TRƯỜNG ĐẠI HỌC SÀI GÒN

KHOA CÔNG NGHỆ THÔNG TIN

ĐỀ TÀI ĐỒ ÁN:

PHÁT HIỆN TIN GIẢ BẰNG XỬ LÝ NGÔN NGỮ TỰ NHIÊN

Môn học: Tính toán thông minh

Giảng viên hướng dẫn: **Huỳnh Minh Trí**

Nhóm lớp học: **Nhóm 12**

Sinh viên:	**Lý Minh Phát**	**MSSV: 3122410294**
	Nguyễn Văn An	**MSSV: 3122410004**
	Nguyễn Quang Đạt	**MSSV: 3122410071**
	Nguyễn Văn Tầm Hoan	**MSSV: 3122410122**
	Trần Đăng Khôi	**MSSV: 3124410166**

TP.HCM, ngày 3 tháng 12 năm 2025

MỤC LỤC

LỜI MỞ ĐẦU

Trong thời đại công nghệ thông tin phát triển mạnh mẽ, việc lan truyền thông tin trên Internet trở nên phổ biến và nhanh chóng. Tuy nhiên, điều này cũng kéo theo một vấn nạn nghiêm trọng – đó là sự xuất hiện của các tin tức giả mạo (Fake News). Các thông tin sai lệch có thể gây ra hậu quả tiêu cực đối với xã hội, chính trị, kinh tế và niềm tin của người dân. Vì vậy, việc phát triển một hệ thống có khả năng tự động phát hiện và phân loại tin giả là một nhu cầu cấp thiết.

Đề tài "Phát hiện tin giả bằng xử lý ngôn ngữ tự nhiên nhằm mục đích nghiên cứu và áp dụng các kỹ thuật của Natural Language Processing (NLP) kết hợp với các mô hình học máy (Machine Learning)" để xây dựng một mô hình có khả năng nhận diện tin thật và tin giả một cách chính xác.

PHẦN 1 : CƠ SỞ LÝ THUYẾT

1.1. Xử lý ngôn ngữ tự nhiên (NLP)

Xử lý ngôn ngữ tự nhiên (natural language processing - NLP) là một nhánh của trí tuệ nhân tạo tập trung vào các ứng dụng trên ngôn ngữ của con người. Trong trí tuệ nhân tạo thì xử lý ngôn ngữ tự nhiên là một trong những phần khó nhất vì nó liên quan đến việc phải hiểu ý nghĩa ngôn ngữ-công cụ hoàn hảo nhất của tư duy và giao tiếp. Không những vậy còn là lĩnh vực giúp máy tính hiểu, phân tích và xử lý ngôn ngữ của con người. Trong bài toán phát hiện tin giả, NLP đóng vai trò quan trọng trong việc chuyển đổi nội dung văn bản thành dạng dữ liệu có thể học được bởi các mô hình.

1.2. Các kỹ thuật biểu diễn văn bản:

1.2.1. GIỚI THIỆU

Trong lĩnh vực xử lý ngôn ngữ tự nhiên (Natural Language Processing - NLP), việc chuyển đổi văn bản từ dạng dữ liệu không cấu trúc sang biểu diễn số học đóng vai trò nền tảng, cho phép các mô hình học máy xử lý, phân tích và rút trích thông tin hiệu quả. Báo cáo này tập trung vào kỹ thuật TF-IDF (Term Frequency - Inverse Document Frequency), đại diện cho sự tiến hóa từ các phương pháp biểu diễn rời rạc đơn giản đến các mô hình liên tục phức tạp hơn. Những kỹ thuật này không chỉ nâng cao hiệu suất trong các nhiệm vụ cốt lõi như phân loại văn bản, tìm kiếm thông tin và hiểu ngữ nghĩa, mà còn hỗ trợ các ứng dụng thực tế trong các hệ thống tiên tiến.

Phân tích trong báo cáo sẽ bao gồm: định nghĩa chi tiết, cơ chế hoạt động, ưu nhược điểm, và ứng dụng kỹ thuật TF-IDF vào thực tiễn để giải quyết bài toán phân loại tin giả.

Văn bản, với bản chất là dữ liệu không cấu trúc gồm chuỗi ký tự, tạo ra thách thức lớn cho máy tính khi xử lý trực tiếp mà không có sự chuyển đổi. Các kỹ thuật biểu diễn văn bản giải quyết vấn đề này bằng cách biến đổi chuỗi ký tự thành các vector số học, từ đó cho phép áp dụng các thuật toán toán học và học máy một cách hiệu quả. Quá trình này là bước đầu tiên quan trọng trong chuỗi xử lý NLP, giúp mô hình học được các mẫu ngữ nghĩa và ngữ cảnh từ dữ liệu văn bản.

Những kỹ thuật biểu diễn này đã được áp dụng rộng rãi trong các hệ thống thực tế, chẳng hạn như:

- Công cụ tìm kiếm (ví dụ: Google Search), nơi văn bản được biểu diễn để tối ưu hóa kết quả tìm kiếm dựa trên độ liên quan.
- Phân tích cảm xúc (sentiment analysis), giúp xác định ý kiến tích cực hoặc tiêu cực từ dữ liệu người dùng.
- Mô hình ngôn ngữ lớn (ví dụ: BERT hoặc GPT), nơi biểu diễn vector hỗ trợ học sâu và hiểu ngữ nghĩa phức tạp.

1.2.2. TF-IDF (Term Frequency - Inverse Document Frequency):

TF-IDF là kỹ thuật cải tiến từ BoW, gán trọng số cho từng từ dựa trên tần suất xuất hiện trong văn bản cụ thể (TF) và mức độ đặc trưng của từ đó trong toàn bộ corpus (IDF). Mục tiêu là nhấn mạnh từ quan trọng, giảm ảnh hưởng của từ phổ biến. Và đây sẽ là kỹ thuật nhóm dùng trong đề tài này để áp dụng trong quá trình tiền xử lí dữ liệu.

TF-IDF là một kỹ thuật thống kê đo lường tầm quan trọng của một từ trong một tài liệu cụ thể so với toàn bộ tập tài liệu. Nó kết hợp hai thành phần chính:

- **Term Frequency (TF)**: Tần suất xuất hiện của từ trong tài liệu, phản ánh mức độ liên quan cục bộ.
- **Inverse Document Frequency (IDF)**: Nghịch đảo tần suất xuất hiện của từ trong toàn bộ tập tài liệu, giúp giảm trọng số cho các từ phổ biến (như "the" hoặc "and") và tăng trọng số cho các từ đặc trưng.

1.2.3.Cơ chế hoạt động của TF-IDF:

TF-IDF hoạt động bằng cách:

1. Tính toán TF: Đếm số lần từ xuất hiện trong tài liệu và chuẩn hóa (ví dụ: chia cho tổng số từ trong tài liệu).
2. Tính toán IDF: Đánh giá độ hiếm của từ trên toàn bộ corpus.
3. Nhân hai giá trị để tạo vector biểu diễn, nơi mỗi chiều đại diện cho một từ trong từ điển.

Kỹ thuật này chuyển đổi văn bản thành ma trận vector thưa (sparse matrix), phù hợp cho các thuật toán như SVM hoặc Naive Bayes.

1.3. Các mô hình học máy được sử dụng:

Sau khi biểu diễn văn bản bằng TF-IDF, các mô hình phân loại được sử dụng để dự đoán nhãn (label) cho văn bản, chẳng hạn như phân loại cảm xúc (positive/negative) hoặc chủ đề (news/sport), đặc biệt là trong chủ đề phát hiện tin giả . Báo cáo này phân tích bốn mô hình phổ biến: Logistic Regression (dễ triển khai với TF-IDF), Decision Tree, Random Forest và KNN (K-Nearest Neighbors). Chúng được áp dụng rộng rãi trong phân loại văn bản, phân tích sẽ nhấn mạnh cách kết hợp chúng với các kỹ thuật biểu diễn từ phần trước.

Phân loại văn bản là nhiệm vụ cốt lõi trong NLP, nơi mô hình học từ dữ liệu có nhãn để dự đoán lớp cho văn bản mới. Logistic Regression và Decision Tree thường kết hợp với TF-IDF để xử lý vector thưa, Vì vậy nhóm đã kết hợp việc biểu diễn văn bản bằng TF-IDF và sử dụng các mô hình phân loại để dự đoán nhãn tin giả hoặc tin thật trong bài toán này

1.3.1 Logistic Regression:

Logistic Regression là một phương pháp phân loại tuyến tính dựa trên mô hình xác suất, trong đó xác suất đầu ra được mô hình hóa bằng hàm sigmoid của một tổ hợp tuyến tính các đặc trưng. Nhờ đặc tính tuyến tính và khả năng tối ưu hiệu quả trong không gian đặc trưng có chiều lớn, mô hình này đặc biệt phù hợp với dữ liệu văn bản được biểu diễn bằng TF-IDF, vốn có tính thưa cao. Ngoài ra, Logistic Regression dễ huấn luyện, ít nguy cơ overfitting khi kết hợp với các kỹ thuật regularization và thường được sử dụng làm mô hình cơ sở trong phân loại văn bản.

1.3.2 Decision Tree:

Decision Tree là một mô hình học máy có giám sát, xây dựng bằng cách phân chia không gian đặc trưng thành các vùng đồng nhất thông qua một chuỗi các quyết định dạng if–then. Tại mỗi nút, thuật toán lựa chọn đặc trưng và ngưỡng chia tối

ưu dựa trên các tiêu chí như Information Gain hoặc Gini Index, nhằm tăng độ thuần khiết của dữ liệu tại các nút con. Decision Tree có khả năng áp dụng cho cả bài toán phân loại và hồi quy, đồng thời cho phép diễn giải trực quan quá trình ra quyết định.

1.3.3 Random Forest:

Random Forest, được đề xuất bởi Breiman (2001), là một phương pháp ensemble dựa trên tập hợp nhiều cây quyết định không tương quan. Thuật toán sử dụng kỹ thuật bootstrap sampling để tạo các tập huấn luyện khác nhau cho mỗi cây, đồng thời lựa chọn ngẫu nhiên một tập con đặc trưng tại mỗi nút chia. Cách tiếp cận này làm giảm sự phụ thuộc giữa các cây thành viên, từ đó cải thiện hiệu năng dự đoán và giảm overfitting so với các mô hình cây quyết định đơn lẻ..

1.3.4 KNN (K-Nearest Neighbors):

K-Nearest Neighbors (KNN) là một thuật toán học máy dựa trên instance, trong đó quá trình dự đoán không yêu cầu huấn luyện mô hình rõ ràng. Thuật toán phân loại một điểm dữ liệu mới bằng cách xác định K điểm dữ liệu gần nhất theo một thước đo khoảng cách nhất định (như Euclidean), sau đó gán nhãn theo nguyên tắc bỏ phiếu đa số. Hiệu năng của KNN phụ thuộc mạnh vào giá trị K và cách chuẩn hóa dữ liệu đầu vào.

PHẦN 2 : PHƯƠNG PHÁP THỰC HIỆN

2.1) Thu thập dữ liệu:

Nguồn dữ liệu: Dataset Fake News từ Kaggle

Kích thước: 23,196 mẫu dữ liệu.

Cấu trúc dữ liệu: gồm các cột

- title: Tiêu đề bài viết.
- source_domain: Tên miền nguồn tin.
- real: Nhãn (Label) - 1 là Tin thật, 0 là Tin giả.

Phân phối nhãn: Dữ liệu có sự mất cân bằng nhẹ (Tin thật: khoảng 17,441 mẫu, Tin giả: khoảng 5,755 mẫu). Nhóm đã sử dụng kỹ thuật Class Weight (trọng số lớp) trong quá trình huấn luyện để xử lý vấn đề này.

Lý do chọn dataset này:

1. Dataset có quy mô đủ lớn và đa dạng để huấn luyện mô hình phân loại văn bản.

2. Được cộng đồng đánh giá cao, giúp đảm bảo chất lượng và tính thực tiễn.

3. Cấu trúc dữ liệu rõ ràng, dễ sử dụng cho các mô hình machine learning và deep learning.

4. Phù hợp trực tiếp với mục tiêu của nhóm: phân loại tin thật – tin giả.

Bối cảnh bài toán:

- Phát hiện tin giả (Fake News Detection) là một bài toán thuộc nhánh Text Classification trong xử lý ngôn ngữ tự nhiên (NLP). Nhiệm vụ chính là phân tích nội dung bài báo hoặc đoạn văn để xác định tính xác thực của thông tin. Việc lựa chọn dataset này giúp nhóm triển khai đầy đủ quy trình: tiền xử lý văn bản, biểu diễn dữ liệu, huấn luyện mô hình và đánh giá hiệu quả.

2.2. Tiền xử lý văn bản (Preprocessing)

Mục tiêu: Làm cho dữ liệu văn bản "sạch" và "thống nhất" để mô hình hiểu được. Khi lấy dữ liệu từ Kaggle, văn bản có thể chứa ký tự lạ, dấu câu, từ vô nghĩa, viết hoa, emoji,… Dễ làm máy tính không hiểu, nên phải tiền xử lý lại.

Các bước thực hiện:

1. Kết hợp đặc trưng:

Tạo cột dữ liệu mới combined bằng cách ghép title (tiêu đề) và source_domain (nguồn tin). Lý do là tên miền và cách đặt tiêu đề thường chứa thông tin quan trọng để xác định tin giả.

2. Làm sạch văn bản (Text Cleaning):

- **Lowercase:** Chuyển toàn bộ văn bản về chữ thường.
- **Remove Punctuation:** Loại bỏ các dấu câu và ký tự đặc biệt bằng Regex (re.sub(r'[^\w\s]', '', sentence)).
- **Remove Stopwords:** Loại bỏ các từ dừng (stopwords) tiếng Anh phổ biến (như "the", "is", "at", "which") không mang ý nghĩa phân loại.

3. **Chia tập dữ liệu:**

Dữ liệu được chia theo tỉ lệ 75% Train (Huấn luyện) và 25% Test (Kiểm thử). Tham số stratify=y được sử dụng để đảm bảo tỉ lệ tin thật/giả trong tập Train và Test là tương đồng.

4. Trích xuất đặc trưng (Feature Extraction)

Sử dụng TfidfVectorizer từ thư viện Scikit-learn để chuyển đổi cột văn bản đã làm sạch thành ma trận số.

- fit_transform trên tập Train.
- transform trên tập Test (để đảm bảo tính khách quan, không để lộ thông tin từ tập Test).

Sau khi tiền xử lý, văn bản được chuyển sang vector TF-IDF, sau đó huấn luyện bằng Logistic Regression và Random Forest.

PHẦN 3 : THỰC HIỆN VÀ KẾT QUẢ

Nhóm đã tiến hành huấn luyện và đánh giá 4 mô hình. Kết quả chi tiết như sau:

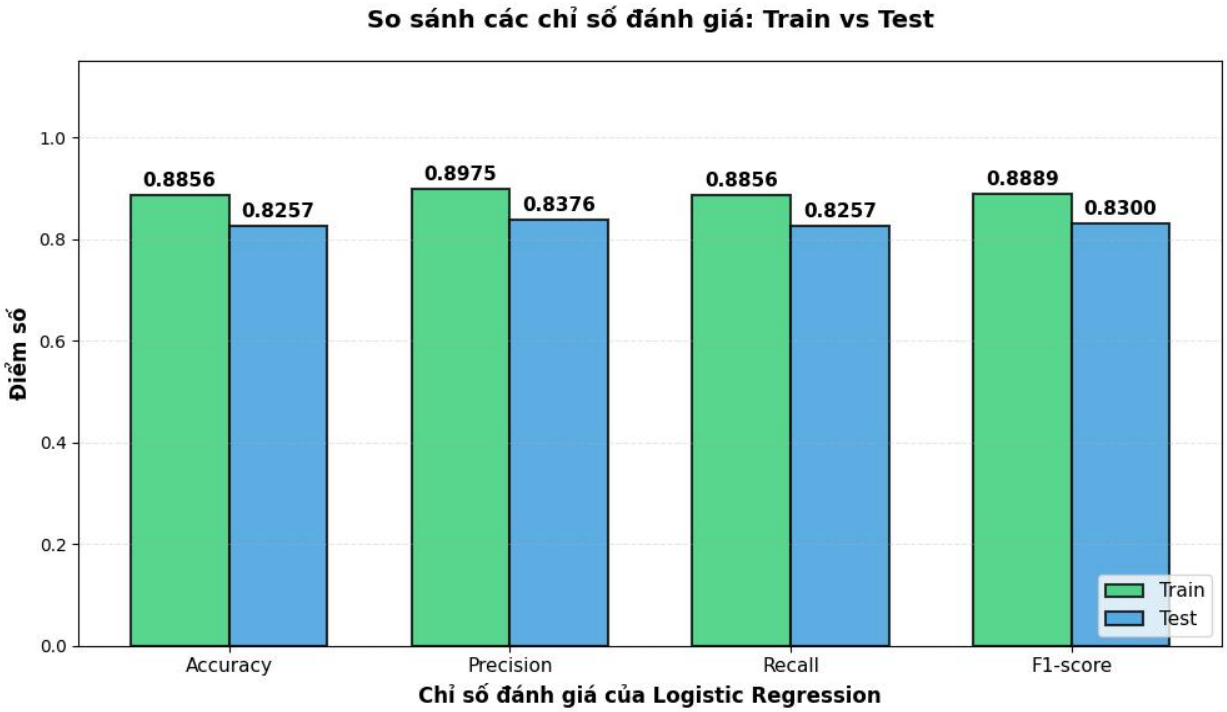
a. Logistic Regression

Là mô hình thống kê tuyến tính được dùng nhiều trong phân loại nhị phân (hai nhãn).
Nó hoạt động bằng cách tính xác suất một văn bản thuộc về một lớp nào đó dựa trên trọng số của các đặc trưng.
Ưu điểm: đơn giản, dễ huấn luyện, hiệu quả với dữ liệu TF-IDF.

Cấu hình: Solver='saga', Class_weight='balanced' (cân bằng trọng số giữa tin thật và giả).

Kết quả:

- ✓ **Accuracy (Train):** ~88.56%
- ✓ **Accuracy (Test): ~82.57%**
- ✓ **F1-Score (Test):** ~83.00%



Nhận xét: Mô hình có độ ổn định cao. Chênh lệch giữa tập Train và Test thấp (~6%), cho thấy mô hình **không bị Overfitting** và có khả năng tổng quát hóa tốt trên dữ liệu mới

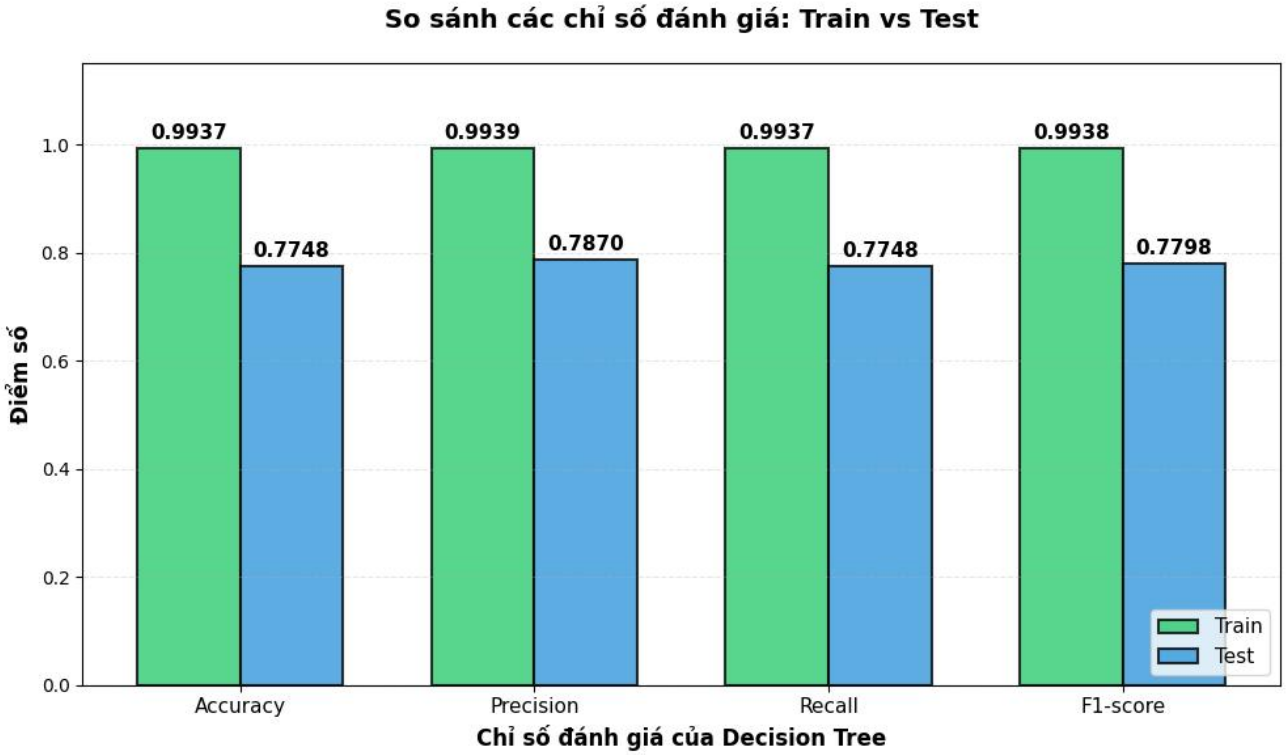
b. Decision Tree

Là mô hình dựa trên tập hợp nhiều cây quyết định.
Mỗi cây học một phần dữ liệu khác nhau, và kết quả cuối cùng là sự tổng hợp của các cây.
Ưu điểm: độ chính xác cao, giảm nguy cơ quá khớp, hoạt động tốt với dữ liệu có nhiều đặc trưng.

· **Cấu hình:** Sử dụng Class_weight='balanced'.

Kết quả:

✓ **Accuracy (Train):** ~99.37% (Gần như tuyệt đối).
✓ **Accuracy (Test): ~77.48%**



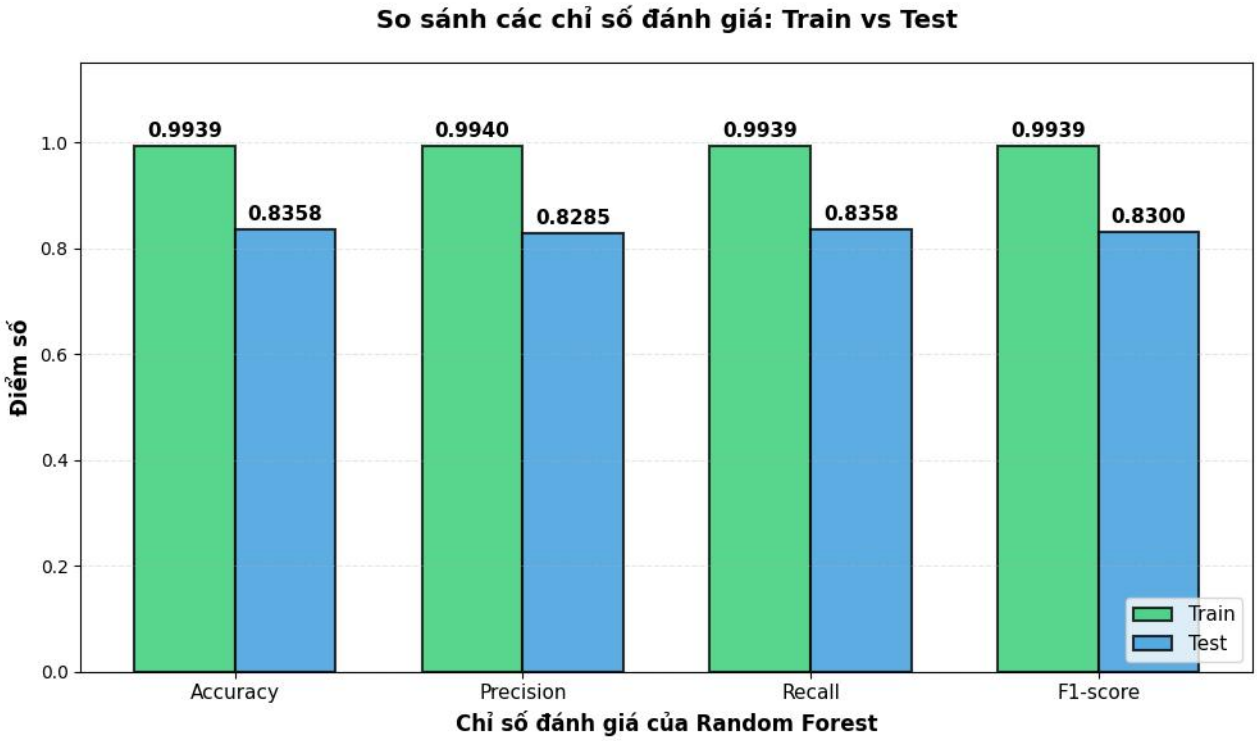
Nhận xét: Có sự chênh lệch rất lớn giữa Train và Test (>20%). Mô hình bị **Overfitting nặng** (học vẹt dữ liệu huấn luyện) nên hoạt động kém hiệu quả trên dữ liệu kiểm thử. Đây là mô hình kém trong thực tế.

c. Random Forest

Cấu hình: n_estimators=200 (200 cây quyết định), min_samples_split=5, Class_weight='balanced'.

Kết quả:

- ✓ **Accuracy (Train):** ~99.39%
- ✓ **Accuracy (Test): ~83.58%**



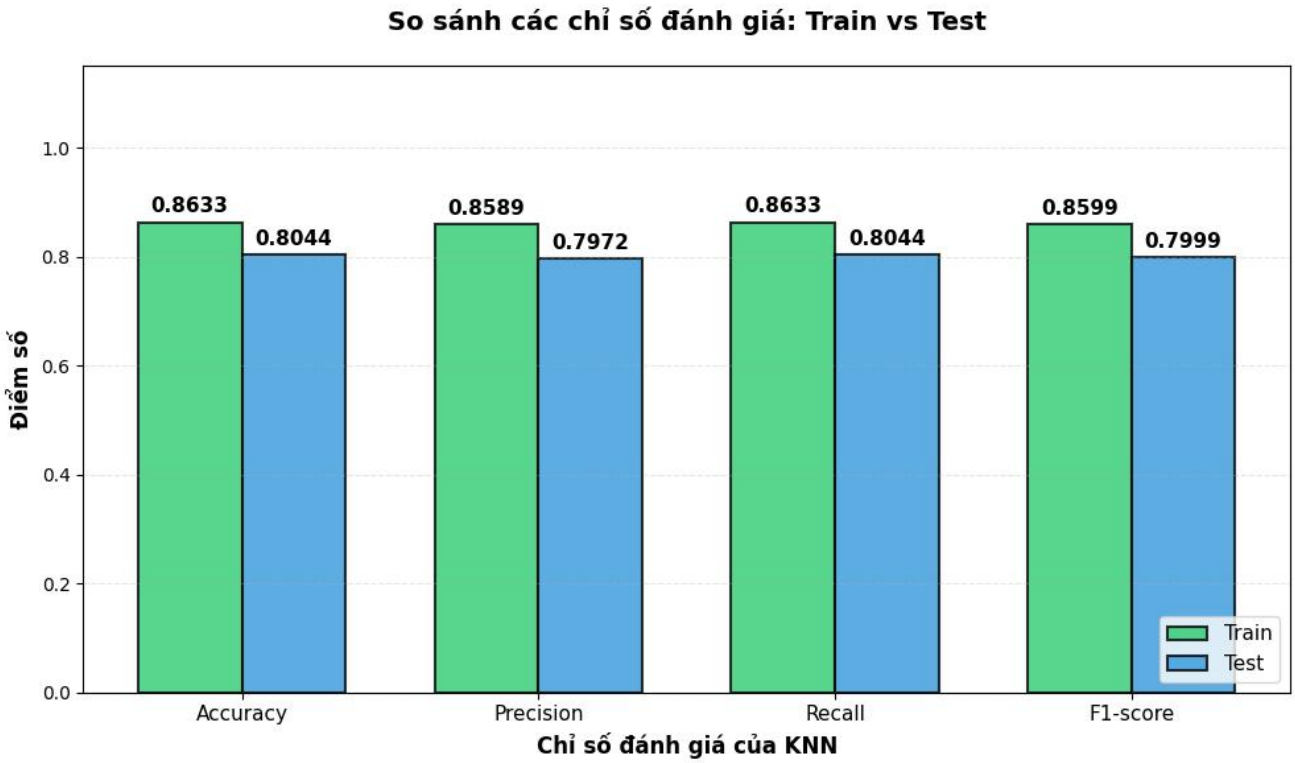
Nhận xét: Random Forest cải thiện đáng kể so với Decision Tree nhờ cơ chế "bỏ phiếu" của nhiều cây. Độ chính xác trên tập Test cao nhất trong các mô hình (~84%), tuy nhiên vẫn tồn tại dấu hiệu Overfitting (Train ~100% vs Test ~84%)

d. KNN(K-Nearest Neighbors)

Cấu hình: n_neighbors=5.

Kết quả:

- ✓ **Accuracy (Train):** ~86.33%
- ✓ **Accuracy (Test): ~80.44%**

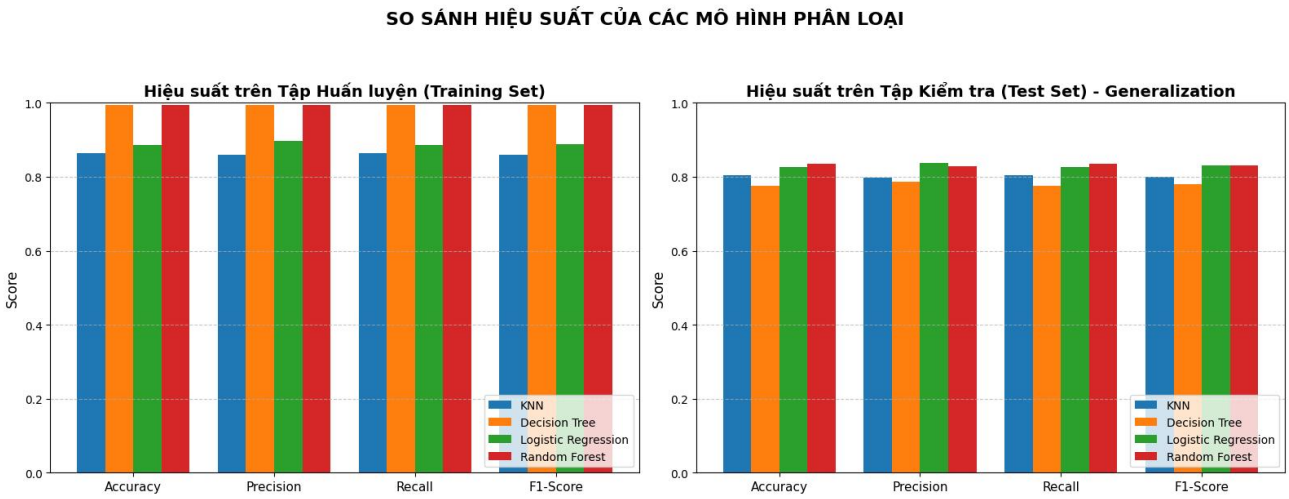


Nhận xét: Hiệu suất ở mức trung bình, thấp hơn Logistic Regression và Random Forest. Ngoài ra, KNN tốn nhiều tài nguyên tính toán khi dự đoán với dữ liệu lớn.

e. Bảng so sánh trên tập test:

Mô hình	Accuracy	Precision	Recall	F1-score
Logistic Regression	0.8257	0.8376	0.8257	0.8300
Decision Tree	0.7748	0.7870	0.7748	0.7798
Random Forest	0.8358	0.8285	0.8358	0.8300
KNN (K-Nearest Neighbors)	0.8044	0.7972	0.8044	0.7999

f. Biểu đồ so sánh 4 mô hình:



PHẦN 4:ĐÁNH GIÁ VÀ THẢO LUẬN

Sau khi huấn luyện, mô hình phân loại văn bản được **đánh giá độ chính xác** dựa trên các chỉ số thống kê nhằm phản ánh khả năng dự đoán của mô hình. Các chỉ số thường dùng gồm:

- **Precision (Độ chính xác)**: cho biết trong số các văn bản mà mô hình dự đoán thuộc một lớp nào đó, có bao nhiêu văn bản thực sự thuộc lớp đó. Precision cao nghĩa là mô hình ít dự đoán sai dương tính.
- **Recall (Độ bao phủ)**: cho biết trong số các văn bản thực sự thuộc một lớp, mô hình nhận diện được bao nhiêu. Recall cao nghĩa là mô hình ít bỏ sót mẫu đúng.
- **F1-score**: là giá trị trung bình điều hòa giữa Precision và Recall, phản ánh mức cân bằng giữa hai chỉ số này. F1-score càng cao thì mô hình càng tốt.

=>Ba chỉ số trên giúp đánh giá mô hình một cách toàn diện hơn so với chỉ dùng Accuracy, đặc biệt trong các trường hợp dữ liệu không cân bằng giữa các lớp.

Dựa trên biểu đồ so sánh và các chỉ số (Accuracy, Precision, Recall, F1-Score), chúng em rút ra các kết luận sau:

Về hiện tượng Overfitting:

Decision Tree là mô hình tệ nhất về mặt tổng quát hóa. Nó ghi nhớ dữ liệu Train nhưng thất bại trên Test.

Random Forest có độ chính xác cao nhất nhưng vẫn bị Overfitting nhẹ.

Về sự cân bằng (Stability):

Logistic Regression là mô hình ổn định nhất. Mặc dù độ chính xác trên tập Test (82.57%) thấp hơn một chút so với Random Forest (83.58%), nhưng độ chênh lệch giữa Train/Test rất nhỏ. Điều này chứng tỏ mô hình thực sự "học" được quy luật của ngôn ngữ thay vì học thuộc lòng.

Hơn nữa, Logistic Regression tính toán rất nhanh, phù hợp cho các ứng dụng thời gian thực.

Lựa chọn mô hình tối ưu:

Xét trên tiêu chí cân bằng giữa Hiệu suất (Performance) và Khả năng tổng quát hóa (Generalization), Logistic Regression là lựa chọn an toàn và hiệu quả nhất cho bài toán này với tập dữ liệu hiện tại.

Nếu ưu tiên độ chính xác tuyệt đối và chấp nhận rủi ro Overfitting nhẹ, Random Forest là phương án thứ hai.

Kết luận:

Các chỉ số Precision, Recall và F1-score cung cấp cái nhìn chi tiết hơn về hiệu suất của mô hình phân loại so với chỉ số Accuracy thông thường.
Khi so sánh giữa các mô hình, mô hình nào có các giá trị Precision, Recall và F1-score cao hơn thì mô hình đó thể hiện khả năng dự đoán chính xác và ổn định hơn.

PHẦN 5: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

5.1.Kết luận

Đề tài đã xây dựng thành công quy trình phát hiện tin giả từ khâu tiền xử lý, trích xuất đặc trưng TF-IDF đến huấn luyện và đánh giá mô hình. Kết quả thực nghiệm cho thấy việc kết hợp các đặc trưng văn bản với các mô hình học máy truyền thống có thể đạt độ chính xác lên tới **~84%**. Logistic Regression và Random Forest là hai thuật toán phù hợp nhất cho bài toán này.

5.2.Hướng phát triển

· **Mô hình Deep Learning:** Áp dụng các mô hình học sâu hiện đại như LSTM (Long Short-Term Memory) hoặc Transformer (BERT, RoBERTa) để nắm bắt ngữ cảnh và ý nghĩa câu tốt hơn so với TF-IDF.

· **Hệ thống thời gian thực:** Tích hợp mô hình vào ứng dụng web hoặc extension trình duyệt để cảnh báo tin giả ngay khi người dùng đọc báo.

· **Đa ngôn ngữ:** Mở rộng bộ dữ liệu để hỗ trợ phát hiện tin giả tiếng Việt, phục vụ cộng đồng trong nước.

PHẦN 6: CODE MINH HỌA – PYTHON

6.1.Tải dataset từ kaggle

Dataset nhóm sẽ dùng là https://www.kaggle.com/datasets/algord/fake-news

```python
# Download the dataset from Kaggle
!kaggle datasets download algord/fake-news
# Unzip the dataset
# The dataset typically gets downloaded as a zip file in the current working directory
!unzip -q fake-news -d fake-news
print("Dataset downloaded and unzipped successfully!")
```

Python

```
Dataset URL: https://www.kaggle.com/datasets/algord/fake-news
License(s): CC0-1.0
```

6.2.Load dataset

```python
df = pd.read_csv("/content/fake-news/FakeNewsNet.csv")
print("Đã tải bộ dữ liệu đầy đủ từ file CSV.")
```

Python

6.3.Data preprocessing

6.3.1.Xử lí dữ liệu bị thiếu

```python
df.isnull().sum()
```

Python

	0
title	0
news_url	330
source_domain	330
tweet_num	0
real	0

dtype: int64

```python
# Drop the "news_url"
df = df.drop("news_url", axis =1)
```

Python

```python
# Handle the missing values in source_domain. replace with "Unkown"
df['source_domain'] = df['source_domain'].fillna("Unknown")
```

Python

6.3.2.Hàm tiền xử lý: chữ thường, loại punctuation, bỏ stopwords

```
stop_words = set([
    'i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "you've", "you'll", "you'd",
    'your', 'yours', 'yourself', 'yourselves', 'he', 'him', 'his', 'himself', 'she', "she's", 'her', 'hers',
    'herself', 'it', "it's", 'its', 'itself', 'they', 'them', 'their', 'theirs', 'themselves', 'what', 'which',
    'who', 'whom', 'this', 'that', "that'll", 'these', 'those', 'am', 'is', 'are', 'was', 'were', 'be', 'been',
    'being', 'have', 'has', 'had', 'having', 'do', 'does', 'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if',
    'or', 'because', 'as', 'until', 'while', 'of', 'at', 'by', 'for', 'with', 'about', 'against', 'between',
    'into', 'through', 'during', 'before', 'after', 'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out',
    'on', 'off', 'over', 'under', 'again', 'further', 'then', 'once', 'here', 'there', 'when', 'where', 'why',
    'how', 'all', 'any', 'both', 'each', 'few', 'more', 'most', 'other', 'some', 'such', 'no', 'nor', 'not',
    'only', 'own', 'same', 'so', 'than', 'too', 'very', 's', 't', 'can', 'will', 'just', 'don', "don't",
    'should', "should've", 'now', 'd', 'll', 'm', 'o', 're', 've', 'y', 'ain', 'aren', "aren't", 'couldn',
    "couldn't", 'didn', "didn't", 'doesn', "doesn't", 'hadn', "hadn't", 'hasn', "hasn't", 'haven', "haven't",
    'isn', "isn't", 'ma', 'mightn', "mightn't", 'mustn', "mustn't", 'needn', "needn't", 'shan', "shan't",
    'shouldn', "shouldn't", 'wasn', "wasn't", 'weren', "weren't", 'won', "won't", 'wouldn', "wouldn't"
])
```

Python

```
def preprocess_text(text_data):
    preprocessed_text = []
    for sentence in tqdm(text_data, desc="Đang xử lý văn bản"):
        # Loại bỏ dấu câu
        sentence = re.sub(r'[^\w\s]', '', str(sentence))
        # Chuyển thành chữ thường và loại bỏ từ dừng (stopwords)
        clean_sentence = ' '.join(token.lower() for token in sentence.split() if token.lower() not in stop_words)
        preprocessed_text.append(clean_sentence)
    return preprocessed_text
```

Python

Ý nghĩa của hàm:

1. Loại bỏ dấu câu (re.sub(r'[^\w\s]', '', str(sentence))).
2. Chuyển về chữ thường (token.lower()).
3. Loại bỏ stopwords.

6.4.Chuẩn bị data cho training

6.4.1.Chia tập train và tập test

Chia dữ liệu thành tập huấn luyện (75%) và tập kiểm tra (25%). Mục đích: Đảm bảo mô hình được đánh giá trên dữ liệu chưa từng thấy để kiểm tra khả năng khái quát hóa.

```
X = df["title"]
y = df["real"]
```

Python

```
# Divide the dataset into Train and Test
X_train , X_test ,y_train ,y_test = train_test_split(
    X,y, test_size=0.25 , random_state =42, stratify= y
)
```

Python

6.4.2.Tính toán class_weight cho mô hình khi dữ liệu mất cân bằng

Xử lí mất cân bằng lớp

```
df['real'].value_counts()
```

Python

real	count
1	17441
0	5755

dtype: int64

Tính toán class_weight

```python
import numpy as np
from sklearn.utils import class_weight
class_weights_vals = class_weight.compute_class_weight(
    class_weight='balanced',
    classes=np.unique(y_train),
    y=y_train
)
class_weights_dict = dict(enumerate(class_weights_vals))
print("Class weights:", class_weights_dict)
```

Python

```
Class weights: {0: np.float64(2.015407784986098), 1: np.float64(0.6649720969344851)}
```

6.4.3.Vector hóa dữ liệu (TF-IDF)

Mục đích: Chuyển đổi dữ liệu văn bản thành định dạng số (vector) mà mô hình học máy có thể xử lý được. TF-IDF (Term Frequency-Inverse Document Frequency) đánh giá mức độ quan trọng của một từ trong tài liệu so với toàn bộ tập dữ liệu, giúp các từ đặc trưng cho tin giả/tin thật có trọng số cao hơn.

```python
vectorization = TfidfVectorizer()
X_train_tfidf = vectorization.fit_transform(X_train)
X_test_tfidf = vectorization.transform(X_test)
```

Python

6.5.Huấn luyện mô hình

6.5.1.Logistic Regression

```python
lr_model = LogisticRegression(
    random_state=42,
    solver='saga',
    penalty='l2',
    max_iter=1000,
    n_jobs=-1,
    class_weight = class_weights_dict
)
lr_model.fit(X_train_tfidf, y_train)
```

Python

```
LogisticRegression(class_weight={0: np.float64(2.015407784986098),
                                 1: np.float64(0.6649720969344851)},
                   max_iter=1000, n_jobs=-1, random_state=42, solver='saga')
```

Đánh giá mô hình

```python
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, classification_report
# Dự đoán trên tập train và test
y_train_pred_lr = lr_model.predict(X_train_tfidf)
y_test_pred_lr = lr_model.predict(X_test_tfidf)
# 1. Accuracy
train_accuracy_lr = accuracy_score(y_train, y_train_pred_lr)
test_accuracy_lr  = accuracy_score(y_test,  y_test_pred_lr)
# 2. Precision, Recall, F1-score
# - macro: tính trung bình không trọng số (phù hợp khi các lớp cân bằng)
# - weighted: tính trung bình có trọng số theo số lượng mẫu mỗi lớp
# - micro: tương đương accuracy trong bài toán đa lớp
train_precision_lr = precision_score(y_train, y_train_pred_lr, average='weighted')
train_recall_lr    = recall_score(y_train, y_train_pred_lr, average='weighted')
train_f1_lr        = f1_score(y_train, y_train_pred_lr, average='weighted')
test_precision_lr  = precision_score(y_test, y_test_pred_lr, average='weighted')
test_recall_lr     = recall_score(y_test, y_test_pred_lr, average='weighted')
test_f1_lr         = f1_score(y_test, y_test_pred_lr, average='weighted')
```

```python
# In kết quả đẹp
print("=== ĐÁNH GIÁ MÔ HÌNH LOGISTIC REGRESSION ===")
print(f"{'':<15} {'Train':>12} {'Test':>12}")
print(f"{'Accuracy':<15} {train_accuracy_lr:>12.4f} {test_accuracy_lr:>12.4f}")
print(f"{'Precision':<15} {train_precision_lr:>12.4f} {test_precision_lr:>12.4f}")
print(f"{'Recall':<15} {train_recall_lr:>12.4f} {test_recall_lr:>12.4f}")
print(f"{'F1-score':<15} {train_f1_lr:>12.4f} {test_f1_lr:>12.4f}")
print("\n=== Classification Report trên tập Test ===")
print(classification_report(y_test, y_test_pred_lr))
```

Python

6.5.2.Decision Tree

```python
dt_model = DecisionTreeClassifier(
    random_state=42,
    class_weight = class_weights_dict
)
dt_model.fit(X_train_tfidf, y_train)
```

Python

```
DecisionTreeClassifier(class_weight={0: np.float64(2.015407784986098),
                                     1: np.float64(0.6649720969344851)},
                       random_state=42)
```

Code đánh giá tương tự logistic Regression chỉ thay đổi lại các biên , tham khảo tại file notebook của nhóm.

6.5.3.Random forest

```python
import matplotlib.pyplot as plt
from sklearn.ensemble import RandomForestClassifier
model = RandomForestClassifier(
    n_estimators=200,
    max_depth=None,
    max_features="sqrt",
    min_samples_split=5,
    random_state=42,
    class_weight = class_weights_dict
)
model.fit(X_train_tfidf, y_train)
```

Python

```
RandomForestClassifier(class_weight={0: np.float64(2.015407784986098),
                                     1: np.float64(0.6649720969344851)},
                       min_samples_split=5, n_estimators=200, random_state=42)
```

Code đánh giá tương tự logistic Regression chỉ thay đổi lại các biên , tham khảo tại file notebook của nhóm.

6.5.4.Nearest Neighbors

```python
from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier(
    n_neighbors=5,
    )
classifier.fit(X_train_tfidf, y_train)
```

Python

```
KNeighborsClassifier()
```

CÁC TÀI LIỆU THAM KHẢO

1. https://www.kaggle.com/datasets/algord/fake-news

2. https://www.geeksforgeeks.org/nlp/fake-news-detection-model-using-tensorflow-in-python/