FAKE NEWS DETECTION USING NLP

TEAM MEMBERS:

1.P.swathi

2.M.Nadhiya

3.V.S.varsha

Department of electronics and communication engineering – III year

Students of NPR college of engineering and technology, natham(tk), Dindigul(dt).

- *This is the begin building of our project fake news detection model by loading and preprocessing the given dataset.*
- *Givendataset:* **https://www.kaggle.com/datasets/clmentbisaillon/fake-and-real-news-dataset**
- *Preprocessed Code:*

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split,GridSearchCV
from sklearn.naive_bayes import MultinomialNB
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.svm import LinearSVC
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics import accuracy_score
from sklearn.utils import shuffle
from scipy.sparse import hstack
from sklearn.model_selection import cross_val_score,learning_curve
import matplotlib.pyplot as plt
import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))
true=pd.read_csv("/kaggle/input/fake-and-real-news-dataset/True.csv"
)
fake=pd.read_csv("/kaggle/input/fake-and-real-news-dataset/Fake.csv"
)
true.head(50)
true["subject"].value_counts()
fake.head()
fake["subject"].value_counts()
true.isnull().sum()
fake.isnull().sum()
true.shape
fake.shape
true.head()
fake.head()
true["label"]=1
fake["label"]=0
data=pd.concat([fake,true],ignore_index=True)
data.head()
X=data["text"]
y=data["label"]
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2,ran
dom_state=42)
vectorizer=CountVectorizer()
X_train_vectors=vectorizer.fit_transform(X_train)
X_test_vectors=vectorizer.transform(X_test)
vectorizer = CountVectorizer()
X_vectors = vectorizer.fit_transform(data['text'])
X_train, X_test, y_train, y_test = train_test_split(X_vectors, data[
'label'], test_size=0.2, random_state=42)
classifier = MultinomialNB()
classifier.fit(X_train, y_train)
y_pred = classifier.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

```
new_texts = ["This news article is definitely fake.",
             "The research study confirms the truth of the news."]
new_texts_vectors = vectorizer.transform(new_texts)
predictions = classifier.predict(new_texts_vectors)
for text, label in zip(new_texts, predictions):
    print(f"Text: {text}\nPrediction: {'Fake' if label == 0 else 'Tr
ue'}\n")
true_df = pd.read_csv('/kaggle/input/fake-and-real-news-dataset/True
.csv')
```

```
fake_df = pd.read_csv('/kaggle/input/fake-and-real-news-dataset/Fake
.csv')
fake_df['label'] = 0
true_df['label'] = 1
combined_df = pd.concat([fake_df, true_df], ignore_index=True)
combined_df = combined_df.sample(frac=1, random_state=42).reset_inde
x(drop=True)
X = combined_df['title'] + " " + combined_df['text']
y = combined_df['label']
vectorizer = TfidfVectorizer()
X_vectors = vectorizer.fit_transform(X)
classifier = MultinomialNB(alpha=1.0)
classifier.fit(X_vectors, y)
def predict_label(input_title):
    input_text = ""
input_data = input_title + " " + input_text
    input_vector = vectorizer.transform([input_data])
    label = classifier.predict(input_vector)[0]
    return label
input_title ="WASHINGTON (Reuters) - The special counsel"
predicted_label = predict_label(input_title)
if predicted_label == 0:
    print("Predicted Label: Fake")
else:
    print("Predicted Label: True")
```

❖ *Code explanation:*

This code is a Python script for performing text classification and analysis on a dataset of news articles to determine whether they are real or fake news . Let's break down the code step by step:

*1. **Importing Libraries**:*

- The code starts by importing necessary Python libraries, including NumPy, pandas, scikit-learn (for machine learning), Matplotlib (for plotting), and other related modules.

*2. **Data Loading**:*

- It loads two CSV files using `pd.read_csv` to read the real news dataset (`True.csv`) and fake news dataset (`Fake.csv`) from a directory, presumably from a Kaggle dataset.

*3**. Data Exploration**:*

- It explores the datasets by printing the first 50 rows of the real news dataset and checking the count of news articles in each subject category for both datasets.

- It also checks for missing values in the datasets and prints the shape (number of rows and columns) of each dataset.

*4. **Label Assignment**:*

- It assigns labels to the news articles in the real and fake datasets. Real news is assigned a label of 1, and fake news is assigned a label of 0.

*5**. Data Concatenation**:*

- It concatenates the real and fake news datasets into a single DataFrame called `data`, which combines both datasets. This merged dataset will be used for further analysis.

*6. **Data Splitting**:*

- It splits the data into training and testing sets. X contains the "text" column (the content of the news articles), and y contains the corresponding labels.

*7. **Text Vectorization**:*
- It uses the `CountVectorizer` to convert the text data into numerical vectors. Both the training and testing data are transformed separately. `X_train_vectors` and `X_test_vectors` store the vectorized training and testing data.

*8. **Multinomial Naive Bayes Classification**:*
- It uses a Multinomial Naive Bayes classifier to train on the vectorized training data (`X_train_vectors`) and their corresponding labels (`y_train`).
- It then makes predictions on the testing data and calculates the accuracy of the model using scikit-learn's `accuracy_score`.

*9. **Predicting New Texts**:*
- It uses the trained Naive Bayes classifier to predict the labels for two new text samples.
- The new texts are vectorized using the same `CountVectorizer`, and their predictions are printed.

*10. **Alternative Text Classification**:*
- The code performs an alternative text classification task. It combines the "title" and "text" columns from the real and fake news datasets and creates a new DataFrame.

- It uses TF-IDF vectorization (`TfidfVectorizer`) to transform the combined text data into numerical vectors.

- Another Multinomial Naive Bayes classifier is trained on this vectorized data.

*11.**Prediction for a Single Title**:*

- There's a function `predict_label` that takes an input title, combines it with an empty input text, vectorizes the input data, and predicts its label (fake or true).

- It demonstrates the function by using it to predict the label for a given input title.

So further after data preprocessing model fiting and accuracy results will be shown as an output during the next phase of project suobmission.