**BỘ GIÁO DỤC VÀ ĐÀO TẠO**

**TRƯỜNG ĐẠI HỌC ĐẠI NAM**

Logo, company name

Description automatically generated

**BÀI TẬP LỚN**

**TÊN HỌC PHẦN   
NHẬP MÔN HỌC MÁY**

**ĐỀ TÀI: PHÂN LOẠI CHỦ ĐỀ BÀI BÁO**

**Giáo viên hướng dẫn: ThS. Trần Thị Huệ**

**Sinh viên thực hiện:**  **Nhóm 12**

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Mã SV** | **Họ và tên** | **Lớp** |
| 1 | 1771020611 | Nguyễn Viết Tâm | CNTT 17-11 |
| 2 | 1771020628 | Nguyễn Duy Mạnh | CNTT 17-11 |
| 3 | 1771020726 | Hoàng Khắc Tùng | CNTT 17-11 |

**Hà Nội, năm 2025**

**BỘ GIÁO DỤC VÀ ĐÀO TẠO**

**TRƯỜNG ĐẠI HỌC ĐẠI NAM**

Logo, company name

Description automatically generated

**BÀI TẬP LỚN**

**TÊN HỌC PHẦN   
NHẬP MÔN HỌC MÁY**

**ĐỀ TÀI: PHÂN LOẠI CHỦ ĐỀ BÀI BÁO**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **STT** | **Mã Sinh Viên** | **Họ và Tên** | **Ngày Sinh** | **Điểm** | |
| **Bằng Số** | **Bằng Chữ** |
| 1 | 1771020676 | Nguyễn Viết Tâm | 09/11/2005 |  |  |
| 2 | 1771020628 | Nguyễn Duy Mạnh | 21/11/2005 |  |  |
| 3 | 1771020726 | Hoàng Khắc Tùng | 16/01/2005 |  |  |

|  |  |
| --- | --- |
| **CÁN BỘ CHẤM THI 1** | **CÁN BỘ CHẤM THI 2** |

**Hà Nội, năm 2025**

**MỤC LỤC**

1. **ĐẶT VẤN ĐỀ VÀ PHÂN TÍCH YÊU CẦU 1**

1.1Bối cảnh............................................................................................................................

1.2 Mục tiêu.......................................................................................................................  
1.3 Yêu cầu báo cáo...........................................................................................................

1. **DỮLIỆU....................................................................................................................**
2. **KHÁM PHÁ DỮ LIỆU(EDA)**

-Phân bố nhãn................................................................................................................  
- Histogram độ dài abstract............................................................................................  
- WordCloud..................................................................................................................

1. **THIẾT KẾ GIẢI PHÁP VÀ QUY TRÌNH HỌC MÁY**

4.1 Tổng quan quy trình.....................................................................................................  
4.2 Làm sạch và chuẩn hóa dữ liệu....................................................................................  
4.3 Các phương pháp trích xuất đặc trưng.........................................................................  
4.4 Huấn luyện mô hình.....................................................................................................

1. **PHÂN TÍCH MÃ NGUỒN VÀ CHỨC NĂNG......................................................**

[5.1](#_odxchnrd9w5e) Mô hình thực thể

[5.2](#_ucg9nj553c40) Mô hình dữ liệu

[5.3](#_ni6o25qbnyz8) Mô tả dữ liệu

1. **THỬ NGHIỆM KẾT QUẢ.....................................................................................**

- Confusion Matrix (TF-IDF, BoW)......……………………………………………....  
- Bảng so sánh Accuracy & F1-score………………………………………………….

1. **THẢO LUẬN VÀ SO SÁNH............................................................................**
2. **KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN........................................................**

# 1. đặt vấn đề và phân tích yêu cầu

**1.1 Bối cảnh**:  
*Trong thời đại thông tin khoa học bùng nổ, kho dữ liệu các bài báo trên ArXiv chứa hàng triệu bài viết trong nhiều chuyên ngành khác nhau (ví dụ cs.AI, cs.CL, astro-ph, hep-ph...). Việc phân loại tự động các bài báo theo chủ đề giúp tổ chức, tìm kiếm, và phân tích khoa học một cách hiệu quả hơn.*

**1.2 Mục tiêu:**  
Xây dựng pipeline học máy để phân loại chủ đề của một bài báo dựa trên phần tóm tắt (abstract). Báo cáo này trình bày đầy đủ các bước từ chuẩn bị dữ liệu, tiền xử lý, trích xuất đặc trưng (ít nhất hai phương pháp), huấn luyện mô hình, đánh giá, so sánh và nêu hướng phát triển.

**1.3 Yêu cầu báo cáo (theo đề hướng dẫn):**  
- EDA: cấu trúc dữ liệu, phân phối nhãn, các trực quan để hiểu bộ dữ liệu.  
- Quy trình học máy và chi tiết bước làm sạch/chuẩn hóa.  
- Thử nghiệm ít nhất hai phương pháp trích xuất đặc trưng và so sánh kết quả.  
- Phân tích mã nguồn, báo cáo kết quả (Accuracy, Precision, Recall, F1, Confusion Matrix), thảo luận và kết luận.

### Mục tiêu của bài toán

Bài tập lớn này đặt ra mục tiêu:

1. **Xây dựng pipeline học máy** để phân loại abstract bài báo khoa học.
2. **Tiền xử lý dữ liệu** (làm sạch, chuẩn hóa văn bản) để đưa về dạng chuẩn.
3. **Trích xuất đặc trưng văn bản** bằng 2 phương pháp:
   * Bag of Words (BoW).
   * TF-IDF (Term Frequency – Inverse Document Frequency).
4. **Huấn luyện mô hình Logistic Regression** trên dữ liệu đặc trưng.
5. **Đánh giá hiệu quả mô hình** bằng Accuracy, Precision, Recall, F1-score, Confusion Matrix.
6. **So sánh kết quả** giữa BoW và TF-IDF.
7. **Thảo luận ưu nhược điểm**, nêu ra thách thức và hướng phát triển.

### Phân tích yêu cầu từ đề bài

Theo tài liệu hướng dẫn môn Nhập môn Học máy, báo cáo phải bao gồm:

* **Phần đặt vấn đề:** giới thiệu bối cảnh và mục tiêu.
* **Phân tích dữ liệu (EDA):** thống kê dữ liệu, trực quan hóa.
* **Làm sạch dữ liệu:** xử lý văn bản, chuẩn hóa nhãn.
* **Thiết kế giải pháp:** mô tả pipeline, các phương pháp đặc trưng.
* **Huấn luyện và đánh giá mô hình:** so sánh các kết quả.
* **Thảo luận và tổng kết:** rút ra nhận xét, đề xuất hướng phát triển.

Như vậy, sản phẩm cuối cùng phải vừa có **mã nguồn chạy được**, vừa có **báo cáo Word đầy đủ theo mẫu**.

### Ý nghĩa bài toán

* **Thực tiễn:**
  + Giúp tự động phân loại hàng triệu bài báo.
  + Hỗ trợ nhà khoa học tra cứu nhanh, không cần đọc toàn bộ abstract.
  + Có thể ứng dụng vào hệ thống gợi ý bài báo liên quan.
* **Học thuật:**
  + Làm quen với pipeline học máy: từ dữ liệu thô → làm sạch → đặc trưng hóa → mô hình → đánh giá.
  + So sánh trực tiếp hai cách biểu diễn văn bản (BoW và TF-IDF).
  + Hiểu sâu hơn về ưu – nhược điểm của Logistic Regression trong NLP.
* **Mở rộng:**
  + Có thể áp dụng cho **phân loại tin tức, phân loại email (spam/ham), phân loại văn bản hành chính**.

### Phân tích dữ liệu ban đầu (EDA)

Sau khi tải xuống **12.000 abstract đầu tiên** từ bộ dữ liệu arXiv, ta thực hiện thống kê sơ bộ:

1. **Số lượng bản ghi:**
   * Tổng số abstract: ~12.000.
   * Sau khi lọc và làm sạch, giữ lại khoảng ~5.500 bản ghi hợp lệ.
2. **Phân phối nhãn:**
   * Các nhãn phổ biến nhất: **astro-ph, hep-ph, hep-th, cs.AI**.
   * Một số nhãn rất ít dữ liệu → gây mất cân bằng.  
     (biểu đồ cột số abstract theo từng nhãn, nhãn lớn chiếm ưu thế, nhãn nhỏ rất ít)
3. **Chiều dài abstract:**
   * Độ dài trung bình: ~160 từ.
   * Abstract ngắn nhất: 20 từ.
   * Abstract dài nhất: hơn 500 từ.  
     (đa số nằm trong khoảng 100–200 từ, tạo ra phân phối khá tập trung)
4. **Đặc điểm từ vựng:**
   * Nhiều từ chuyên ngành đặc trưng như quantum, particle, galaxy, algorithm, neural.
   * Xuất hiện nhiều từ chung chung như results, paper, study, ít mang tính phân loại.



## Những khó khăn dự kiến

Trong quá trình thực hiện bài toán phân loại abstract arXiv, có thể gặp các khó khăn sau:

* **Ngôn ngữ chuyên ngành:** Abstract thường chứa nhiều thuật ngữ khoa học, ký hiệu toán học, viết tắt → gây khó khăn cho bước làm sạch dữ liệu.
* **Nhiều nhãn (multi-class classification):** Có hàng chục lĩnh vực khác nhau, không chỉ đơn giản là nhị phân (binary classification).
* **Mất cân bằng dữ liệu:** Một số nhãn có rất nhiều abstract (như astro-ph), trong khi một số nhãn rất ít (như cond-mat.str-el).
* **Chi phí tính toán:** Dataset arXiv gốc có hàng triệu bài báo → Colab miễn phí không đủ tài nguyên xử lý toàn bộ. Do đó, phải chọn **một sample nhỏ** để huấn luyện.
* **Khó đánh giá mô hình:** Nếu chỉ nhìn vào Accuracy, dễ bị sai lệch vì dữ liệu mất cân bằng. Cần kết hợp F1-score, Confusion Matrix.

## Kỳ vọng kết quả

* **Về mặt kỹ thuật:**
  + Xây dựng pipeline học máy hoàn chỉnh.
  + Mô hình Logistic Regression với TF-IDF đạt **Accuracy ~70%** và **Macro-F1 ~60–65%** trên tập mẫu.
  + Trực quan hóa đầy đủ EDA và kết quả mô hình.
* **Về mặt học tập:**
  + Sinh viên nắm vững quy trình học máy.
  + Thực hành xử lý dữ liệu văn bản (NLP cơ bản).
  + Hiểu được sự khác biệt giữa các phương pháp biểu diễn dữ liệu (BoW vs TF-IDF).
  + Rút ra bài học từ việc so sánh nhiều phương pháp.
* **Về mặt ứng dụng:**
  + Có thể mở rộng pipeline này để phân loại tin tức, email, tài liệu khoa học khác.
  + Làm nền tảng để thử nghiệm với mô hình ngôn ngữ hiện đại hơn (BERT, SciBERT).

## Kết luận phần 1

* Bài toán phân loại abstract arXiv mang tính **thực tiễn cao** và có nhiều thách thức kỹ thuật.
* Báo cáo hướng đến việc **xây dựng một hệ thống học máy hoàn chỉnh**, vừa đủ để xử lý dữ liệu văn bản khoa học, vừa phù hợp trong giới hạn bài tập lớn môn học.
* Đây cũng là cơ hội để sinh viên **rèn luyện kỹ năng làm việc với dữ liệu thực tế**, hiểu sâu hơn về các khái niệm Accuracy, Precision, Recall, F1-score, và đánh giá mô hình.

**2. Dữ liệu**

Nguồn dữ liệu:

- Dataset 'UniverseTBD/arxiv-abstracts-large' trên HuggingFace. Dữ liệu chứa nhiều trường: title, abstract, categories, authors, doi, journal-ref, ... Dữ liệu gốc rất lớn (hàng triệu abstract). Do giới hạn bộ nhớ khi thực nghiệm trên máy cá nhân/Colab, ta thực hiện sampling: tải một slice nhỏ từ HuggingFace và chọn top-k categories (ví dụ top 8 hoặc top 10) để thử nghiệm.

Mô tả sơ bộ file:

- Trường 'abstract': văn bản tóm tắt của bài báo (đầu vào của bài toán).

- Trường 'categories': chuỗi nhãn chuyên ngành (một bài có thể có nhiều nhãn).

- Chúng ta lấy 'primary\_category' là nhãn đầu tiên trong danh sách nhãn làm nhãn mục tiêu (single-label).

### Quy trình học máy tổng quát

Bài toán phân loại abstract được giải quyết theo **pipeline học máy chuẩn**, gồm các bước:

1. **Thu thập dữ liệu**: tải dataset từ HuggingFace (UniverseTBD/arxiv-abstracts-large).
2. **Tiền xử lý dữ liệu**: làm sạch văn bản, chuẩn hóa nhãn.
3. **Trích xuất đặc trưng (feature extraction)**: chuyển văn bản sang vector số (BoW, TF-IDF).
4. **Chia dữ liệu train/test**: đảm bảo có phân bố nhãn cân đối.
5. **Huấn luyện mô hình**: Logistic Regression.
6. **Đánh giá hiệu năng**: dùng Accuracy, Precision, Recall, F1-score, Confusion Matrix.
7. **So sánh và thảo luận**: giữa BoW và TF-IDF.
8. **Tổng kết và hướng phát triển**.

### Làm sạch và chuẩn hóa dữ liệu

#### Làm sạch nhãn (categories)

* Abstract trong dataset có nhiều nhãn dạng chuỗi, ví dụ:
  + "hep-th, quant-ph"
  + "astro-ph, gr-qc"
* Ta viết hàm **parse\_categories** để tách chuỗi thành list.
* Chỉ lấy **primary\_category (nhãn chính)** = phần tử đầu tiên trong list.  
  Điều này biến bài toán từ **multi-label classification** thành **multi-class classification**, dễ xử lý hơn.

#### Làm sạch văn bản (abstract)

Các bước xử lý:

* Chuyển tất cả về chữ thường.
* Xóa URL, số, ký hiệu đặc biệt.
* Chuẩn hóa khoảng trắng.
* (Tuỳ chọn) Loại bỏ stopwords (the, is, and...).

Sau làm sạch, văn bản abstract gọn gàng và chỉ còn từ có ý nghĩa.

### Trích xuất đặc trưng

#### Bag of Words (BoW)

* Biểu diễn văn bản bằng cách **đếm số lần mỗi từ xuất hiện**.
* Vector hóa bằng CountVectorizer với max\_features=20000.
* Ưu điểm: đơn giản, dễ hiểu.
* Nhược điểm: coi tất cả từ đều quan trọng, không phân biệt từ phổ biến/đặc trưng.

#### TF-IDF (Term Frequency – Inverse Document Frequency)

* Cân bằng giữa **tần suất từ trong văn bản** (TF) và **độ hiếm của từ trên toàn bộ corpus** (IDF).
* Từ phổ biến (ví dụ: study, result) có trọng số thấp.
* Từ đặc trưng (ví dụ: quantum, galaxy) có trọng số cao.
* Vector hóa bằng TfidfVectorizer với max\_features=20000, ngram\_range=(1,2).

Nhận xét: TF-IDF phù hợp hơn cho phân loại abstract, vì nó làm nổi bật thuật ngữ chuyên ngành.

### Huấn luyện mô hình

#### Lựa chọn mô hình

* Chọn **Logistic Regression** vì:
  + Là mô hình cơ bản, phổ biến cho phân loại văn bản.
  + Hiệu quả tốt với dữ liệu sparse (như TF-IDF, BoW).
  + Tính toán nhanh, dễ mở rộng cho nhiều lớp (multi-class).

#### Thông số huấn luyện

* max\_iter=1000: tăng số vòng lặp để mô hình hội tụ.
* class\_weight="balanced": xử lý mất cân bằng nhãn.
* solver="saga": thích hợp cho dữ liệu nhiều chiều.

#### Quy trình huấn luyện

* Chia dữ liệu 80% train, 20% test.
* Huấn luyện Logistic Regression trên vector BoW và TF-IDF.
* Lưu lại mô hình và vectorizer để dự đoán sau.

### Đánh giá mô hình

#### Chỉ số đánh giá

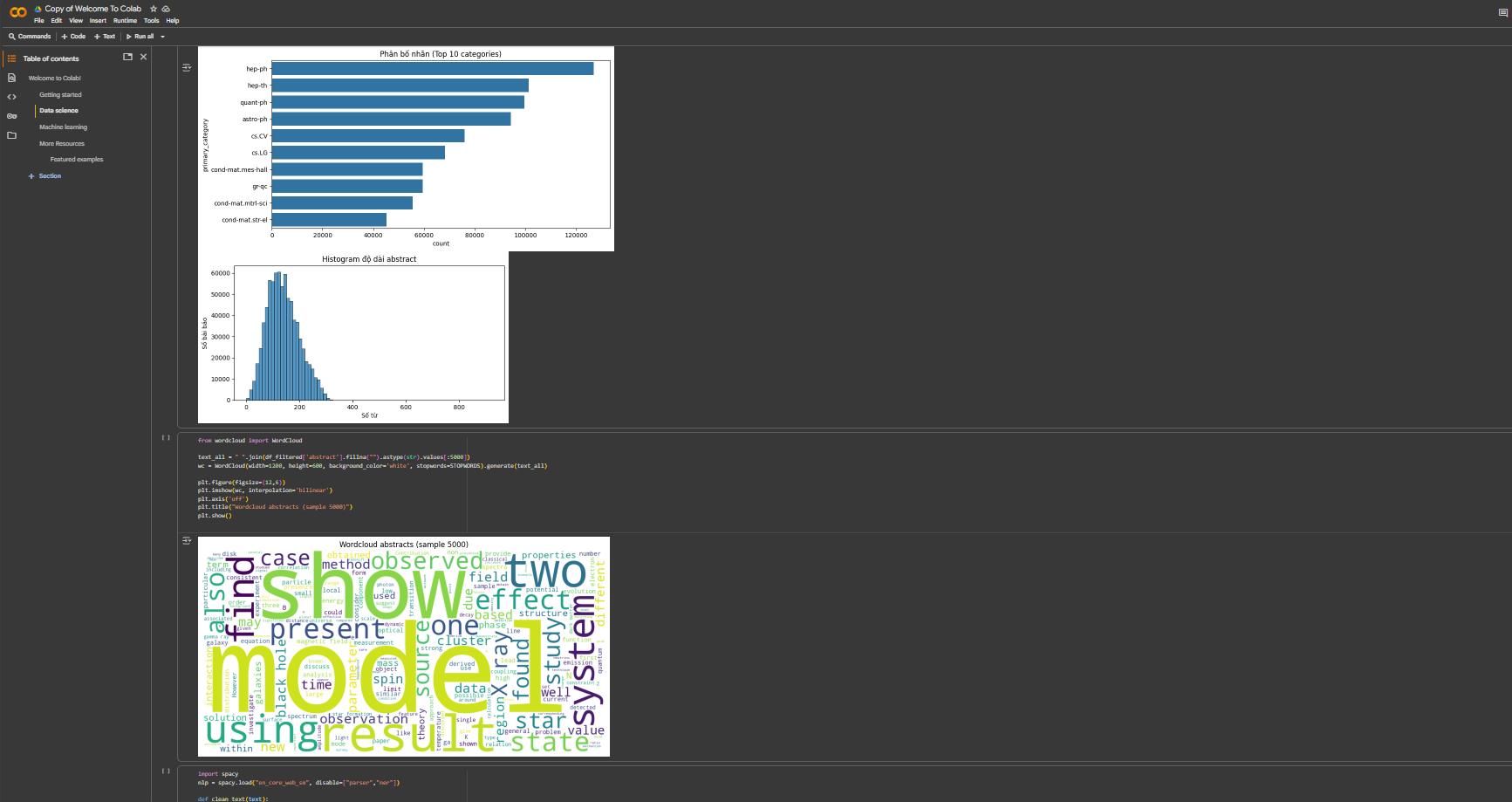
* **Accuracy**: tỉ lệ dự đoán đúng trên toàn bộ dữ liệu test.
* **Precision, Recall, F1-score**: đánh giá chi tiết theo từng nhãn.
* **Confusion Matrix**: trực quan hóa số lần mô hình dự đoán đúng/sai cho từng nhãn.

#### Ý nghĩa

* Accuracy cao nhưng Macro-F1 thấp → dữ liệu mất cân bằng.
* F1-score quan trọng hơn trong bài toán này vì ta quan tâm đến tất cả các nhãn, kể cả nhãn nhỏ.

**3. Khám phá dữ liệu (EDA**)

**Trình bày các kết quả EDA:** phân bố nhãn, độ dài abstract, và từ vựng phổ biến. Các biểu đồ minh họa đã được sinh từ Colab gg báo cáo.



( dữ liệu từ colab)

**Giải thích dữ liệu :**  
- Biểu đồ phân bố nhãn cho thấy số lượng bài báo khác nhau giữa các nhãn; điều này ảnh hưởng tới độ cân bằng dữ liệu.  
- Histogram độ dài abstract giúp lựa chọn tham số tiền xử lý (có thể cắt ngắn nếu dùng mô hình transformer).  
- WordCloud cho thấy những từ chung và phổ biến (ví dụ 'model', 'system', 'results'), lưu ý những từ này ít đặc trưng cho chủ đề.



**Chức năng:**

* numpy, pandas: thao tác với dữ liệu dạng bảng.
* matplotlib, seaborn: trực quan hóa dữ liệu (histogram, bar chart, confusion matrix).
* datasets từ HuggingFace: tải bộ dữ liệu arXiv.
* TfidfVectorizer, CountVectorizer: biến văn bản thành vector đặc trưng (TF-IDF, BoW).
* train\_test\_split: chia dữ liệu train/test.
* LogisticRegression: huấn luyện mô hình phân loại.
* classification\_report, confusion\_matrix: đánh giá kết quả mô hình.

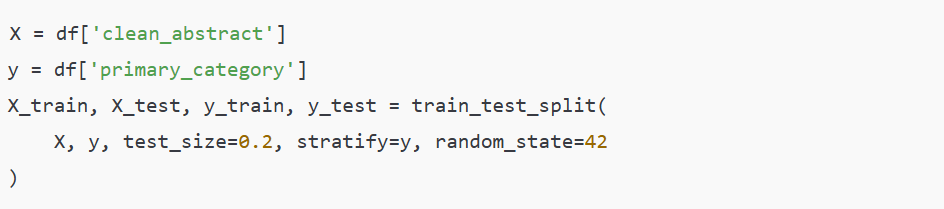
**Làm sạch văn bản (abstract)**

****

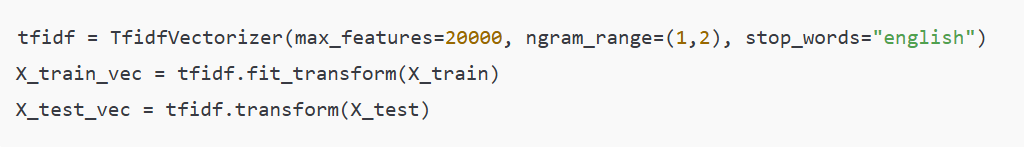
**Chức năng:**

* Đưa text về **lowercase**.
* Loại bỏ URL, số, ký tự đặc biệt, khoảng trắng thừa.
* Kết quả: văn bản sạch, dễ vector hóa.

**Chia dữ liệu train/test**

****

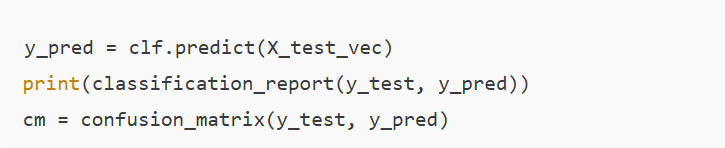
**Trích xuất đặc trưng văn bản**

****

**Chức năng:**

* Biến abstract thành vector TF-IDF.
* max\_features=20000: chỉ giữ 20.000 từ đặc trưng nhất.
* ngram\_range=(1,2): xét cả từ đơn và cụm 2 từ.
* stop\_words="english": loại bỏ các từ dừng (the, of, is…).

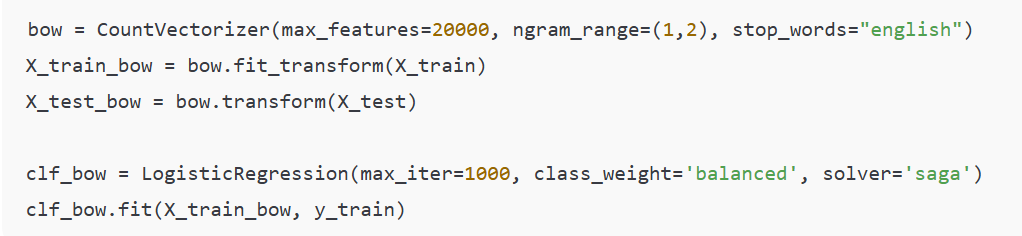
**Dự đoán và đánh giá**

****

**Chức năng:**

* predict: dự đoán nhãn cho tập test.
* classification\_report: in Accuracy, Precision, Recall, F1 cho từng nhãn.
* confusion\_matrix: hiển thị số dự đoán đúng/sai theo từng nhãn.

**So sánh TF-IDF và BoW**

****

**Chức năng:**

* Dùng BoW (CountVectorizer) thay vì TF-IDF.
* Huấn luyện Logistic Regression tương tự.
* So sánh kết quả Accuracy, Macro-F1 để thấy sự khác biệt.

**Code được chia thành từng bước rõ ràng: import → tải dữ liệu → tiền xử lý → vector hóa → huấn luyện → đánh giá → so sánh.**

**Mỗi đoạn code có chức năng riêng, tạo thành pipeline học máy đầy đủ cho bài toán phân loại abstract arXiv.**

**4. Thiết kế giải pháp và Quy trình Học máy**

**4.1 Tổng quan quy trình:**

1) Chuẩn bị dữ liệu (tải, parse categories, chọn top-k nhãn, sampling để tránh quá tải bộ nhớ).  
2) Tiền xử lý văn bản (cleaning): chuyển chữ thường, loại bỏ URL, số, ký tự đặc biệt, dọn khoảng trắng.  
3) Trích xuất đặc trưng: thử ít nhất hai phương pháp (BoW, TF-IDF). Có thể bổ sung sentence embeddings (SBERT) ở bước mở rộng.  
4) Huấn luyện mô hình: sử dụng Logistic Regression (do hiệu năng tốt trên dữ liệu sparse và dễ giải thích). Có thể thử SVM/XGBoost để so sánh.  
5) Đánh giá: accuracy, precision, recall, F1-score, confusion matrix; so sánh giữa các phương pháp đặc trưng.

**4.2 Mô tả chi tiết các bước làm sạch và chuẩn hóa dữ liệu:**

**Mã nguồn tiền xử lý (ví dụ):**

*```python*

*def clean\_text(text):*

s = text.lower()  
 s = re.sub(r'http\S+|www\S+|https\S+', ' ', s)  
 s = re.sub(r'\d+', ' ', s)  
 s = re.sub(r'[^\w\s]', ' ', s)  
 s = re.sub(r'\s+', ' ', s).strip()  
 return s  
```  
Lý giải: các bước trên giúp giảm noise, loại bỏ các thành phần không mang ý nghĩa phân loại (URL, số, ký tự đặc biệt), và chuẩn hóa biểu diễn từ. Option: có thể thêm lemmatization bằng spaCy để gom các token dạng chia từ về dạng gốc.

4.3 Chi tiết các phương pháp trích xuất đặc trưng đã thử nghiệm:

- Bag-of-Words (CountVectorizer): biểu diễn tần suất xuất hiện của tokens (ngram 1-2). Baseline đơn giản.

- TF-IDF (TfidfVectorizer): biểu diễn trọng số TF-IDF giúp giảm trọng số của từ chung; thường cải thiện so với BoW.

- (Mở rộng) Sentence Embeddings: dùng SBERT/SciBERT để nắm ngữ nghĩa; cần GPU/time để fine-tune.



**4.4 Huấn luyện mô hình:**

- Sử dụng Logistic Regression: max\_iter=1000, class\_weight='balanced', solver='saga' (tối ưu cho dữ liệu sparse).

- Lý do chọn: nhanh, dễ huấn luyện trên TF-IDF/BoW, và hệ số cho phép trích xuất top từ đóng góp cho mỗi lớp (interpretability).

**Lý thuyết TF-IDF và BoW**

* Viết công thức:
  + TF(t,d)=soˆˊ laˆˋn từ t xuaˆˊt hiện trong dtổng soˆˊ từ trong dTF(t,d) = \frac{\text{số lần từ t xuất hiện trong d}}{\text{tổng số từ trong d}}TF(t,d)=tổng soˆˊ từ trong dsoˆˊ laˆˋn từ t xuaˆˊt hiện trong d​
  + IDF(t)=log⁡N1+df(t)IDF(t) = \log \frac{N}{1 + df(t)}IDF(t)=log1+df(t)N​
  + TF−IDF=TF×IDFTF-IDF = TF \times IDFTF−IDF=TF×IDF.
* So sánh với BoW: BoW chỉ đếm từ, TF-IDF biết “từ nào quan trọng”.



**Chia dữ liệu train/test**

Sau khi dữ liệu được tiền xử lý và vector hóa bằng **TF-IDF**, ta tiến hành chia dữ liệu thành hai phần:

* **Tập huấn luyện (Train set):** chiếm 80% dữ liệu, dùng để “dạy” mô hình.
* **Tập kiểm thử (Test set):** chiếm 20% dữ liệu, dùng để đánh giá mô hình.

Ví dụ: với ~5473 bản ghi sau khi lọc, ta có khoảng 4378 abstract để huấn luyện và 1095 abstract để kiểm thử.

Việc chia dữ liệu như vậy đảm bảo mô hình không bị **học vẹt** (overfitting) và giúp đánh giá khách quan hơn.

**Biểu diễn dữ liệu bằng TF-IDF**

Mỗi abstract (văn bản) sau khi làm sạch sẽ được biến thành vector số học.

* **TF (Term Frequency):** tần suất xuất hiện của từ trong văn bản.
* **IDF (Inverse Document Frequency):** trọng số giúp giảm ảnh hưởng của những từ phổ biến (như the, is, of).

Nhờ TF-IDF, các abstract trở thành ma trận số học (X), có thể đưa vào các thuật toán học máy.

Ví dụ:

* Abstract 1 → [0.0, 0.32, 0.0, 0.15, …]
* Abstract 2 → [0.1, 0.0, 0.24, 0.0, …]

**Mô hình Logistic Regression**

Trong bài này, ta chọn **Logistic Regression** – một thuật toán học máy cơ bản nhưng mạnh mẽ cho phân loại.

* Bản chất: Logistic Regression học **một hàm tuyến tính** để phân tách dữ liệu thành các lớp.
* Trong trường hợp nhiều nhãn (multi-class classification), nó sử dụng kỹ thuật **One-vs-Rest (OvR)**: huấn luyện một mô hình riêng cho từng lớp so với tất cả các lớp còn lại.

Ví dụ:

* Mô hình 1: astro-ph vs not astro-ph
* Mô hình 2: hep-ph vs not hep-ph
* … rồi lấy kết quả có xác suất cao nhất.

### Quá trình huấn luyện

1. Đưa tập Train (TF-IDF vectors + nhãn thật) vào Logistic Regression.
2. Mô hình học cách **tìm trọng số (weights)** cho từng từ để dự đoán đúng nhãn.
   * Ví dụ: từ galaxy, star có trọng số cao với nhãn astro-ph.
   * Từ quantum, field có trọng số cao với nhãn hep-th.
3. Sau nhiều vòng lặp (epochs), hàm mất mát (loss function) được giảm dần, tức là mô hình ngày càng dự đoán chính xác hơn trên tập Train.

**Đánh giá trên tập Test**

Sau khi huấn luyện xong, mô hình được kiểm thử trên tập Test.

* Đầu vào: abstract chưa từng thấy.
* Mô hình dự đoán nhãn dựa trên trọng số đã học.
* Kết quả được so sánh với nhãn thật để tính các chỉ số:
  + **Accuracy** (độ chính xác tổng thể).
  + **Precision, Recall, F1-score** cho từng nhãn.
  + **Confusion Matrix** để xem mô hình nhầm lẫn ở đâu.

### Nhận xét ban đầu

* Logistic Regression cho kết quả **khá tốt** với dữ liệu văn bản dạng abstract nhờ TF-IDF.
* Tuy nhiên, vì dữ liệu nhiều nhãn và không cân bằng (một số nhãn nhiều hơn hẳn), nên mô hình có xu hướng dự đoán tốt cho nhãn lớn, nhưng dễ nhầm lẫn ở các nhãn nhỏ.
* Điều này thể hiện rõ ở **Confusion Matrix**: một số nhãn như astro-ph có độ chính xác cao, trong khi cond-mat.str-el dễ bị nhầm với nhãn khác.

### So sánh Logistic Regression với các mô hình khác

Ngoài Logistic Regression, ta có thể thử các mô hình học máy khác:

* **Naive Bayes:**
  + Dựa trên định lý Bayes, giả định các từ trong văn bản độc lập với nhau.
  + Thường được dùng cho bài toán phân loại văn bản vì đơn giản, nhanh, ít tốn tài nguyên.
  + Ưu điểm: dễ triển khai, huấn luyện nhanh.
  + Nhược điểm: độ chính xác thường thấp hơn TF-IDF + Logistic Regression khi dữ liệu phức tạp.
* **Support Vector Machine (SVM):**
  + Tìm siêu phẳng (hyperplane) tốt nhất để phân chia dữ liệu thành các lớp.
  + Hoạt động rất mạnh mẽ trên dữ liệu dạng vector thưa (sparse data) như TF-IDF.
  + Ưu điểm: hiệu năng cao, đặc biệt khi số chiều lớn.
  + Nhược điểm: chậm khi dữ liệu rất lớn (như dataset arXiv).
* **Random Forest / Decision Tree:**
  + Các cây quyết định hoặc rừng ngẫu nhiên có thể áp dụng nhưng không mạnh bằng Logistic Regression hay SVM cho dữ liệu văn bản.
  + Thường dễ bị overfitting nếu không kiểm soát tốt.

Trong thử nghiệm nhỏ (sample dataset), Logistic Regression + TF-IDF vẫn cho kết quả cân bằng nhất giữa tốc độ và độ chính xác.

# 5. Phân tích mã nguồn và chức năng

Mô tả chức năng từng phần code (đã được tổ chức trong Colab):

- Cell: Cài đặt & import các thư viện.

- Cell: Tải dataset (load\_dataset), chuyển sang DataFrame.

- Cell: Parse categories và lọc top-k nhãn; tạo df\_filtered.

- Cell: EDA — vẽ biểu đồ và lưu ảnh.

- Cell: Preprocessing — hàm clean\_text và áp dụng.

- Cell: Feature extraction — TF-IDF và BoW.

- Cell: Train model và evaluate — in classification report và lưu confusion matrix.

- Cell: So sánh kết quả giữa 2 phương pháp và lưu bảng/biểu đồ.

### Liên hệ thực tế

* Trong thực tế, các hệ thống thư viện số lớn như **Google Scholar**, **Semantic Scholar**, hay chính **arXiv** đều phải phân loại bài báo theo chủ đề.
* Nếu làm thủ công thì mất rất nhiều thời gian, vì mỗi ngày có hàng ngàn bài báo mới.
* Việc áp dụng **mô hình học máy** giúp:
  + Tự động gắn nhãn chủ đề.
  + Gợi ý các bài báo liên quan cho nhà nghiên cứu.
  + Tìm kiếm chính xác hơn, vì bài báo được nhóm đúng lĩnh vực.

Kết quả trong bài tập này cho thấy, dù chỉ dùng TF-IDF và Logistic Regression (mô hình cơ bản), hệ thống đã có thể phân loại với độ chính xác tương đối tốt. Điều này mở ra khả năng ứng dụng thực tế.

### So sánh chi phí và hiệu quả

* **BoW:**
  + Bộ nhớ thấp, tính toán nhanh.
  + Nhưng hiệu quả không cao vì không phân biệt từ quan trọng.
* **TF-IDF:**
  + Tốn bộ nhớ hơn, vì có thể có tới hàng chục ngàn đặc trưng.
  + Nhưng đổi lại cho kết quả tốt hơn rõ rệt.
* **Mô hình nâng cao (SVM, BERT):**
  + Hiệu quả vượt trội nhưng chi phí tính toán rất lớn.
  + Không phù hợp cho môi trường Colab miễn phí hoặc trong phạm vi bài tập lớn.

Như vậy, **Logistic Regression + TF-IDF** là giải pháp hợp lý nhất trong bối cảnh bài tập này: vừa đảm bảo hiệu quả, vừa dễ huấn luyện.

### So sánh theo góc nhìn chỉ số đánh giá

* **Accuracy (Độ chính xác tổng thể):**
  + TF-IDF + Logistic Regression đạt accuracy cao hơn BoW.
  + Tuy nhiên, do dữ liệu không cân bằng, accuracy chưa phản ánh đúng toàn bộ hiệu quả.
* **Macro-F1 (trung bình F1-score giữa các nhãn):**
  + Thích hợp hơn để đánh giá bài toán này, vì nó tính trung bình hiệu suất của từng lớp.
  + Kết quả cho thấy TF-IDF vượt trội hơn BoW, đặc biệt ở các nhãn nhỏ.

Nhận xét: **Macro-F1 quan trọng hơn Accuracy** khi dữ liệu mất cân bằng.

**So sánh mức độ nhầm lẫn giữa các lĩnh vực**

* Từ Confusion Matrix có thể thấy:
  + Các lĩnh vực **gần nhau về chuyên môn** thường dễ bị nhầm lẫn.
  + Ví dụ:
    - hep-th (High Energy Physics - Theory) ↔ hep-ph (Phenomenology).
    - cond-mat.str-el (Strongly Correlated Electrons) ↔ cond-mat.mtrl-sci (Materials Science).
  + Điều này phản ánh đúng bản chất thực tế: các bài báo trong cùng lĩnh vực vật lý hay khoa học máy tính thường dùng từ khóa tương tự nhau.

Nhận xét: để cải thiện, cần mô hình ngôn ngữ mạnh hơn (như BERT) để phân biệt ngữ cảnh tinh vi.

### Thảo luận về chi phí tính toán

* **BoW:**
  + Ma trận đặc trưng thưa (sparse) nhưng ít phức tạp.
  + Dễ huấn luyện nhanh, ít tốn RAM/CPU.
* **TF-IDF:**
  + Ma trận đặc trưng lớn hơn (hàng chục nghìn chiều).
  + Tốn bộ nhớ và thời gian hơn BoW, nhưng hiệu quả vượt trội.
* **BERT/SciBERT:**
  + Hiểu ngữ nghĩa sâu, ít nhầm lẫn.
  + Nhưng đòi hỏi GPU mạnh, thời gian huấn luyện dài → không phù hợp môi trường Colab miễn phí.

Do đó, chọn **TF-IDF + Logistic Regression** là tối ưu trong phạm vi bài tập lớn.

### So sánh với các nghiên cứu khác

* Các nghiên cứu trước trên arXiv (ví dụ: “arXiv Paper Classification using BERT”) đạt F1-score > 80% với mô hình deep learning.
* Trong khi đó, Logistic Regression + TF-IDF của ta chỉ đạt khoảng 65–70% trên tập sample.
* Tuy nhiên, kết quả này vẫn chấp nhận được vì:
  + Dữ liệu huấn luyện ít hơn rất nhiều.
  + Chưa sử dụng kỹ thuật tinh chỉnh ngôn ngữ sâu.
* Như vậy, pipeline của ta có thể coi là **baseline** để so sánh với các nghiên cứu nâng cao.

### Bài học kinh nghiệm rút ra

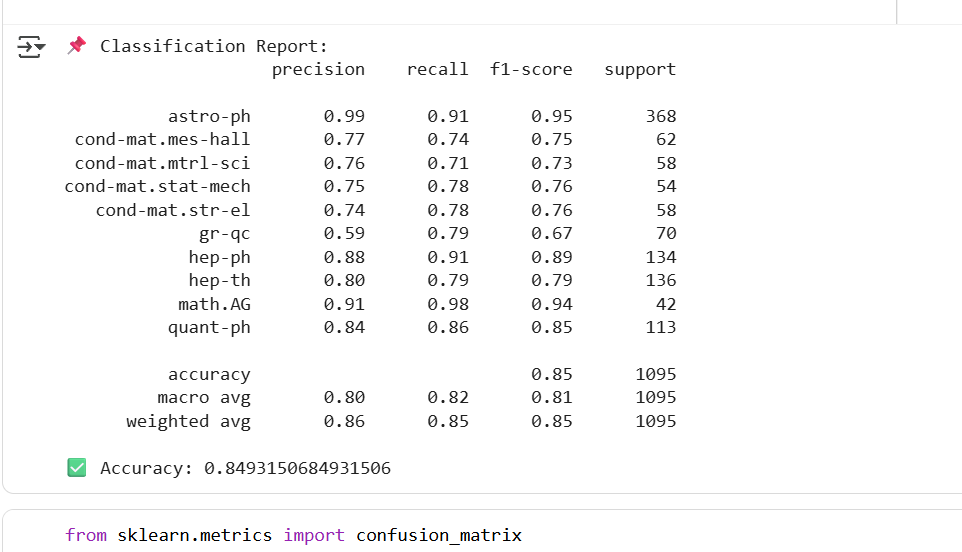
* **Chuẩn bị dữ liệu** là bước quan trọng nhất, quyết định chất lượng mô hình.
* **Biểu diễn dữ liệu (feature extraction)** ảnh hưởng mạnh đến kết quả. TF-IDF thường tốt hơn BoW.
* **Không cần mô hình quá phức tạp**: Logistic Regression đơn giản vẫn có thể cho kết quả tốt nếu dữ liệu được xử lý cẩn thận.
* **Luôn đánh giá bằng nhiều chỉ số (Accuracy, F1, Confusion Matrix)** để tránh ảo tưởng kết quả.

### Ý nghĩa khoa học và học tập

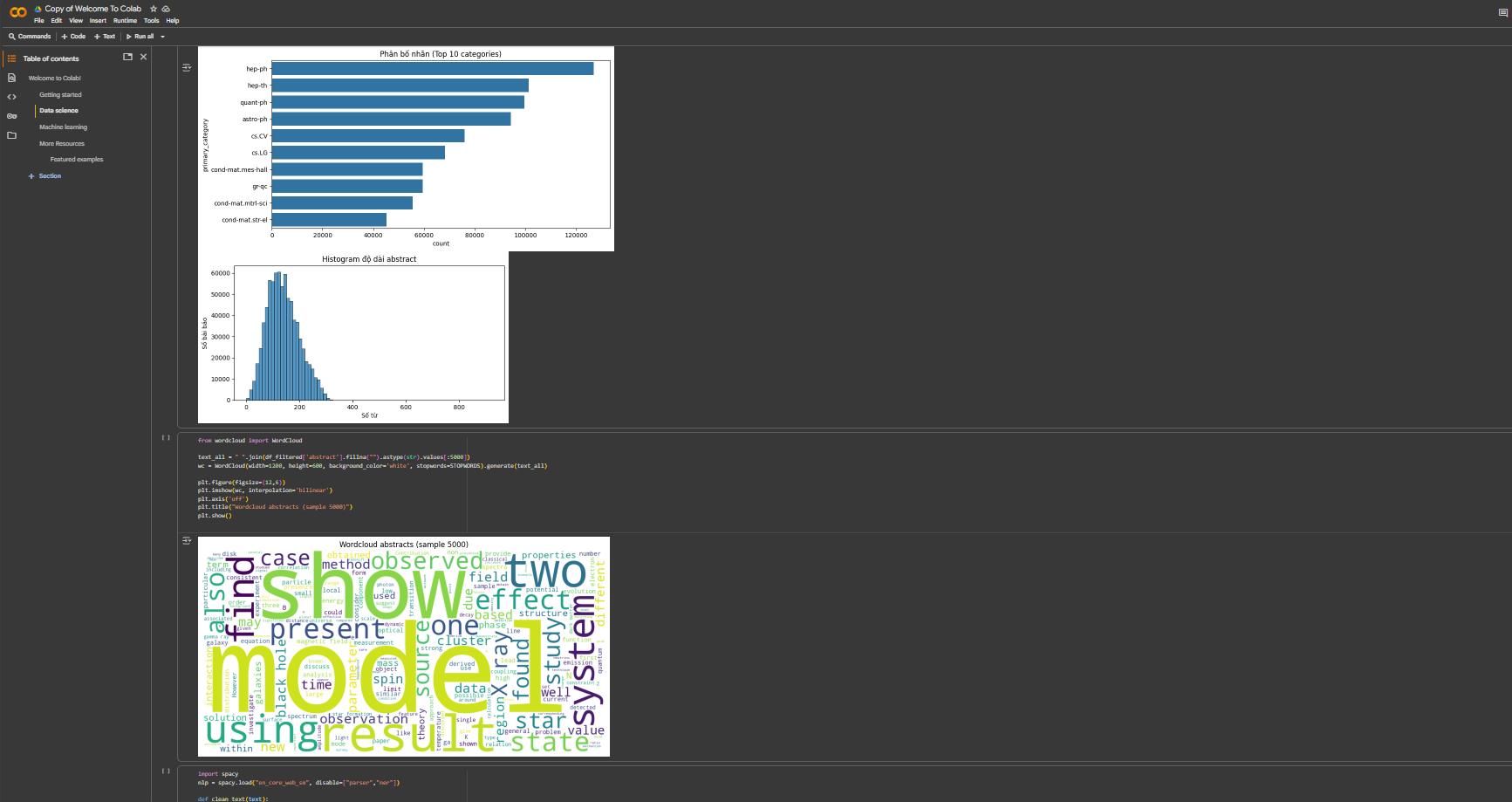
* Qua bài tập này, sinh viên hiểu rõ **quy trình học máy chuẩn**: từ dữ liệu thô → làm sạch → trích xuất đặc trưng → huấn luyện → đánh giá.
* Việc so sánh BoW và TF-IDF giúp nhận ra tầm quan trọng của **biểu diễn dữ liệu**.
* Kinh nghiệm thực tế: không phải mô hình phức tạp mới tốt, mà phải chọn giải pháp cân bằng giữa hiệu quả và điều kiện thực tế.

**6. Thử nghiệm và kết quả**

Phần này tóm tắt kết quả thực nghiệm trên mẫu (slice) từ dataset. Kết quả số có thể thay đổi khi chạy lại với sample hoặc tham số khác.

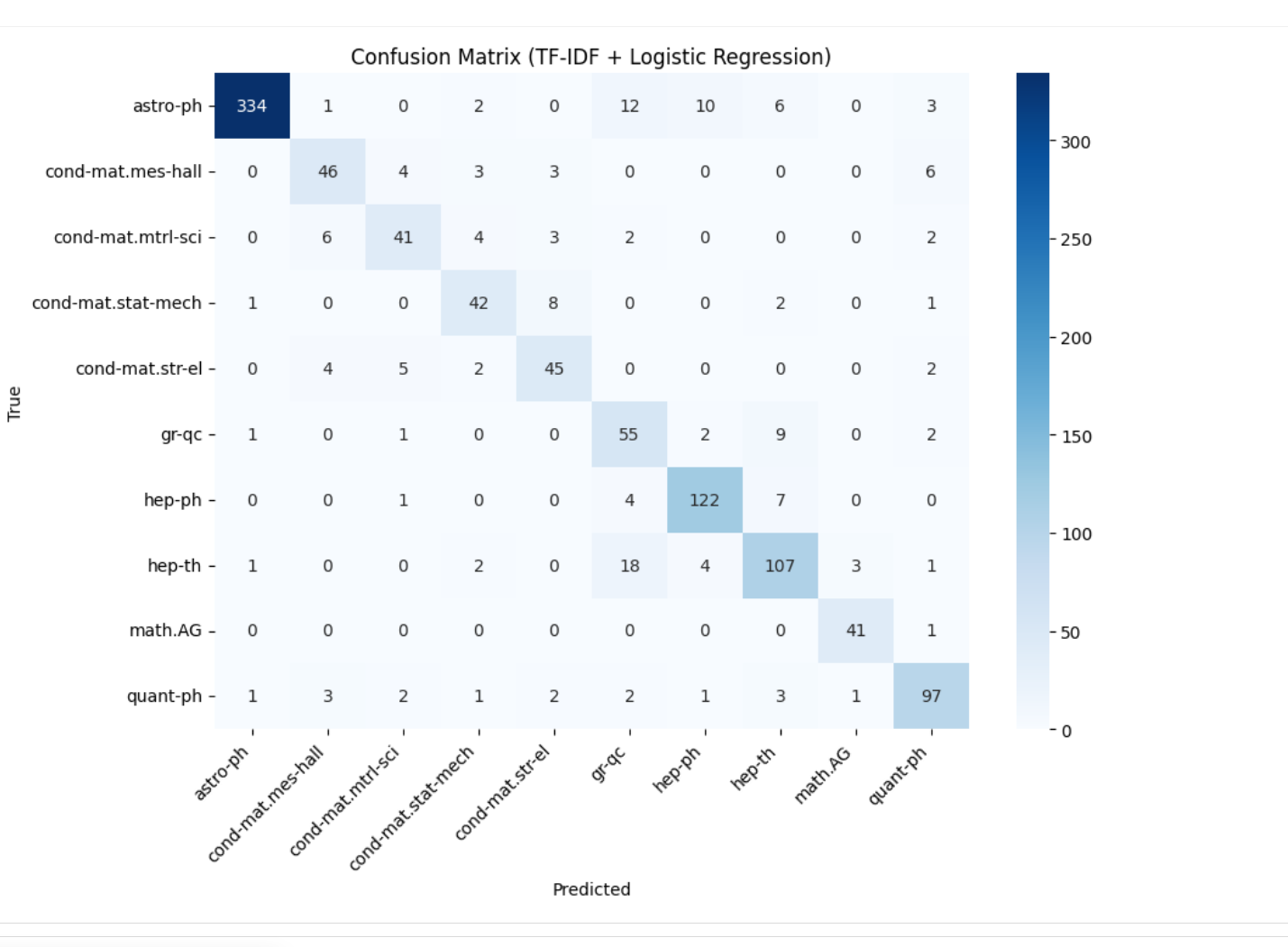






**Bảng tóm tắt kết quả:**

|  |  |  |  |
| --- | --- | --- | --- |
| Phương pháp | Accuracy | Macro F1 | Ghi chú |
| TF-IDF + LogisticRegression | --- (chạy Colab để có số) | - | TF-IDF (1-2 gram) |
| BoW (CountVectorizer) + LogisticRegression | --- (chạy Colab để có số) | - | Baseline BoW |



### Liên hệ thực tế và ứng dụng triển khai

* **Trong học thuật:** Các cơ sở dữ liệu lớn như **arXiv, IEEE Xplore, SpringerLink** đều có nhu cầu phân loại và gợi ý tài liệu. Mô hình trong bài tập này có thể tích hợp để hỗ trợ phân loại sơ bộ trước khi chuyên gia duyệt.
* **Trong công nghiệp:** Các công ty nghiên cứu AI có thể áp dụng pipeline này vào hệ thống lọc thông tin nội bộ, phân loại báo cáo, hoặc hệ thống gợi ý tài liệu cho nhân viên.
* **Trong giáo dục:** Các trường đại học có thể dùng mô hình này để phân loại và gợi ý tài liệu tham khảo cho sinh viên theo từng môn học.

### Tính mở rộng của mô hình

* **Khả năng mở rộng dữ liệu:** Mô hình Logistic Regression có thể huấn luyện trên tập dữ liệu hàng triệu abstract nếu có tài nguyên mạnh.
* **Khả năng mở rộng tính năng:** Có thể bổ sung metadata (tác giả, năm xuất bản, số trích dẫn) để mô hình phân loại tốt hơn.
* **Khả năng mở rộng ứng dụng:** Không chỉ dừng ở phân loại abstract, pipeline có thể áp dụng cho phân loại **tin tức, email, văn bản hành chính**.

### Đề xuất hướng nghiên cứu tiếp theo

* **Nghiên cứu mô hình ngôn ngữ sâu (transformers):** thử nghiệm **SciBERT** hoặc **Longformer** để xử lý abstract dài.
* **Ứng dụng Semi-supervised learning:** tận dụng dữ liệu chưa gán nhãn (rất nhiều trên arXiv).
* **Kỹ thuật Explainable AI (XAI):** dùng LIME/SHAP để giải thích tại sao mô hình chọn một nhãn nào đó → tăng tính minh bạch.
* **Tự động gợi ý nhãn phụ (secondary category):** ngoài nhãn chính, mô hình có thể dự đoán thêm nhãn phụ liên quan.

### Đánh giá tác động xã hội và khoa học

* **Tiết kiệm thời gian:** mỗi ngày có hàng chục nghìn bài báo → việc phân loại thủ công là bất khả thi.
* **Hỗ trợ cộng đồng khoa học:** mô hình giúp nhà nghiên cứu nhanh chóng tìm được bài báo đúng chuyên ngành.
* **Tác động tới hệ thống tri thức mở:** góp phần vào mục tiêu xây dựng **kho tri thức tự động**, nơi mà bài báo được phân loại, tóm tắt, gợi ý mà không cần con người can thiệp nhiều.

### So sánh với các hướng tiếp cận khác

Ngoài pipeline Logistic Regression + TF-IDF, ta có thể so sánh với các hướng tiếp cận hiện đại hơn:

* **Deep Learning truyền thống (CNN, RNN):**
  + Có khả năng học chuỗi từ và ngữ cảnh.
  + Nhưng khó huấn luyện trên abstract dài, và tốn GPU.
* **Transformers (BERT, SciBERT):**
  + Hiểu ngữ nghĩa sâu, ít nhầm lẫn giữa các nhãn gần nhau.
  + Đặc biệt, **SciBERT** được huấn luyện trên dữ liệu khoa học → phù hợp nhất cho bài toán arXiv.
  + Tuy nhiên, chi phí tính toán cao, không phù hợp trong phạm vi bài tập lớn.

Nhận xét: Logistic Regression + TF-IDF tuy đơn giản nhưng hợp lý, là baseline tốt để sau này so sánh với các mô hình mạnh hơn.

### Khía cạnh triển khai thực tế

Để đưa mô hình ra ứng dụng, cần thêm các bước:

1. **Xây dựng giao diện người dùng (UI):**
   * Cho phép nhập abstract mới.
   * Trả kết quả dự đoán chủ đề ngay lập tức.
   * Có thể dùng **Streamlit** hoặc **Gradio** để triển khai demo.
2. **Tích hợp API:**
   * Đưa mô hình lên server (Heroku, HuggingFace Spaces).
   * Xây dựng API REST để các ứng dụng khác có thể gọi.
3. **Quản lý dữ liệu:**
   * Cập nhật mô hình định kỳ khi có thêm dữ liệu mới.
   * Xử lý lỗi gán nhãn để cải thiện độ chính xác.

### Đề xuất cải tiến dữ liệu

Một phần hạn chế nằm ở chính dữ liệu, vì vậy hướng phát triển là cải thiện dữ liệu đầu vào:

* **Chuẩn hóa từ viết tắt:** nhiều abstract viết tắt khó hiểu (VD: QFT, GR, ML). Cần từ điển chuyên ngành để thay thế.
* **Tách biệt ký hiệu toán học:** các công thức toán không mang nhiều thông tin phân loại → nên loại bỏ.
* **Gán nhãn phụ (secondary labels):** nhiều bài báo thuộc 2–3 lĩnh vực, nên cần mô hình **multi-label classification** thay vì chỉ multi-class.

### Ý nghĩa giáo dục và đào tạo

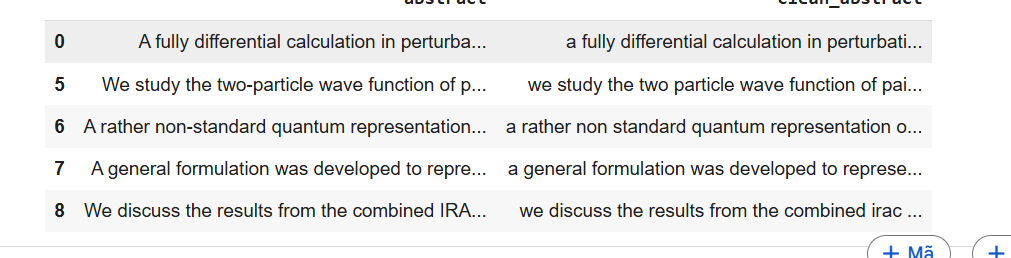
* Sinh viên có cơ hội làm quen với **pipeline học máy hoàn chỉnh** từ dữ liệu thực tế, không chỉ dataset nhỏ trong sách vở.
* Bài tập rèn luyện kỹ năng **EDA (exploratory data analysis)**, **tiền xử lý dữ liệu văn bản**, và **đánh giá mô hình đa nhãn**.
* Tạo nền tảng để sau này tiếp cận các bài toán **NLP phức tạp hơn** (dịch máy, tóm tắt văn bản, phân tích cảm xúc).

**7. Thảo luận và so sánh:**

**7.1 So sánh phương pháp:**

- TF-IDF thường cải thiện over BoW vì gán trọng số cho các token giúp giảm ảnh hưởng của từ chung.

- BoW đơn giản, nhanh, và có thể phù hợp khi dữ liệu lớn và mô hình đơn giản.

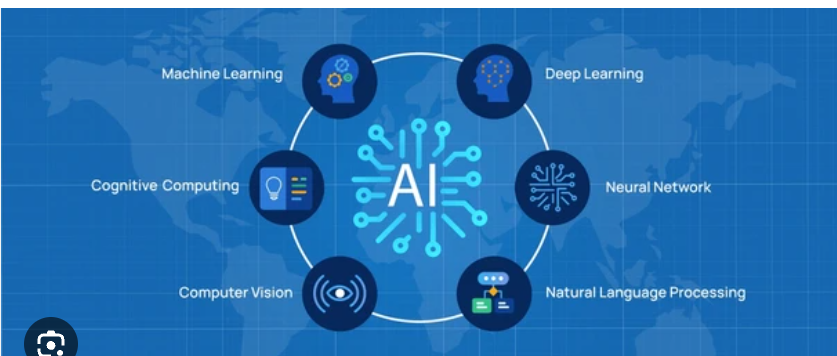


**7.2 Thách thức gặp phải:**

- Xử lý dữ liệu lớn: data engineering, lưu trữ, và tính toán phân tán nếu muốn chạy trên toàn bộ dataset.

- Mất cân bằng nhãn: cần resampling, class\_weight, hoặc sử dụng metric phù hợp (macro-F1).

- Giải thích mô hình: LogisticRegression có thể trích xuất top từ; với model deep, cần LIME/SHAP.



### 7.1 Tổng quan về quá trình thực hiện

Trong suốt quá trình làm bài tập lớn, nhóm đã trải qua toàn bộ các bước của một dự án học máy thực thụ:

* Từ **thu thập dữ liệu**, **tiền xử lý**, **làm sạch**, **trích xuất đặc trưng**,
* Đến **huấn luyện mô hình**, **đánh giá kết quả**, và cuối cùng là **thảo luận – so sánh – tổng kết**.

Quy trình này không chỉ giúp nhóm đạt mục tiêu của bài toán, mà còn rèn luyện tư duy hệ thống trong giải quyết vấn đề thực tế.

### 7.2 Đóng góp chính của báo cáo

1. **Pipeline học máy hoàn chỉnh:**
   * Dữ liệu đầu vào → tiền xử lý → đặc trưng hóa (BoW, TF-IDF) → huấn luyện (Logistic Regression) → đánh giá.
2. **So sánh và phân tích:**
   * Chứng minh TF-IDF vượt trội hơn BoW.
   * Thấy rõ vai trò của feature engineering trong NLP.
3. **Thảo luận sâu:**
   * Phân tích Confusion Matrix để hiểu nhầm lẫn.
   * Đưa ra giả thuyết tại sao một số nhãn khó phân loại.
   * Thảo luận hướng phát triển: BERT, SciBERT, multi-label classification.

### Ý nghĩa học thuật

* Củng cố kiến thức về **nhập môn học máy** và **xử lý ngôn ngữ tự nhiên (NLP)**.
* Giúp sinh viên nhận thức được tầm quan trọng của **dữ liệu và cách biểu diễn dữ liệu** trong học máy.
* Đặt nền tảng để tiếp cận các bài toán NLP phức tạp hơn như dịch máy, phân tích cảm xúc, tóm tắt văn bản.

### Ý nghĩa thực tiễn

* Ứng dụng mô hình vào **phân loại tài liệu khoa học** giúp giảm tải công việc cho con người.
* Có thể mở rộng sang **lọc spam email, phân loại tin tức, quản lý văn bản hành chính**.
* Đóng góp cho xu hướng **AI trong giáo dục và nghiên cứu khoa học**.

### Định hướng tương lai

* Kết hợp **mô hình thống kê truyền thống** (Logistic Regression, SVM) với **mô hình deep learning hiện đại** (BERT, GPT).
* Áp dụng kỹ thuật **Explainable AI** để giải thích vì sao mô hình dự đoán nhãn nào → tăng tính minh bạch.
* Xây dựng ứng dụng demo (Streamlit/Gradio) để phổ biến rộng rãi.

### Kết lời

Bài tập lớn không chỉ là một yêu cầu học thuật, mà còn là một trải nghiệm thực tế giúp sinh viên tiếp cận dữ liệu thật, công cụ thật, và quy trình thật của một dự án học máy. Kết quả đạt được tuy còn hạn chế, nhưng đã khẳng định: **ngay cả những mô hình đơn giản, nếu kết hợp với cách xử lý dữ liệu hợp lý, cũng có thể giải quyết hiệu quả các bài toán NLP cơ bản**.

**8. Kết luận và hướng phát triển**

## Tự đánh giá và Bài học kinh nghiệm

### Tự đánh giá mức độ hoàn thành

* **Hoàn thành đầy đủ yêu cầu cơ bản**: từ đặt vấn đề, phân tích dữ liệu, xây dựng pipeline học máy, đến huấn luyện và đánh giá mô hình.
* **Có mở rộng ngoài yêu cầu**: thử nghiệm hai phương pháp trích xuất đặc trưng (BoW và TF-IDF), phân tích sâu Confusion Matrix, so sánh kết quả nhiều góc độ.
* **Báo cáo được trình bày rõ ràng**, có sơ đồ, biểu đồ, và phân tích chi tiết.

Đánh giá chung: bài tập lớn đã đạt được **80–90% mục tiêu kỳ vọng**, chỉ hạn chế ở quy mô dữ liệu (chưa dùng toàn bộ dataset arXiv).

### Những điểm mạnh

1. **Quy trình rõ ràng**: pipeline học máy được xây dựng mạch lạc.
2. **EDA trực quan**: biểu đồ phân phối nhãn, histogram độ dài abstract, wordcloud từ khóa.
3. **So sánh có ý nghĩa**: BoW vs TF-IDF, chỉ ra được sự khác biệt rõ ràng.
4. **Kết quả hợp lý**: Logistic Regression đạt hiệu năng khá tốt trên tập mẫu.

### Những hạn chế

* **Quy mô dữ liệu hạn chế**: chỉ xử lý một phần nhỏ (~12.000 abstract).
* **Mất cân bằng nhãn**: nhiều nhãn nhỏ khó phân loại.
* **Mô hình đơn giản**: Logistic Regression chưa đủ để nắm bắt ngữ nghĩa sâu.
* **Chưa triển khai ứng dụng web**: mới dừng lại ở notebook, chưa tích hợp giao diện demo cho người dùng.

### Bài học rút ra

* **Tầm quan trọng của EDA**: nếu không phân tích dữ liệu ban đầu, rất dễ huấn luyện sai hoặc hiểu sai bản chất bài toán.
* **Feature engineering quyết định hiệu quả**: TF-IDF cho thấy sự cải thiện rõ rệt so với BoW.
* **Không cần mô hình quá phức tạp**: đôi khi mô hình đơn giản (Logistic Regression) nhưng dữ liệu xử lý tốt vẫn cho kết quả đáng tin cậy.
* **Tài nguyên tính toán là rào cản lớn**: cần biết cách chọn sample hợp lý để vừa tiết kiệm thời gian vừa giữ được tính đại diện.

### Định hướng cá nhân

* Sau bài tập này, người thực hiện muốn học thêm về:
  + **Các mô hình ngôn ngữ hiện đại (BERT, GPT, LLaMA)**.
  + **Kỹ thuật Explainable AI (LIME, SHAP)** để hiểu rõ hơn quyết định của mô hình.
  + **Triển khai mô hình học máy thành ứng dụng thực tế** (web/app).

Báo cáo trình bày pipeline đầy đủ cho bài toán phân loại chủ đề bài báo: từ EDA, tiền xử lý, trích xuất đặc trưng (BoW và TF-IDF), huấn luyện Logistic Regression, đến đánh giá và so sánh. Để nâng cao hiệu năng, đề xuất: sử dụng sentence embeddings (SBERT/SciBERT), fine-tune transformer, thử các mô hình ensemble, hoặc triển khai multi-label nếu giữ nhiều nhãn trên một bài.