TRƯỜNG ĐẠI HỌC VĂN LANG

**KHOA CÔNG NGHỆ THÔNG TIN**

A red and black shield with white logo

Description automatically generated

NGÀNH: CÔNG NGHỆ THÔNG TIN

***Môn học:***

**Nhập Môn Phân Tích Dữ Liệu Và Học Sâu**

**Chủ đề : Nhận Diện Fake News**

**SVTH-MSSV thực hiện :**

**Nguyễn Hải Quỳnh- 207CT28431**

**Lê Đức Duy – 2174802010232**

**Đỗ Minh Thông- 207ct65831**

**Trần Lương Hòa – 2174802010453**

**GVHD: Nguyễn Thái Anh**

TP. Hồ Chí Minh – năm 2024

|  |
| --- |
| **Lời Mở Đầu**  Trong thời đại kỹ thuật số hiện nay, thông tin được lan truyền với tốc độ chưa từng có, nhờ vào sự phát triển mạnh mẽ của các nền tảng mạng xã hội và internet. Mặc dù sự bùng nổ thông tin mang lại nhiều lợi ích đáng kể cho việc trao đổi kiến thức, giao lưu văn hóa và kết nối xã hội, nhưng nó cũng đặt ra một thách thức lớn: sự xuất hiện và lan truyền của tin giả. Tin giả không chỉ gây hoang mang và nhầm lẫn cho người đọc, mà còn có thể dẫn đến những hậu quả nghiêm trọng, ảnh hưởng đến cá nhân, cộng đồng và toàn xã hội.  Tin giả, hay còn gọi là fake news, là những thông tin sai lệch, không có căn cứ, được tạo ra với mục đích gây rối hoặc lừa dối người đọc. Trong bối cảnh này, việc nhận dạng và phòng chống tin giả trở thành một nhiệm vụ cấp thiết và quan trọng, nhằm bảo vệ sự thật và duy trì sự tin cậy trong thông tin.    **Giới Thiệu**  Báo cáo này tập trung vào việc nghiên cứu và phát triển các phương pháp nhận dạng tin giả, với mục tiêu xây dựng một hệ thống hiệu quả để phát hiện và loại bỏ những thông tin sai lệch. Nội dung báo cáo được tổ chức thành các phần sau:   1. **Khái niệm và tác động của tin giả:** Nhóm chúng em sẽ cung cấp một cái nhìn tổng quan về khái niệm tin giả, phân loại các loại hình tin giả phổ biến, và phân tích những tác động tiêu cực của tin giả đối với cá nhân, tổ chức và xã hội. Điều này bao gồm việc xem xét các trường hợp thực tế và các nghiên cứu điển hình để minh họa mức độ nghiêm trọng của vấn đề. 2. **Phương pháp nhận dạng tin giả:** Trong phần này, nhóm chúng em sẽ trình bày các kỹ thuật và công cụ hiện có để nhận dạng tin giả. Điều này bao gồm cả các phương pháp truyền thống như kiểm chứng thông tin qua các nguồn đáng tin cậy và các giải pháp hiện đại dựa trên trí tuệ nhân tạo (AI) và học máy (machine learning). Chúng em sẽ giới thiệu các thuật toán, mô hình và hệ thống đã được phát triển và áp dụng thành công trong việc phát hiện tin giả. 3. **Thách thức và hướng đi tương lai:** Nhận dạng tin giả không phải là một nhiệm vụ dễ dàng. Chúng em sẽ đánh giá những khó khăn và thách thức hiện tại trong việc phát hiện tin giả, chẳng hạn như sự phát triển của các kỹ thuật làm giả tinh vi và sự đa dạng của các nguồn tin. Ngoài ra, chúng em cũng sẽ đề xuất các hướng nghiên cứu tiềm năng và các chiến lược cải tiến nhằm nâng cao hiệu quả của hệ thống nhận dạng tin giả trong tương lai.   Thông qua việc phân tích sâu sắc và áp dụng các phương pháp tiên tiến, chúng em hy vọng báo cáo này sẽ góp phần nâng cao khả năng nhận diện và loại trừ tin giả. Đồng thời, chúng em cũng mong muốn thúc đẩy nhận thức cộng đồng về tầm quan trọng của việc kiểm chứng thông tin và bảo vệ sự thật, từ đó tăng cường độ tin cậy và minh bạch của thông tin trên mạng.  Vậy nên chúng em quyết định chọn chủ đề này và phát triển dự án tạo ra 1 ứng dụng phân tích dữ liệu nhận dạng đó có phải là tin giả hay là tin thật? và sau đây là cách chúng em thực hiện dự án này. Trước tiên hãy cùng nhóm em tổng quan các khái niệm cơ bản của môn học và các thứ cần thiết .  Bước 1 : Các thư viện cần phải khai báo để chạy code mượt mà : |
| |  |  | | --- | --- | | Pandas là một thư viện Python cung cấp các cấu trúc dữ liệu nhanh, mạnh mẽ, linh hoạt. Pandas được thiết kế để làm việc dễ dàng và trực quan với dữ liệu có cấu trúc (dạng bảng, đa chiều, không đồng nhất) và dữ liệu chuỗi thời gian. |  | | NumPy (Numerical Python) là một thư viện mạnh mẽ để làm việc với mảng và ma trận trong Python.  Nó cung cấp nhiều hàm và phép toán số học để thực hiện các tính toán số học nhanh chóng và hiệu quả. |  | | Seaborn là một thư viện trực quan hóa dữ liệu dựa trên matplotlib. Nó cung cấp các chủ đề màu sắc hấp dẫn và các chức năng trực quan hóa dữ liệu phổ biến để giúp bạn tạo ra các biểu đồ thể hiện dữ liệu một cách dễ dàng và hấp dẫn hơn. |  | | Matplotlib là một thư viện trực quan hóa mạnh mẽ trong Python. Nó cho phép bạn tạo ra các biểu đồ, đồ thị và hình ảnh chất lượng cao để trực quan hóa dữ liệu một cách linh hoạt. |  | |
| |  |  | | --- | --- | | scikit-learn là một thư viện phổ biến trong machine learning. Nó cung cấp nhiều công cụ để thực hiện các nhiệm vụ phân loại, hồi quy, gom cụm và huấn luyện mô hình machine learning. |  | | train\_test\_split() là một hàm trong sklearn để chia tách tập dữ liệu thành tập huấn luyện và tập kiểm tra. Bằng cách sử dụng hàm này, bạn có thể chia dữ liệu thành các tập con riêng biệt để huấn luyện và đánh giá mô hình machine learning. |  | | accuracy\_score là một hàm trong sklearn để tính toán độ chính xác của mô hình dự đoán. Nó so sánh đầu ra dự đoán của mô hình với nhãn thực tế và tính toán tỷ lệ dự đoán chính xác trên tập dữ liệu. |  | | classification\_report là một hàm trong sklearn để tạo báo cáo phân loại chi tiết cho mô hình dự đoán. Nó cung cấp thông tin về độ chính xác, precision, recall và f1-score cho mỗi lớp trong bài toán phân loại |  |  |  |  | | --- | --- | | re là một thư viện trong Python cho phép bạn làm việc với biểu thức chính quy (regular expressions). Nó cung cấp các phương thức để tìm kiếm, trích xuất và thay thế các mẫu chuỗi trong văn bản. |  | | string là một thư viện cung cấp các hằng số và phương thức hữu ích để làm việc với chuỗi trong Python. Nó bao gồm các phương thức để xử lý, kiểm tra và định dạng chuỗi. | |

Bước 2 : importing dataset

A screenshot of a computer program

Description automatically generated

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | Function (hàm): Trong lập trình, một function (hàm) là một khối mã độc lập được đặt tên, thực hiện một tác vụ cụ thể và có thể được gọi từ các phần khác của chương trình để thực hiện tác vụ đó. Hàm giúp tổ chức và tái sử dụng mã, giúp giảm độ phức tạp của chương trình | Description (mô tả): Trong lập trình, một description (mô tả) là một phần giải thích hoặc hướng dẫn liên quan đến một đối tượng, một hàm hoặc một phần của mã. | |  | pd.read\_csv: Đây là một phương thức trong thư viện pandas để đọc dữ liệu từ file CSV (Comma-Separated Values). Phương thức này cho phép bạn đọc dữ liệu từ file CSV và tạo thành một đối tượng DataFrame trong pandas. Ví dụ: | | Ví dụ : | | head(): Đây là một phương thức trong đối tượng DataFrame trong thư viện pandas. Phương thức này được sử dụng để hiển thị một số hàng đầu tiên trong DataFrame. Theo mặc định, nếu không chỉ định tham số, phương thức "head()" sẽ hiển thị 5 hàng đầu tiên của DataFrame. | | VÍ dụ |   KẾT QUẢ : |

BƯỚC 3 : ASSIGNING CLASSES TO THE DATASET ( GÁN NHÃN CHO TẬP DỮ LIỆU )

A close-up of a computer code

Description automatically generated

Trong các bài toán phân loại nhị phân (binary classification), thông thường ta cần gán một trong hai lớp cho mỗi mẫu dữ liệu. Thông thường, ta sử dụng giá trị 0 để biểu diễn lớp negative (âm) hoặc lớp không phải của sự quan tâm, và sử dụng giá trị 1 để biểu diễn lớp positive (dương) hoặc lớp của sự quan tâm.

Trong ví dụ của bạn, giả sử có hai tập dữ liệu: "data\_fake" và "data\_true". Chúng ta gán lớp 0 cho "data\_fake" và lớp 1 cho "data\_true". Điều này có thể ám chỉ rằng "data\_fake" đại diện cho các mẫu không đúng hoặc giả mạo, trong khi "data\_true" đại diện cho các mẫu chính xác hoặc thật sự. Tuy nhiên, nếu không có thông tin cụ thể về bài toán hoặc dữ liệu, không thể xác định chính xác lý do cụ thể cho việc gán lớp như trên.

Như vậy, việc gán giá trị 0 và 1 cho các lớp cụ thể trong tập dữ liệu phụ thuộc vào yêu cầu và ý nghĩa của bài toán cụ thể mà bạn đang làm việc.

BƯỚC 4: **Checking Number of Rows and Columns in the Dataset**" (Kiểm tra số hàng và số cột trong tập dữ liệu)



A close up of numbers

Description automatically generated

A screenshot of a computer

Description automatically generated

"shape" là một thuộc tính của một mảng numpy hoặc một đối tượng numpy ndarray. Thuộc tính này trả về thông tin về kích thước và hình dạng của mảng.

Các mảng numpy là cấu trúc dữ liệu đa chiều có thể chứa các phần tử cùng kiểu dữ liệu. Mỗi mảng numpy có một hình dạng (shape) đại diện cho số chiều và kích thước của mỗi chiều.

--------------------------------------------------------------------------------------------------------------------

A close up of numbers

Description automatically generated

Kết quả ((23481, 5), (21417, 5)) cho biết rằng cả "data\_fake" và "data\_true" đều có cùng số lượng hàng và số lượng cột là 5.

"data\_fake" có kích thước là (23481, 5), có nghĩa là nó chứa 23,481 mẫu và mỗi mẫu có 5 đặc trưng hoặc thuộc tính.

Tương tự, "data\_true" có kích thước là (21417, 5), có nghĩa là nó chứa 21,417 mẫu và mỗi mẫu có 5 đặc trưng hoặc thuộc tính

-------------------------------------------

=>Kết quả này cung cấp thông tin về số lượng mẫu và số lượng đặc trưng trong mỗi tập dữ liệu. Nó có thể hữu ích để hiểu kích thước của dữ liệu và xác định số lượng mẫu và đặc trưng trong quá trình xử lý và phân tích dữ liệu.

Bước 5 : Manual Testing for Both the Dataset (Kiểm tra thủ công cho cả hai tập dữ liệu)

A screenshot of a computer code

Description automatically generated

Dòng **data\_fake\_manual\_testing = data\_fake.tail(10)** tạo ra một bản sao của 10 hàng cuối cùng trong "data\_fake" và lưu vào "data\_fake\_manual\_testing". Điều này cho phép bạn giữ lại một tập dữ liệu con để thực hiện kiểm thử thủ công hoặc kiểm tra dữ liệu.

Trong vòng lặp **for i in range(23480, 23470,-1)**:, vòng lặp này được sử dụng để xóa các hàng từ 23480 đến 23470 trong "data\_fake". Dòng **data\_fake.drop([i]), axis == 0, inplace == True** được sử dụng để xóa hàng có chỉ số **i** từ "data\_fake".

Tương tự, **dòng data\_true\_manual\_testing = data\_true.tail(10)** tạo ra một bản sao của 10 hàng cuối cùng trong "data\_true" và lưu vào "data\_true\_manual\_testing".

Trong vòng lặp **for i in range(21416, 21406,-1)**:, vòng lặp này được sử dụng để xóa các hàng từ 21416 đến 21406 trong "data\_true". Dòng **data\_true.drop([i]), axis == 0, inplace == True** được sử dụng để xóa hàng có chỉ số **i** từ "data\_true".

A screenshot of a computer

Description automatically generated

Manual testing (kiểm thử thủ công) là quá trình kiểm tra và xác nhận chức năng và tính năng của một phần mềm, ứng dụng hoặc hệ thống thông qua việc thực hiện các bước kiểm tra một cách thủ công. Trong quá trình kiểm thử thủ công, người thử nghiệm sẽ thực hiện các bước kiểm tra bằng tay, đối chiếu kết quả với các kết quả mong đợi và ghi lại các lỗi và vấn đề mà họ tìm thấy.

BƯỚC 6 : **Assigning Classes to the Dataset ( GÁN NHÃN CHO TẬP DỮ LIỆU )**

A close-up of a test

**data\_fake\_manual\_testing['class'] = 0**

**data\_true\_manual\_testing['class'] = 1**

có nghĩa là bạn đang thêm một cột mới có tên là "class" vào các tập dữ liệu "data\_fake\_manual\_testing" và "data\_true\_manual\_testing" và gán giá trị 0 cho cột "class" trong "data\_fake\_manual\_testing" và gán giá trị 1 cho cột "class" trong "data\_true\_manual\_testing".

Giả sử bạn đã tạo các tập dữ liệu "data\_fake\_manual\_testing" và "data\_true\_manual\_testing" từ các bước trước đó. Bằng cách thêm cột "class" và gán giá trị 0 cho "data\_fake\_manual\_testing['class']" và gán giá trị 1 cho "data\_true\_manual\_testing['class']", bạn đang gắn nhãn các mẫu dữ liệu trong tập "data\_fake\_manual\_testing" và "data\_true\_manual\_testing" là "0" và "1" tương ứng.

BƯỚC 7 : Merging Both the DATASET( GỘP CẢ 2 TẬP DỮ LIỆU )



A white rectangular object with a black border

Description automatically generated

Hàm pd.concat trong thư viện pandas là một phương thức để nối (concatenate) các đối tượng dữ liệu theo một hoặc nhiều trục. Nó cho phép bạn kết hợp các đối tượng dữ liệu, chẳng hạn như các DataFrame hoặc Series, thành một đối tượng dữ liệu lớn hơn



**data\_merge = pd.concat([data\_fake, data\_true], axis=0) :**

là bạn đang tạo một DataFrame mới được gọi là "data\_merge" bằng cách nối (concatenate) các DataFrame "data\_fake" và "data\_true" theo chiều dọc (axis=0).

**data\_merge.head(10) :**

được sử dụng để hiển thị 10 hàng đầu tiên của DataFrame "data\_merge". Phương thức head() trả về một tập dữ liệu con gồm các hàng đầu tiên của DataFrame, giúp bạn nhanh chóng xem qua dữ liệu.

==🡺 data\_merge = pd.concat([data\_fake, data\_true], axis=0) :

là bạn đang tạo một DataFrame mới được gọi là "data\_merge" bằng cách nối (concatenate) các DataFrame "data\_fake" và "data\_true" theo chiều dọc (axis=0).

**data\_merge.head(10) :**

được sử dụng để hiển thị 10 hàng đầu tiên của DataFrame "data\_merge". Phương thức head() trả về một tập dữ liệu con gồm các hàng đầu tiên của DataFrame, giúp bạn nhanh chóng xem qua dữ liệu.

A screenshot of a computer

Description automatically generated

BƯỚC 8 : Dropping Unwanted Columns" (Loại bỏ các cột không mong muốn)



|  |  |
| --- | --- |
|  | thư viện pandas, hàm drop() được sử dụng để loại bỏ (xóa) các hàng hoặc cột từ một đối tượng dữ liệu như DataFrame hoặc Series. |

|  |  |
| --- | --- |
|  | **data = data\_merge.drop(['title', 'subject', 'date'], axis=1) :**  bạn đang xóa các cột có nhãn là 'title', 'subject' và 'date' từ DataFrame "data\_merge" và gán kết quả vào biến "data".  Trong trường hợp này, **drop()** được áp dụng trên DataFrame "data\_merge" và các cột có nhãn 'title', 'subject' và 'date' sẽ bị loại bỏ. Tham số **axis=1** được sử dụng để chỉ ra rằng các cột sẽ bị xóa (loại bỏ). Kết quả của phép xóa sẽ được gán vào biến "data", nên biến "data" sẽ chứa DataFrame mới sau khi xóa các cột đã nêu.  Sau dòng mã này, **"data"** sẽ là một phiên bản của "data\_merge" mà không chứa các cột 'title', 'subject' và 'date'. |

Bước 9 : Create a Function to Clean Text( tạo 1 hàm để làm sạch văn bản

A computer code with text

Description automatically generated

A screenshot of a computer

Description automatically generated

|  |  |
| --- | --- |
| **lower()** là một phương thức được áp dụng trên một chuỗi (string) để chuyển đổi tất cả các ký tự chữ hoa thành chữ thường. Ví dụ: "Hello".lower() sẽ trả về chuỗi "hello". | **re.escape()** là một hàm trong module re (regular expression) của Python. Nó được sử dụng để đảm bảo các ký tự đặc biệt trong một chuỗi được xem như là các ký tự chữ thường. Điều này làm cho các ký tự đặc biệt không được xem như các thành phần của biểu thức chính quy (regular expression). Ví dụ: re.escape("Hello.") sẽ trả về chuỗi "Hello\.". |
| **re.sub()** là một hàm trong module re của Python được sử dụng để thay thế các chuỗi phù hợp với biểu thức chính quy (regular expression) bằng một giá trị mới. Cú pháp: re.sub(pattern, replacement, string). Ví dụ: re.sub("[0-9]", "-", "abc123xyz") sẽ trả về chuỗi "abc---xyz" | **string.punctuation** là một chuỗi trong module string của Python chứa các ký tự đặc biệt thường được coi là dấu câu. Chuỗi này bao gồm các ký tự như "!", ".", ",", "?", và nhiều ký tự đặc biệt khác. Nó thường được sử dụng để xác định các ký tự dấu câu trong một chuỗi và thực hiện các thao tác xử lý chuỗi liên quan đến dấu câu. |

A computer code with text

Description automatically generated

**wordopt(text),** được sử dụng để xử lý và làm sạch một chuỗi văn bản. Dưới đây là ý nghĩa của từng dòng mã:

**text = text.lower():** chuyển đổi toàn bộ chuỗi văn bản text thành chữ thường.

**text = re.sub('\[.\*?\]','',text):** sử dụng hàm re.sub() để loại bỏ các chuỗi nằm trong cặp dấu [...] trong text.

**text = re.sub("\\W"," ", text):** sử dụng hàm re.sub() để thay thế mọi ký tự không phải là chữ cái, chữ số hoặc dấu gạch dưới bằng dấu cách.

**text = re.sub('https?://\S+|www\.\S+','',text):** sử dụng hàm re.sub() để loại bỏ các đường dẫn URL và tên miền trong text.

**text = re.sub('<.\*?>+','', text):** sử dụng hàm re.sub() để loại bỏ các thẻ HTML (<...>) trong text.

**text = re.sub('[%s]'% re.escape(string.punctuation),'', text):** sử dụng hàm re.sub() để loại bỏ các ký tự đặc biệt (dấu câu) trong text bằng cách sử dụng string.punctuation.

**text = re.sub('\n','', text):** sử dụng hàm re.sub() để loại bỏ các ký tự xuống dòng (\n) trong text.

**text = re.sub('\w\*\d\w\*', '', text):** sử dụng hàm re.sub() để loại bỏ các từ chứa chữ số trong text.

Cuối cùng, hàm trả về giá trị text sau khi đã qua quá trình xử lý và làm sạch.

hàm wordopt(text) được thiết kế để xử lý và làm sạch một chuỗi văn bản, loại bỏ các thành phần không mong muốn như chuỗi nằm trong cặp dấu [...], đường dẫn URL, thẻ HTML, ký tự đặc biệt, ký tự xuống dòng và các từ chứa chữ số. Kết quả trả về là một phiên bản làm sạch của chuỗi ban đầu.

BƯỚC 10 : applying function to text column and assigning X and Y (Cấp dữ liệu cho hàm và gán X và Y)

A screenshot of a computer code

Description automatically generated

=>>>> hàm apply() được sử dụng để áp dụng một hàm hoặc một phương thức lên một trục của DataFrame.

A screenshot of a computer

Description automatically generated

**data['text'] = data['text'].apply(wordopt) :**

áp dụng hàm wordopt() lên cột 'text' của DataFrame data và gán kết quả vào cột 'text'.

Trong trường hợp này, data['text'] truy cập đến cột 'text' trong DataFrame data. Hàm apply() được gọi trên cột này, và hàm wordopt() được áp dụng lên từng phần tử của cột 'text'. Kết quả của wordopt() sẽ được gán trở lại cột 'text' của DataFrame data.

A screenshot of a computer code

Description automatically generated

**x = data['text'] và y = data['class']** là việc gán giá trị của cột 'text' và 'class' trong DataFrame data cho các biến x và y.

**x = data['text']** gán giá trị của cột 'text' trong DataFrame data cho biến x. Điều này có nghĩa là biến x sẽ chứa một Series chứa nội dung của cột 'text' trong DataFrame data. Biến x thường được sử dụng để lưu trữ các đặc trưng hoặc dữ liệu đầu vào cho một mô hình học máy.

**y = data['class']** gán giá trị của cột 'class' trong DataFrame data cho biến y. Điều này có nghĩa là biến y sẽ chứa một Series chứa nội dung của cột 'class' trong DataFrame data. Biến y thường được sử dụng để lưu trữ nhãn hoặc dữ liệu đầu ra tương ứng với dữ liệu đầu vào trong biến x.

Bước 11 : Defining Training and Testing Data and Splitting Them Into &5 -25 Percent Ratio( Định nghĩa dữ liệu huấn luyện và kiểm tra và chia chúng vào tỷ lệ 75-25 phần trăm)



**x** là dữ liệu đầu vào, thường là một Series hoặc DataFrame chứa các đặc trưng hoặc thuộc tính.

**y** là dữ liệu đầu ra tương ứng với **x**, thường là một Series hoặc mảng chứa nhãn hoặc dữ liệu đầu ra.

test\_size xác định tỷ lệ (phần trăm) của dữ liệu sẽ được sử dụng cho tập kiểm tra. Giá trị của test\_size thường nằm trong khoảng từ 0 đến 1, ví dụ: **0.25** cho tập kiểm tra chiếm **25%** tổng số dữ liệu.

Kết quả của **train\_test\_split()** là bốn tập dữ liệu: **x\_train, x\_test, y\_train, y\_test**.

**x\_train** là dữ liệu đầu vào được sử dụng để huấn luyện mô hình. Đây là một phần của x được chia từ tập dữ liệu ban đầu.

**x\_test** là dữ liệu đầu vào được sử dụng để kiểm tra và đánh giá mô hình đã huấn luyện. Đây là một phần của **x** được chia từ tập dữ liệu ban đầu.

**y\_train** là dữ liệu đầu ra tương ứng với **x\_train**. Đây là một phần của y được chia từ tập dữ liệu ban đầu.

**y\_test** là dữ liệu đầu ra tương ứng với **x\_test**. Đây là một phần của y được chia từ tập dữ liệu ban đầu.

Tổng quát, dòng mã **x\_train**, **x\_test**, **y\_train**, **y\_test** **=** **train\_test\_split(x**, **y**, **test\_size=0.25)** chia tập dữ liệu **x** và **y** thành hai tập con **x\_train**, **x\_test**, **y\_train**, **y\_test**, trong đó **x\_train** và **y\_train** được sử dụng để huấn luyện mô hình và **x\_test** và **y\_test** được sử dụng để kiểm tra và đánh giá mô hình.

**train\_test\_split()**

sử dụng để chia dữ liệu thành hai tập dữ liệu con: tập huấn luyện (training set) và tập kiểm tra (test set).

Bước 12: Converting Raw Data Into Matrix for Further Process( Chuyển đổi dữ liệu thô thành ma trận để tiếp tục xử lý)

A screen shot of a computer code

Description automatically generated

A screenshot of a computer

Description automatically generated

**fit\_transform()** là một phương thức trong TfidfVectorizer (và cũng là phương thức chung trong scikit-learn) được sử dụng để thực hiện quy trình fit và transform cùng một lúc.

scikit-learn, **TfidfVectorizer** là một công cụ để chuyển đổi văn bản thành một ma trận các giá trị TF-IDF. TF-IDF là một phương pháp để đánh giá tầm quan trọng của một từ trong một văn bản.

Bước **fit** trong **fit\_transform()** tính toán các tham số cần thiết từ tập dữ liệu đầu vào. Ví dụ, trong trường hợp của TfidfVectorizer, bước này tính toán các tham số cần thiết để tính toán giá trị TF-IDF cho các từ trong tập dữ liệu văn bản.

Bước **transform** trong **fit\_transform()** sử dụng các tham số đã tính toán từ bước trước để chuyển đổi dữ liệu đầu vào thành ma trận các giá trị TF-IDF. Kết quả là một ma trận có số hàng tương ứng với số mẫu văn bản và số cột tương ứng với số từ trong tập từ vựng.

Vì **fit\_transform()** kết hợp cả hai bước fit và transform, nó cung cấp một cách thuận tiện để tính toán giá trị TF-IDF từ dữ liệu văn bản một cách đồng thời.

A screen shot of a computer code

Description automatically generated

**TfidfVectorizer** được import từ module **sklearn.feature\_extraction.text**. Đây là một công cụ trong thư viện scikit-learn để chuyển đổi văn bản thành một ma trận các giá trị TF-IDF.

**vectorization = TfidfVectorizer()** tạo một đối tượng vectorization từ lớp TfidfVectorizer. Đối tượng này sẽ được sử dụng để chuyển đổi dữ liệu văn bản thành ma trận các giá trị TF-IDF.

**xv\_train = vectorization.fit\_transform(x\_train)** sử dụng phương thức **fit\_transform()** để tính toán các tham số từ tập dữ liệu huấn luyện x\_train và chuyển đổi nó thành ma trận xv\_train chứa các giá trị TF-IDF tương ứng.

**xv\_test = vectorization.transform(x\_test)** sử dụng phương thức **transform()** để sử dụng các tham số đã tính toán từ **x\_train** để chuyển đổi tập dữ liệu kiểm tra **x\_test** thành ma trận **xv\_test** chứa các giá trị TF-IDF tương ứng.

Kết quả của quá trình này là hai ma trận **xv\_train** và **xv\_test** chứa các giá trị TF-IDF tương ứng cho tập dữ liệu huấn luyện và tập dữ liệu kiểm tra. Các ma trận này có cùng số hàng, tương ứng với số mẫu dữ liệu, nhưng số cột có thể khác nhau tùy thuộc vào số từ vựng trong tập dữ liệu và các tham số cấu hình trong **TfidfVectorizer**

Bước 13 : creating first model

A white background with black text

Description automatically generated

A screenshot of a computer

Description automatically generated

thư viện scikit-learn, LogisticRegression là một mô hình học máy dùng để thực hiện phân loại dựa trên thuật toán hồi quy logistic.

LR.fit() là một phương thức trong lớp LogisticRegression được sử dụng để huấn luyện (fit) mô hình dựa trên dữ liệu huấn luyện.

A white background with black text

Description automatically generated

**LogisticRegression** được **import** từ **module** **sklearn.linear\_model.** Đây là một lớp trong thư viện scikit-learn để xây dựng mô hình hồi quy logistic.

**LR = LogisticRegression()** tạo một đối tượng LR từ lớp LogisticRegression. Đối tượng này sẽ được sử dụng để xây dựng mô hình hồi quy logistic.

**LR.fit(xv\_train, y\_train)** sử dụng phương thức **fit()** để huấn luyện mô hình LR trên dữ liệu huấn luyện xv\_train và nhãn tương ứng y\_train.

Khi gọi **LR.fit(xv\_train, y\_train),** mô hình logistic regression sẽ được huấn luyện trên ma trận các giá trị TF-IDF **xv\_train** và nhãn **y\_train**. Quá trình huấn luyện sẽ tìm ra các tham số tốt nhất cho mô hình dựa trên thuật toán hồi quy logistic.

Sau khi hoàn thành quá trình huấn luyện, mô hình LR đã được huấn luyện và có thể được sử dụng để dự đoán nhãn của các mẫu dữ liệu mới thông qua phương thức **predict().**

Bước 14 : **Checking the Model Accuracy and Classification Report**

A close-up of a code

Description automatically generated

A screenshot of a computer

Description automatically generated

**pred\_lr = LR.predict(xv\_test)** sử dụng phương thức **predict()** của mô hình LR để dự đoán nhãn của dữ liệu kiểm tra **xv\_test**. Kết quả được lưu vào biến **pred\_lr.**

**LR.score(xv\_test, y\_test)** sử dụng phương thức **score()** của mô hình LR để tính toán độ chính xác trên dữ liệu kiểm tra **xv\_test** và nhãn tương ứng **y\_test**. Độ chính xác là tỷ lệ giữa số mẫu dự đoán đúng và tổng số mẫu.

**classification\_report(y\_test, pred\_lr)** sử dụng hàm **classification\_report()** để tính toán báo cáo phân loại dựa trên nhãn thực tế **y\_test** và nhãn dự đoán pred\_lr. Báo cáo phân loại cung cấp thông tin về độ chính xác, độ phủ và các thông số khác liên quan đến phân loại.

Các thông số trong báo cáo phân loại bao gồm precision (độ chính xác), recall (độ phủ), f1-score (điểm F1) và support (số lượng mẫu trong mỗi lớp). Báo cáo phân loại giúp đánh giá hiệu suất của mô hình phân loại trên dữ liệu kiểm tra.

Bước 15 : **Creating a Second Model.**

A close-up of a computer code

Description automatically generated

A screenshot of a computer

Description automatically generated

=🡺DecisionTreeClassifier sử dụng thuật toán cây quyết định để xây dựng mô hình phân loại dựa trên các thuộc tính đầu vào và nhãn của dữ liệu huấn luyện. Quá trình xây dựng cây quyết định dựa trên việc tìm kiếm các điểm chia dữ liệu sao cho tối ưu hóa một độ đo, chẳng hạn như độ tinh purity hoặc entropy.

DecisionTreeClassifier là một lớp trong thư viện scikit-learn được sử dụng để xây dựng mô hình phân loại dựa trên cây quyết định (decision tree). Cây quyết định là một cấu trúc dữ liệu dạng cây, trong đó mỗi nút đại diện cho một điều kiện hoặc một thuộc tính và các nhánh từ nút đại diện cho các giá trị của thuộc tính. Các nút lá của cây đại diện cho các lớp hoặc nhãn của dữ liệu.

A close-up of a computer code

Description automatically generated

from **sklearn.tree** import DecisionTreeClassifier import lớp DecisionTreeClassifier từ module sklearn.tree.

**DT = DecisionTreeClassifier()** tạo một đối tượng DT từ lớp DecisionTreeClassifier. Đối tượng này sẽ được sử dụng để xây dựng mô hình phân loại dựa trên cây quyết định.

**DT.fit(xv\_train, y\_train)** sử dụng phương thức **fit()** để huấn luyện mô hình DT trên dữ liệu huấn luyện **xv\_train** và nhãn tương ứng **y\_train**.

Khi gọi **DT.fit(xv\_train, y\_train),** mô hình cây quyết định sẽ được huấn luyện trên ma trận các giá trị TF-IDF xv\_train và nhãn **y\_train**. Quá trình huấn luyện sẽ xây dựng cây quyết định dựa trên thuật toán cây quyết định và tìm kiếm các điểm chia dữ liệu tối ưu.

Sau khi hoàn thành quá trình huấn luyện, mô hình DT đã được huấn luyện và có thể được sử dụng để dự đoán nhãn của các mẫu dữ liệu mới thông qua phương thức predict().

Bước 16 : **Checking the Model Accuracy and Classification Report**A close up of text

Description automatically generated

**pred\_dt = DT.predict(xv\_test)** sử dụng phương thức predict() của mô hình DT để dự đoán nhãn của dữ liệu kiểm tra xv\_test. Kết quả được lưu vào biến pred\_dt.

**DT.score(xv\_test, y\_test)** sử dụng phương thức **score()** của mô hình DT để tính toán độ chính xác trên dữ liệu kiểm tra xv\_test và nhãn tương ứng y\_test. Độ chính xác là tỷ lệ giữa số mẫu dự đoán đúng và tổng số mẫu.

**classification\_report(y\_test, pred\_dt)** sử dụng hàm **classification\_report()** để tính toán báo cáo phân loại dựa trên nhãn thực tế y\_test và nhãn dự đoán pred\_dt. Báo cáo phân loại cung cấp thông tin về độ chính xác, độ phủ và các thông số khác liên quan đến phân loại.

Các thông số trong báo cáo phân loại bao gồm precision (độ chính xác), recall (độ phủ), f1-score (điểm F1) và support (số lượng mẫu trong mỗi lớp). Báo cáo phân loại giúp đánh giá hiệu suất của mô hình phân loại trên dữ liệu kiểm tra.

A screenshot of a computer

Description automatically generated

Bước 17 : **Checking Fake News**

A computer screen shot of a computer program

Description automatically generated

**output\_lable(n)** là một hàm được định nghĩa để chuyển đổi nhãn số thành nhãn văn bản. Nếu n là 0, hàm sẽ trả về chuỗi 'Fake News', và nếu n là 1, hàm sẽ trả về chuỗi 'Not a Fake News'.

**manual\_testing(news)** là một hàm dùng để kiểm tra thủ công một đoạn văn bản news mới. Quá trình kiểm tra bao gồm các bước sau:

Tạo một **DataFrame new\_def\_test** chứa văn bản kiểm tra trong cột 'text'.

Áp dụng hàm wordopt để tiền xử lý văn bản trong cột **'text'** của **new\_def\_test.**

Trích xuất văn bản đã được tiền xử lý **new\_x\_test** từ cột '**text**' của **new\_def\_test**.

Sử dụng **vectorization.transform()** để chuyển đổi new\_**x\_tes**t thành ma trận các giá trị TF-IDF **new\_xv\_test.**

Sử dụng mô hình Logistic Regression (LR) để dự đoán nhãn **pred\_LR** của **new\_xv\_test.**

Sử dụng mô hình Decision Tree (DT) để dự đoán nhãn pred\_DT của **new\_xv\_test.**

Cuối cùng, hàm sẽ in ra dự đoán của mô hình Logistic Regression và Decision Tree dưới dạng chuỗi văn bản.

Việc sử dụng hàm **print()** trong cuối hàm **manual\_testing(news**) chỉ để in ra kết quả dự đoán và không trả về giá trị nào.

**pred\_LR = LR.predict(new\_xv\_test)** sử dụng mô hình Logistic Regression (LR) để dự đoán nhãn của new\_xv\_test. Kết quả dự đoán được lưu trong biến pred\_LR.

**pred\_DT = DT.predict(new\_xv\_test)** sử dụng mô hình Decision Tree (DT) để dự đoán nhãn của new\_xv\_test. Kết quả dự đoán được lưu trong biến pred\_DT.

**return print('\n\nLR Prediction: {} \nDT Prediction:'.format(output\_lable(pred\_LR[0]), output\_lable(pred\_DT[0])))** in kết quả dự đoán của mô hình Logistic Regression và Decision Tree. Kết quả được định dạng và hiển thị thông qua hàm print(). Cụ thể, kết quả dự đoán của mô hình Logistic Regression và Decision Tree sẽ được truyền vào trong chuỗi định dạng 'LR Prediction: {}' và 'DT Prediction: {}' để in ra. Hàm output\_lable() được sử dụng để chuyển đổi nhãn dự đoán từ số sang chuỗi văn bản tương ứng.

A computer screen shot of a computer program

Description automatically generated

**news = str(input())** hàm input() để nhận đầu vào từ người dùng dưới dạng chuỗi. Hàm str() được sử dụng để đảm bảo rằng đầu vào được chuyển đổi thành kiểu dữ liệu chuỗi.

**manual\_testing(news)** gọi hàm **manual\_testing(news)** để thực hiện kiểm tra thủ công trên đoạn văn bản news. Hàm này sẽ tiền xử lý và dự đoán nhãn của đoạn văn bản đó bằng cách sử dụng mô hình Logistic Regression (LR) và Decision Tree (DT). Kết quả dự đoán sẽ được in ra màn hình.

**KẾT LUẬN CHUNG :**

Sau bài học, trainning data thật sự cần phải lấy những dữ liệu chuẩn xác nhất có thể, để có thể đạt được độ chính xác cao nhất có thể để nhận diện trong việc phát hiện Fake News, việc sử dụng những hàm phương thức tính toán cũng khá là khó, chúng em cũng phải thảo luận rất nhiều về việc sử dụng những hàm chuẩn xác từ nhiều nguồn cung cấp uy tín khác nhau như là w3school, những trang thuật toán iT.

Sau bài học chúng em rút ra được nhiều kinh nghiệm trong việc làm việc nhóm, chọn lọc dữ liệu, rút ra được nhiều kinh nghiệm trong việc triển khai data training, rút ra được nhiều bài học quý giá trong việc thực hiện triển khai cùng một nhóm mới lạ. Là sinh viên chúng em sẽ đút kết được nhiều kinh nghiệm quý giá, trau dòi kiến thức, học tập tốt hơn trong lĩnh vực phân tích dữ liệu học sâu này.

**Tài Liệu Tham Khảo:**

<https://www.simplilearn.com/tutorials/machine-learning-tutorial/how-to-create-a-fake-news-detection-system>

Link Github: <https://github.com/Daisy0401/Doanmonhoc>