news_detection_dataset.csv')

# Display basic info about the dataset
print('Dataset Info:')
print(data.info())
print('\nSample Data:')
print(data.head())
# Remove missing values and duplicates
data.dropna(inplace=True)
data.drop_duplicates(inplace=True)
# Text Preprocessing Function
def preprocess_text(text):
    # Lowercase the text
    text = text.lower()
    # Remove special characters and digits
    text = re.sub(r'[^a-zA-Z\s]', '', text)
    # Tokenize and remove stopwords
    tokens = text.split()
    tokens = [word for word in tokens if word not in stop_words]
    # Lemmatize the words
    tokens = [lemmatizer.lemmatize(word) for word in tokens]
    # Join tokens back into a single string
    return ' '.join(tokens)
# Apply preprocessing to the text column
data['cleaned_text'] = data['text'].apply(preprocess_text)
```

```
# Display the cleaned data
print('\nCleaned Data Sample:')
print(data[['text', 'cleaned_text']].head())

# Save the cleaned dataset
data.to_csv('cleaned_fake_news_dataset.csv', index=False)
print('Data Preprocessing Completed and Saved as
cleaned_fake_news_dataset.csv')
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from wordcloud import WordCloud

# Load the cleaned dataset
data = pd.read_csv('cleaned_fake_news_dataset.csv')

# Display basic statistics
print('Label Distribution:')
print(data['label'].value_counts())

# Plot label distribution
plt.figure(figsize=(6,4))
sns.countplot(x='label', data=data, palette='Set2')
plt.title('Distribution of Fake vs Real News')
plt.show()
# Generate word clouds for Fake and Real news
plt.figure(figsize=(10,5))

# Word Cloud for Fake News
plt.subplot(1, 2, 1)
fake_news = data[data['label'] == 'Fake']
wordcloud_fake = WordCloud(width=400, height=300,
background_color='black').generate(' '.join(fake_news['cleaned_text']))
plt.imshow(wordcloud_fake, interpolation='bilinear')
plt.axis('off')
plt.title('Fake News Word Cloud')
# Word Cloud for Real News
plt.subplot(1, 2, 2)
real_news = data[data['label'] == 'Real']
wordcloud_real = WordCloud(width=400, height=300,
background_color='black').generate(' '.join(real_news['cleaned_text']))
plt.imshow(wordcloud_real, interpolation='bilinear')
plt.axis('off')
plt.title('Real News Word Cloud')
plt.show()
# Plot text length distribution
data['text_length'] = data['cleaned_text'].apply(len)
plt.figure(figsize=(8,5))
```

```
sns.histplot(data=data, x='text_length', hue='label', element='step',
bins=30, palette='Set2')
plt.title('Text Length Distribution')
plt.show()
from sklearn.feature_extraction.text import TfidfVectorizer
import pandas as pd
import pickle

# Load the cleaned dataset
data = pd.read_csv('cleaned_fake_news_dataset.csv')

# TF-IDF Vectorization
tfidf_vectorizer = TfidfVectorizer(max_features=5000, ngram_range=(1,
2))
X = tfidf_vectorizer.fit_transform(data['cleaned_text'])
# Display the shape of the feature matrix
print(f'Feature Matrix Shape: {X.shape}')

# Save the vectorizer and feature matrix
pickle.dump(tfidf_vectorizer, open('tfidf_vectorizer.pkl', 'wb'))
pickle.dump(X, open('tfidf_features.pkl', 'wb'))
pickle.dump(data['label'], open('labels.pkl', 'wb'))

print('Feature Extraction Completed and Saved: tfidf_vectorizer.pkl,
tfidf_features.pkl, labels.pkl')
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import classification_report, confusion_matrix,
accuracy_score
import pickle

# Load the feature matrix and labels
X = pickle.load(open('tfidf_features.pkl', 'rb'))
y = pickle.load(open('labels.pkl', 'rb'))
# Train-Test Split
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

# Initialize models
models = {
    'Logistic Regression': LogisticRegression(max_iter=1000),
    'Support Vector Machine': SVC(),
    'Random Forest': RandomForestClassifier(),
    'Naive Bayes': MultinomialNB()}
# Save the best model (Logistic Regression in this case for simplicity)
```

```
pickle.dump(models['Logistic Regression'], open('best_model.pkl',
'wb'))
print('Model Training Completed and Best Model Saved as
best_model.pkl')

import matplotlib.pyplot as plt
from sklearn.metrics import ConfusionMatrixDisplay, roc_curve, auc
import pickle

# Load the model, features, and labels
model = pickle.load(open('best_model.pkl', 'rb'))
X_test = pickle.load(open('tfidf_features.pkl', 'rb'))[12000:]  # Last
20% for testing
y_test = pickle.load(open('labels.pkl', 'rb'))[12000:]

# Predict the test set
y_pred = model.predict(X_test)

# Confusion Matrix Display
print('Confusion Matrix:')
ConfusionMatrixDisplay.from_estimator(model, X_test, y_test,
cmap='Blues')
plt.title('Confusion Matrix')
plt.show()

# ROC Curve
y_proba = model.decision_function(X_test)
fpr, tpr, _ = roc_curve(y_test.map({'Fake': 0, 'Real': 1}), y_proba)
roc_auc = auc(fpr, tpr)

plt.figure(figsize=(6, 4))
plt.plot(fpr, tpr, color='darkorange', lw=2, label=f'ROC curve (area =
{roc_auc:.2f})')
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC)')
plt.legend(loc="lower right")
plt.show()
import os
import pickle

# Load the model and vectorizer
model = pickle.load(open('best_model.pkl', 'rb'))
vectorizer = pickle.load(open('tfidf_vectorizer.pkl', 'rb'))
```

```
# Create the 'model' directory if it doesn't exist
os.makedirs('model', exist_ok=True)

# Save them for deployment
with open('model/fake_news_model.pkl', 'wb') as model_file:
    pickle.dump(model, model_file)

with open('model/tfidf_vectorizer.pkl', 'wb') as vectorizer_file:
    pickle.dump(vectorizer, vectorizer_file)

print('Model and Vectorizer saved successfully in the model/
directory.')
!curl -X POST http://127.0.0.1:5000/predict -H "Content-Type:
application/json" -d "{\"text\": \"Sample news article to predict.\"}"
import requests

import requests

# Sample news articles for testing
fake_news = {
    "text": "Breaking news! Alien life was discovered on Mars by
private space agencies."
}
```