ĐẠI NAM UNIVERSITY

NHÓM 7

ĐỀ SỐ 2

Phát hiện tin giả bằng học máy

Thành viên nhóm 7

1. Nguyễn Đức Duy
2. Nguyễn Minh Đức
3. Nguyễn Tất Toàn
4. Nguyễn Văn Nguyên

Mục tiêu đề tài

"Phát hiện tin giả bằng học máy" nhằm xây dựng mô hình tự động phân loại tin thật và tin giả. Sử dụng các thuật toán như Logistic Regression và Decision Tree, kết hợp với tiền xử lý văn bản và vector hóa bằng TF-IDF, mô hình giúp phát hiện tin giả từ dữ liệu tin tức. Kết quả thực nghiệm cho thấy Logistic Regression hoạt động ổn định, còn Decision Tree dễ bị quá khớp. Đề tài cũng trực quan hóa dữ liệu và phân tích từ khóa để hỗ trợ việc đánh giá và cải tiến các phương pháp phát hiện tin giả.

Giới thiệu chương trình

Mục tiêu chương trình

Xây dựng hệ thống phát hiện tin giả bằng cách: Tiền xử lý văn bản từ bài báo. Huấn luyện và đánh giá mô hình máy học để phân loại tin thật hay giả.

Ngôn ngữ lập trình: Python

Thư viện xử lý dữ liệu: Pandas, Seaborn, Matplotlib.

Xử lý ngôn ngữ tự nhiên: NLTK, WordCloud.

Máy học: Scikit-learn (Logistic Regression, Decision Tree).

Lưu trữ: Joblib.

Thư viện đã cài đặt

Pandas, NLTK, WordCloud, Scikit-learn, Joblib, TQDM.

Thư viện đã cài đặt

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import re
import nltk
from nltk.corpus import stopwords
from tqdm import tqdm
from wordcloud import WordCloud
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
from sklearn import metrics
import joblib
```

Kiểm tra file đầu vào

```python
# Check if file is exist
def check_file_exist(file_path) -> bool:
    import os
    return os.path.exists(file_path)
```

Kiểm tra xem file dữ liệu có tồn tại ở đường dẫn được chỉ định không

```python
pkl_dir = r"C:\Users\admin\OneDrive\Desktop\ML\data\News.pkl"
csv_dir = r"C:\Users\admin\OneDrive\Desktop\ML\News.csv"
```

Tải và tiền xử lý dữ liệu

Đọc dữ liệu từ file CSV.

Loại bỏ các cột không cần thiết để tập trung vào dữ liệu chính là văn bản và nhãn phân loại (tin thật hoặc tin giả).

Xáo trộn dữ liệu để tránh thứ tự trong tập dữ liệu gây ra lệch về phân loại.

```python
# 1. Load and preprocess the data
def load_and_preprocess_data(file_path):
    data = pd.read_csv(file_path, index_col=0)
    data = data.drop(["title", "subject", "date"], axis=1)

    # Shuffling data
    data = data.sample(frac=1).reset_index(drop=True)
    return data
```

Tiền xử lí dữ liệu văn bản

Loại bỏ các ký tự không cần thiết, dấu câu, và các từ dừng (stopwords) để đơn giản hóa văn bản.

Chuyển văn bản thành chữ thường và tạo danh sách các từ quan trọng.

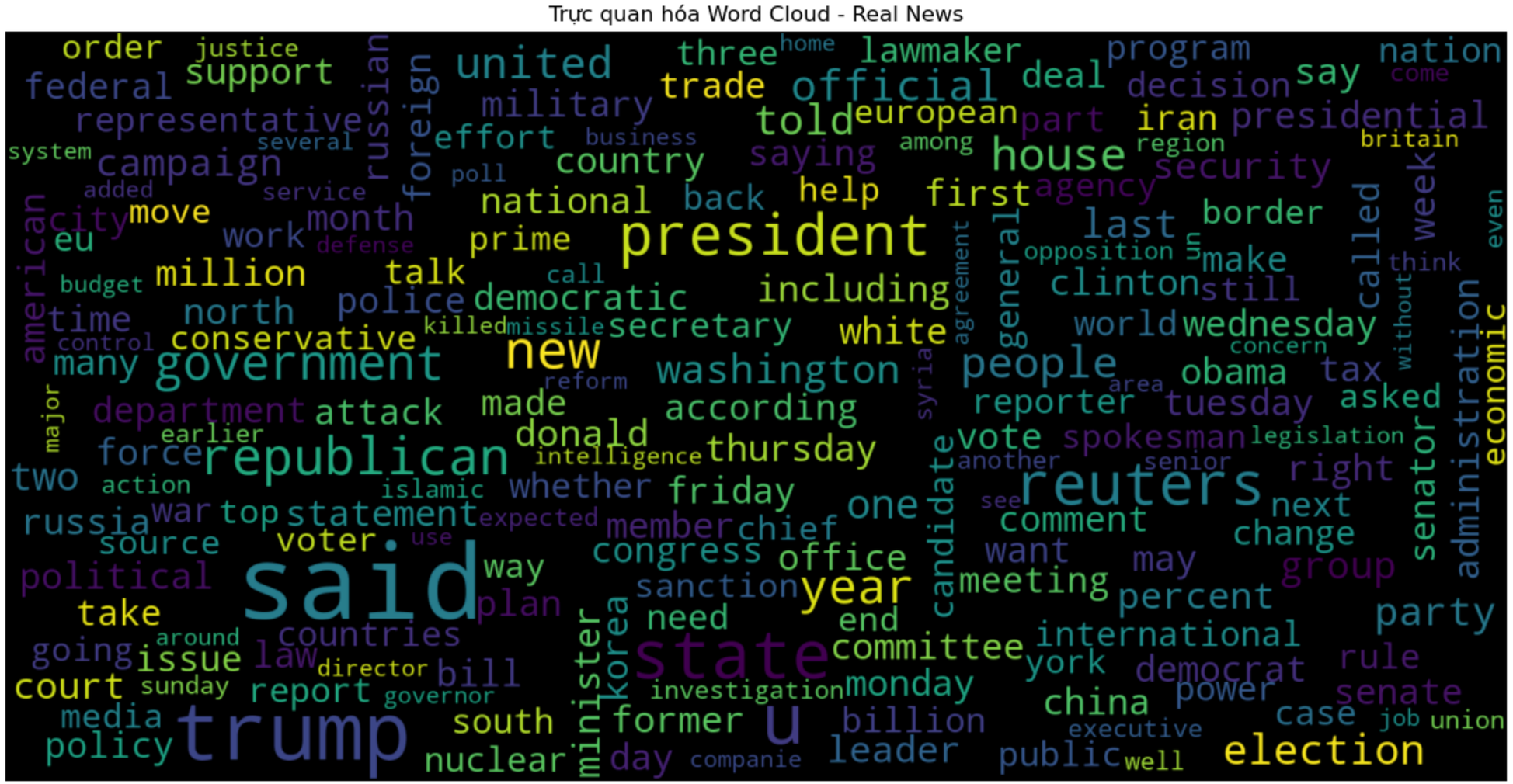
```
# 2. Preprocess the text data
def preprocess_text(text_data):
    preprocessed_text = []
    for sentence in tqdm(text_data):
        sentence = re.sub(r'[^\w\s]', '', sentence)
        preprocessed_text.append(' '.join(token.lower() for token in str(sentence).split()
    if token not in stopwords.words('english')))
    return preprocessed_text
```

Trực quan hóa dữ liệu bằng Word cloud

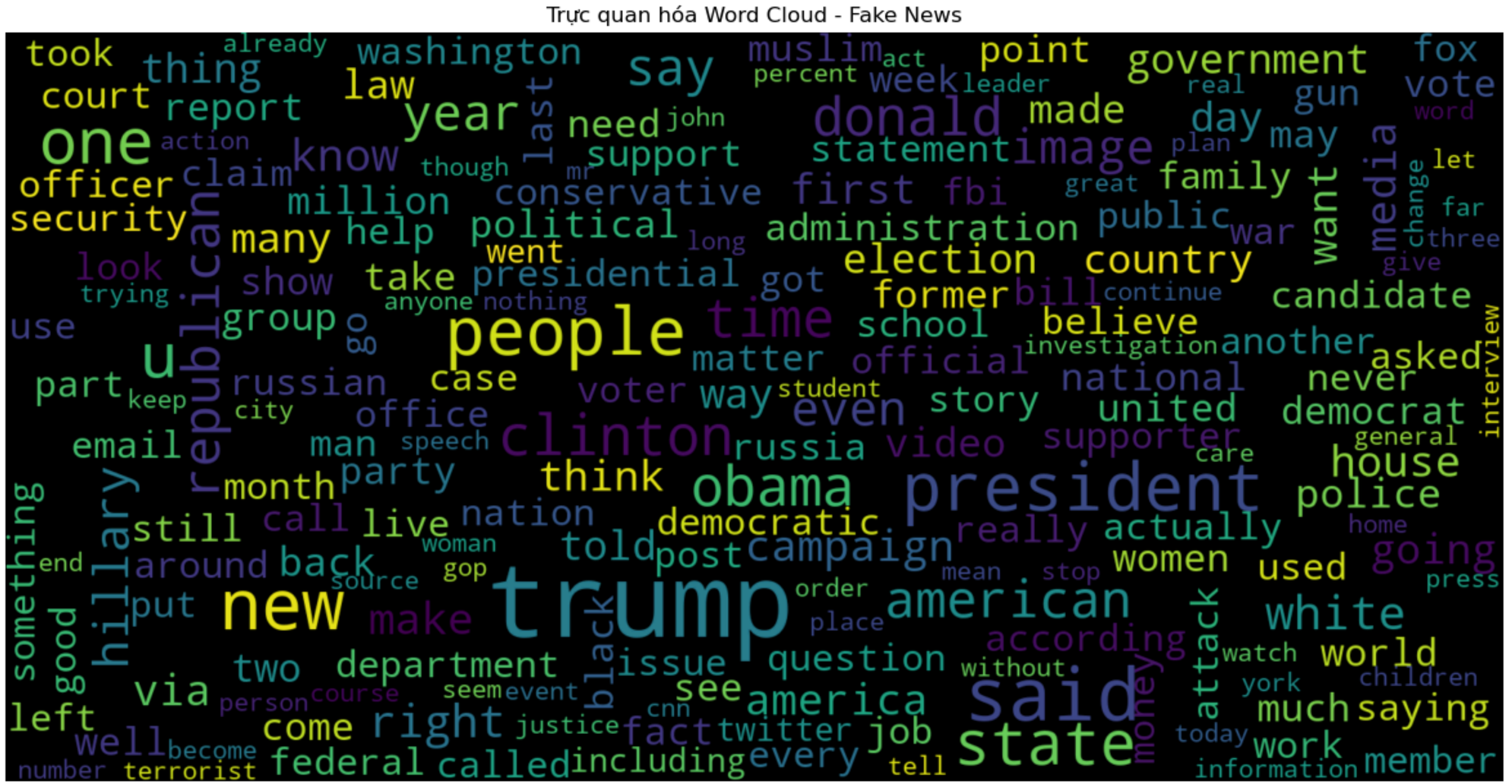
Tạo Word Cloud để trực quan hóa những từ xuất hiện nhiều nhất trong dữ liệu "tin thật" hoặc "tin giả".

```
# Trực quan hóa dữ liệu
visualize_word_cloud(data, 1, "Trực quan hóa Word Cloud - Real News")
visualize_word_cloud(data, 0, "Trực quan hóa Word Cloud - Fake News")
```

Trực quan hóa dữ liệu bằng Word cloud



Trực quan hóa dữ liệu bằng Word cloud

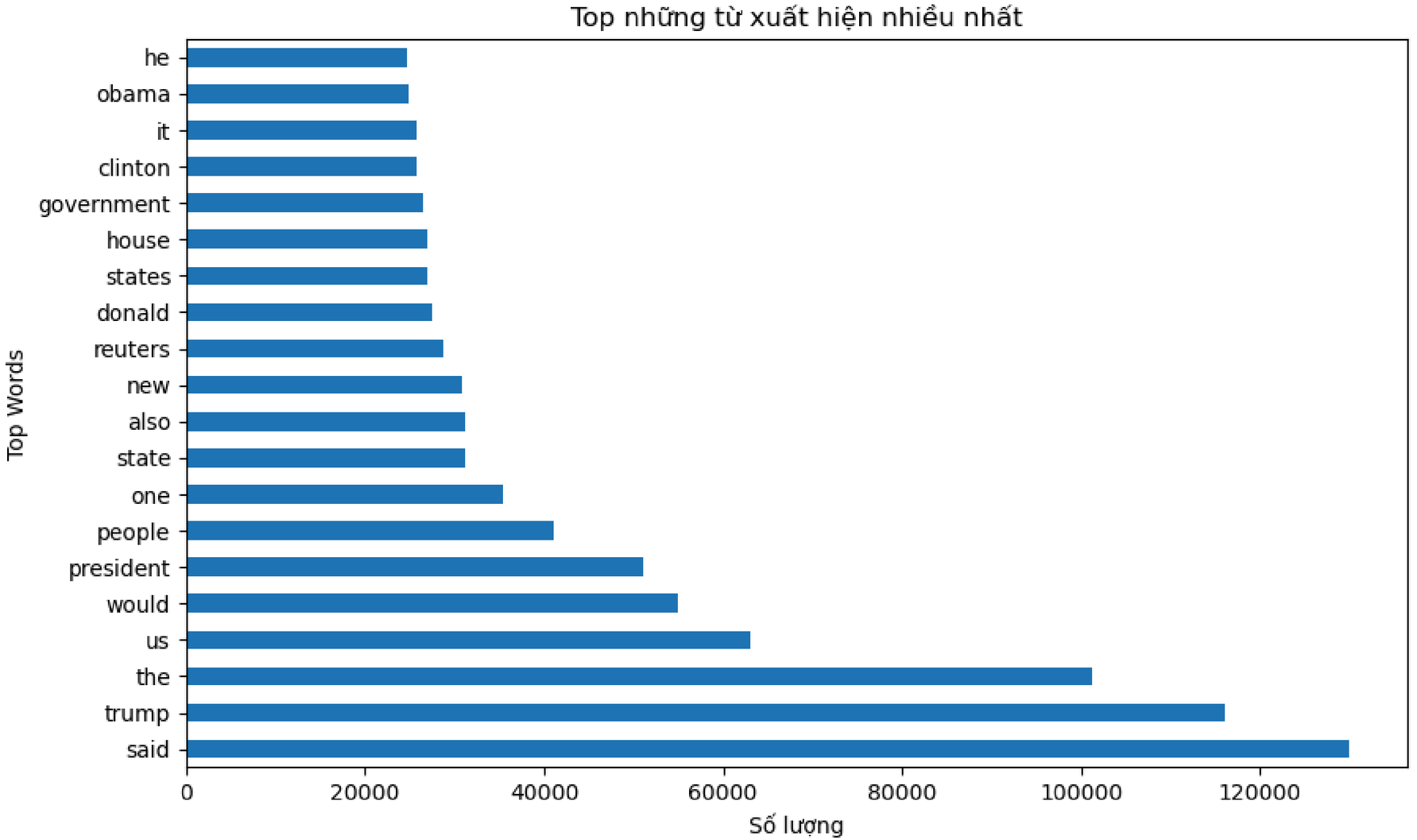


Lọc ra các từ phổ biến nhất

Tìm ra các từ phổ biến nhất trong tập dữ liệu và hiển thị bằng biểu đồ.

```
# Top 20 từ phổ biến trong dữ liệu
common_words = get_top_n_words(data['text'], 20)
df1 = pd.DataFrame(common_words, columns=['Word', 'Count'])
df1.groupby('Word').sum()['Count'].sort_values(ascending=False).plot(kind='barh', figsize=(10, 6))
plt.title("Top những từ xuất hiện nhiều nhất")
plt.ylabel("Top Words")
plt.xlabel("Số lượng")
plt.show()
```

Lọc ra các từ phổ biến nhất



ĐẠI NAM UNIVERSITY

```python
# 5. Train and evaluate model
def train_and_evaluate_model(x_train, y_train, x_test, y_test, model):
    model.fit(x_train, y_train)
    train_acc = accuracy_score(y_train, model.predict(x_train))
    test_acc = accuracy_score(y_test, model.predict(x_test))
    return train_acc, test_acc

# 6. Plot confusion matrix
def plot_confusion_matrix(model, x_test, y_test):
    cm = metrics.confusion_matrix(y_test, model.predict(x_test))
    cm_display = metrics.ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=[False, True])
    cm_display.plot()
    plt.show()
```

Chia dữ liệu thành train/test và chuyển đổi sang dạng vector

- Chia dữ liệu: Sử dụng train_test_split từ sklearn để chia dữ liệu thành 75% cho huấn luyện (train) và 25% cho kiểm tra (test). Văn bản (text) sẽ là đầu vào (X), còn nhãn phân loại (class) sẽ là đầu ra (y).
- Chuyển đổi văn bản thành vector:
- Dùng TfidfVectorizer để chuyển văn bản từ dạng text thành dạng vector số bằng cách tính toán TF-IDF (Term Frequency-Inverse Document Frequency) cho từng từ.
- fit_transform được sử dụng cho tập huấn luyện để học các giá trị từ dữ liệu.
- transform được sử dụng cho tập kiểm tra để chuyển đổi dữ liệu test theo mô hình đã học.

Chia dữ liệu thành train/test và chuyển đổi sang dạng vector

```
# Chia tập dữ liệu thành 75% train và 25% test và chuyển dữ liệu văn bản thành dạng vector
x_train, x_test, y_train, y_test = train_test_split(data['text'], data['class'], test_size=0.25)
print("Đã chia dữ liệu xong ! 75% train và 25% test")
vectorizer = TfidfVectorizer()
x_train = vectorizer.fit_transform(x_train)
x_test  = vectorizer.transform(x_test)
```

```
9]

··  Đã chia dữ liệu xong ! 75% train và 25% test
```

Huấn luyện và đánh giá mô hình

Huấn luyện mô hình Logistic Regression và Decision Tree.

Tính toán độ chính xác (accuracy) trên cả tập huấn luyện và tập kiểm tra.

```
# Huấn luyện model với mô hình Logistic Regression
log_model = LogisticRegression()
log_reg_train_acc, log_reg_test_acc = train_and_evaluate_model(x_train, y_train, x_test, y_test,log_model)
log_reg_test_acc = round(log_reg_test_acc * 100, 4)
log_reg_train_acc = round(log_reg_train_acc * 100, 4)
print(f"=== Logistic Regression ===\nĐộ chính xác của quá trình train: {log_reg_train_acc}%\nĐộ chính xác của quá trình test: {log_reg_test_acc}%")
```

```
=== Logistic Regression ===
Độ chính xác của quá trình train: 99.3915%
Độ chính xác của quá trình test: 98.7801%
```

Huấn luyện và đánh giá mô hình

Huấn luyện mô hình Logistic Regression và Decision Tree.

Tính toán độ chính xác (accuracy) trên cả tập huấn luyện và tập kiểm tra.

```
# Huấn luyện model với mô hình Decision Tree
decision_model = DecisionTreeClassifier()
dt_train_acc, dt_test_acc = train_and_evaluate_model(x_train, y_train, x_test, y_test, decision_model)
dt_test_acc = round(dt_test_acc * 100, 4)
dt_train_acc = round(dt_train_acc * 100, 4)
print(f"=== Decision Tree ===\nĐộ chính xác của quá trình train: {dt_train_acc}%\nĐộ chính xác của quá trình test: {dt_test_acc}%")
plot_confusion_matrix(decision_model, x_test, y_test)
```

```
=== Decision Tree ===
Độ chính xác của quá trình train: 99.997%
Độ chính xác của quá trình test: 99.528%
```

Huấn luyện và đánh giá mô hình

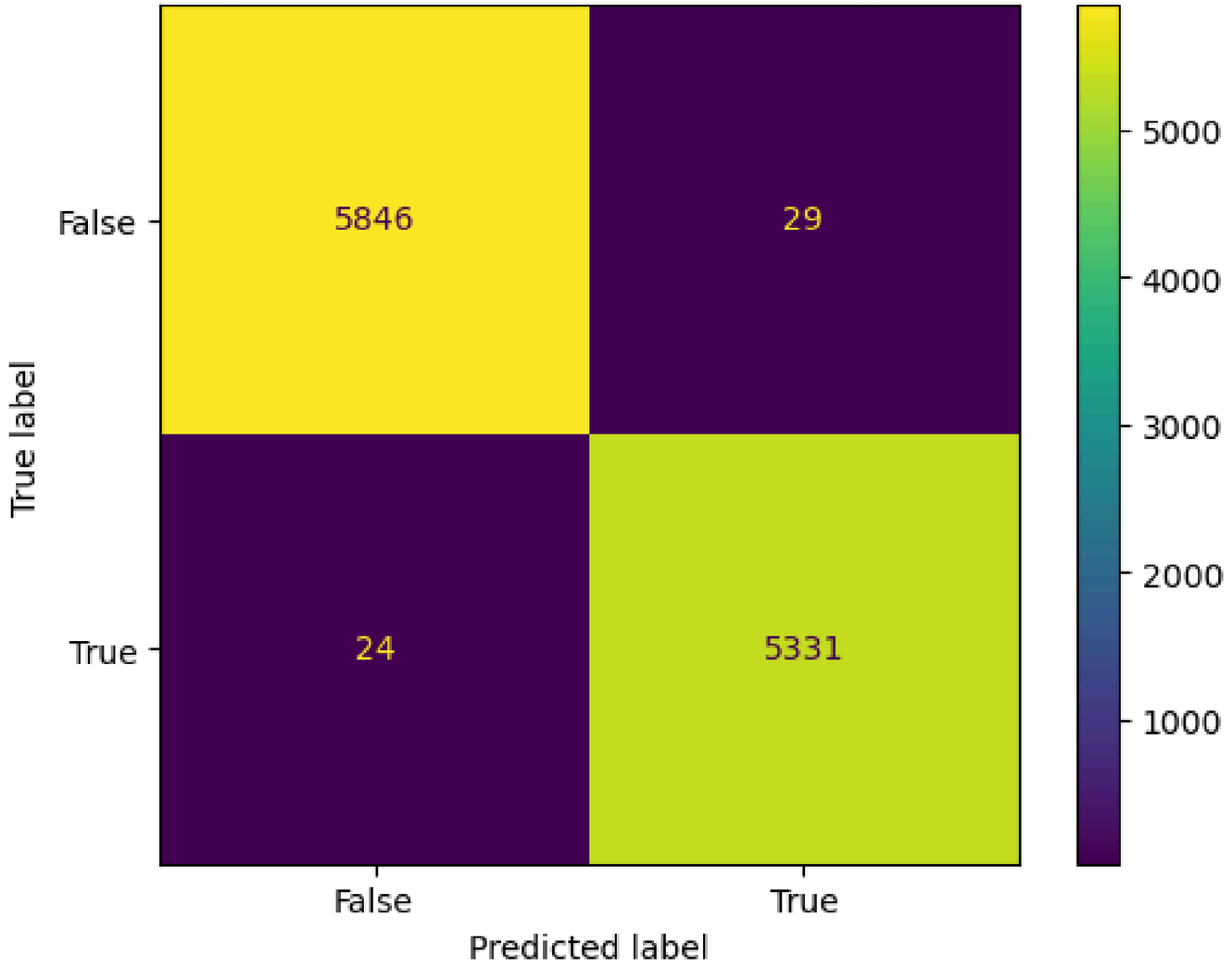
Huấn luyện mô hình Logistic Regression và Decision Tree.

Tính toán độ chính xác (accuracy) trên cả tập huấn luyện và tập kiểm tra.

```
# Huấn luyện model với mô hình Decision Tree
decision_model = DecisionTreeClassifier()
dt_train_acc, dt_test_acc = train_and_evaluate_model(x_train, y_train, x_test, y_test, decision_model)
dt_test_acc = round(dt_test_acc * 100, 4)
dt_train_acc = round(dt_train_acc * 100, 4)
print(f"=== Decision Tree ===\nĐộ chính xác của quá trình train: {dt_train_acc}%\nĐộ chính xác của quá trình test: {dt_test_acc}%")
plot_confusion_matrix(decision_model, x_test, y_test)
```

```
=== Decision Tree ===
Độ chính xác của quá trình train: 99.997%
Độ chính xác của quá trình test: 99.528%
```

Huấn luyện và đánh giá mô hình



Kiểm tra dữ liệu thực tế

Nhập dữ liệu văn bản mới từ người dùng thông qua input().

Sử dụng hàm preprocess_text() để tiền xử lý dữ liệu văn bản mới (loại bỏ ký tự đặc biệt, chuyển thành chữ thường, xóa stopwords).

Chuyển văn bản đã xử lý thành dạng vector TF-IDF bằng vectorizer.transform().

Dự đoán kết quả bằng mô hình đã huấn luyện (Decision Tree hoặc Logistic Regression).

Hiển thị kết quả phân loại: Tin thật (1) hoặc Tin giả (0).

Kiểm tra dữ liệu thực tế

```
# Test thử mô hình đã huấn luyện
test_data = input("Nhập dữ liệu cần kiểm tra: ")
test_data = preprocess_text([test_data])
test_data = vectorizer.transform(test_data)
result = decision_model.predict(test_data)
if result[0] == 1:
    print("Dữ liệu này là tin thật !")
else:
    print("Dữ liệu này là tin giả !")
```

```
100%|██████████| 1/1 [00:00<00:00, 24.49it/s]
Dữ liệu này là tin thật !
```

Kết quả và phân tích

Trực quan hóa giúp hiểu rõ hơn về các từ quan trọng trong mỗi lớp (tin thật và tin giả).

Logistic Regression thường ổn định và ít overfitting hơn Decision Tree, nhưng có thể không đạt hiệu suất cao trên dữ liệu phức tạp.

Decision Tree có thể bị overfit nếu không được kiểm soát, đặc biệt là khi độ chính xác trên tập huấn luyện cao nhưng thấp trên tập kiểm tra.

Ma trận nhầm lẫn cung cấp thông tin chi tiết về các loại lỗi (false positives/negatives) mà mô hình gặp phải.

Kiểm tra dữ liệu thực tế giúp đánh giá khả năng của mô hình khi xử lý văn bản mới, chưa được thấy trước đó.

Cải tiến và hướng phát triển

Tăng chất lượng dữ liệu: Thu thập thêm dữ liệu và đảm bảo dữ liệu đa dạng, chính xác.

Tối ưu hóa quy trình tiền xử lý: Nâng cao hiệu suất xử lý ngôn ngữ tự nhiên để cải thiện chất lượng đầu vào cho mô hình.

Sử dụng các mô hình tiên tiến hơn: Tận dụng các mô hình machine learning mạnh mẽ như ensemble learning hoặc deep learning.

Biểu diễn văn bản tốt hơn: Sử dụng Word Embeddings hoặc các mô hình transformer như BERT để biểu diễn văn bản.

Đánh giá mô hình toàn diện: Sử dụng nhiều chỉ số và kỹ thuật đánh giá để hiểu rõ hơn về hiệu suất của mô hình.

Giám sát sau triển khai: Đảm bảo hệ thống luôn được cải tiến và thích ứng với dữ liệu mới.