1. **I. Cơ sở lý thuyết để xây dựng mô hình**
   1. **Quy trình xây dựng mô hình Machine Learning/Deep learning phân loại tài liệu văn bản**
      1. **Tổng quan**

Diagram

Description automatically generated

Hình 2. 3 Sơ đồ mô hình phân loại văn bản

Bài toán phân loại văn bản (Text Classification) là một bài toán học giám sát (supervised learning) trong học máy (machine learning). Bài toán này yêu cầu dữ liệu cần có nhãn (label). Mô hình sẽ học từ dữ liệu có nhãn đó, sau đó được dùng để dự đoán nhãn cho các dữ liệu mới mà mô hình chưa gặp.

Mục tiêu của một hệ thống phân loại văn bản là nó có thể tự động phân loại một văn bản cho trước, để xác định xem văn bản đó thuộc thể loại nào.

Một số ứng dụng của hệ thống phân loại như:

* + - * 1. Hiểu được ý nghĩa, đánh giá, bình luận của người dùng từ mạng xã hội.
        2. Phân loại emails là spam hay không spam.
        3. Tự động gắn thẻ cho những truy vấn, tìm kiếm của người dùng.
        4. Phân loại các bài báo điện tử.
    1. **Tiền xử lý dữ liệu văn bản**

Đây là bước đầu tiên cần làm, tiền xử lý dữ liệu là quá trình chuẩn hóa dữ liệu và loại bỏ các thành phần không có ý nghĩa cho việc phân loại văn bản.

Tiền xử lý dữ liệu tiếng Việt cho bài toán phân loại văn bản thường gồm các việc sau:

* + - * 1. Xóa HTML code: Dữ liệu được thu thập từ các Website có thể vẫn còn sót lại các đoạn mã HTML. Các đoạn mã HTML gây ảnh hưởng đến kết quả phân loại văn bản nên cần phải bị loại bỏ. Áp dụng Regex giúp cho việc loại bỏ các đoạn mã HTML trở nên đơn giản hơn:

regex.sub(r'<[^>]\*>', '', text)

Text

Description automatically generated

Hình 2. 4 Loại bỏ HTML code

* + - * 1. Chuẩn hóa bảng mã Unicode: Hiện nay, có 2 loại mã Unicode được sử dụng phổ biến, Unicode tổ hợp và Unicode dựng sẵn. Nhìn bình thường thì 2 bảng mã này khá là giống nhau tuy nhiên đối với máy tính thì khác, 2 bảng mã này phân biệt với nhau. Nếu không xử lý vấn đề này thì khi đưa vào mô hình học máy tính sẽ hiểu đó là các từ khác nhau mặc dù ta đang nhìn thấy chúng chẳng khác nhau gì. Vì vậy cần phải chuẩn hoá chúng lại cùng một bảng mã để tránh gây ảnh hưởng đến quá trình học.

Text

Description automatically generated

Hình 2. 5 Chuẩn hoá về UTF8

* + - * 1. Chuẩn hóa kiểu gõ dấu tiếng Việt: tương tự như việc sử dụng bảng mã Unicode thì vị đặt dấu cũng ảnh hưởng đến kết quả học của mô hình. Ví dụ: òa úy với oà uý

Text

Description automatically generated

Hình 2. 6 Ảnh hưởng của vị trí đặt dấu thanh

import regex as re

uniChars = "àáảãạâầấẩẫậăằắẳẵặèéẻẽẹêềếểễệđìíỉĩịòóỏõọôồốổỗộơờớởỡợùúủũụưừứửữựỳýỷỹỵÀÁẢÃẠÂẦẤẨẪẬĂẰẮẲẴẶÈÉẺẼẸÊỀẾỂỄỆĐÌÍỈĨỊÒÓỎÕỌÔỒỐỔỖỘƠỜỚỞỠỢÙÚỦŨỤƯỪỨỬỮỰỲÝỶỸỴÂĂĐÔƠƯ"

unsignChars = "aaaaaaaaaaaaaaaaaeeeeeeeeeeediiiiiooooooooooooooooouuuuuuuuuuuyyyyyAAAAAAAAAAAAAAAAAEEEEEEEEEEEDIIIOOOOOOOOOOOOOOOOOOOUUUUUUUUUUUYYYYYAADOOU"

def loaddicchar():

    dic = {}

    char1252 = 'à|á|ả|ã|ạ|ầ|ấ|ẩ|ẫ|ậ|ằ|ắ|ẳ|ẵ|ặ|è|é|ẻ|ẽ|ẹ|ề|ế|ể|ễ|ệ|ì|í|ỉ|ĩ|ị|ò|ó|ỏ|õ|ọ|ồ|ố|ổ|ỗ|ộ|ờ|ớ|ở|ỡ|ợ|ù|ú|ủ|ũ|ụ|ừ|ứ|ử|ữ|ự|ỳ|ý|ỷ|ỹ|ỵ|À|Á|Ả|Ã|Ạ|Ầ|Ấ|Ẩ|Ẫ|Ậ|Ằ|Ắ|Ẳ|Ẵ|Ặ|È|É|Ẻ|Ẽ|Ẹ|Ề|Ế|Ể|Ễ|Ệ|Ì|Í|Ỉ|Ĩ|Ị|Ò|Ó|Ỏ|Õ|Ọ|Ồ|Ố|Ổ|Ỗ|Ộ|Ờ|Ớ|Ở|Ỡ|Ợ|Ù|Ú|Ủ|Ũ|Ụ|Ừ|Ứ|Ử|Ữ|Ự|Ỳ|Ý|Ỷ|Ỹ|Ỵ'.split(

        '|')

    charutf8 = "à|á|ả|ã|ạ|ầ|ấ|ẩ|ẫ|ậ|ằ|ắ|ẳ|ẵ|ặ|è|é|ẻ|ẽ|ẹ|ề|ế|ể|ễ|ệ|ì|í|ỉ|ĩ|ị|ò|ó|ỏ|õ|ọ|ồ|ố|ổ|ỗ|ộ|ờ|ớ|ở|ỡ|ợ|ù|ú|ủ|ũ|ụ|ừ|ứ|ử|ữ|ự|ỳ|ý|ỷ|ỹ|ỵ|À|Á|Ả|Ã|Ạ|Ầ|Ấ|Ẩ|Ẫ|Ậ|Ằ|Ắ|Ẳ|Ẵ|Ặ|È|É|Ẻ|Ẽ|Ẹ|Ề|Ế|Ể|Ễ|Ệ|Ì|Í|Ỉ|Ĩ|Ị|Ò|Ó|Ỏ|Õ|Ọ|Ồ|Ố|Ổ|Ỗ|Ộ|Ờ|Ớ|Ở|Ỡ|Ợ|Ù|Ú|Ủ|Ũ|Ụ|Ừ|Ứ|Ử|Ữ|Ự|Ỳ|Ý|Ỷ|Ỹ|Ỵ".split(

        '|')

    for i in range(len(char1252)):

        dic[char1252[i]] = charutf8[i]

    return dic

dicchar = loaddicchar()

# Đưa toàn bộ dữ liệu qua hàm này để chuẩn hóa lại

def covert\_unicode(txt):

    return re.sub(

        r'à|á|ả|ã|ạ|ầ|ấ|ẩ|ẫ|ậ|ằ|ắ|ẳ|ẵ|ặ|è|é|ẻ|ẽ|ẹ|ề|ế|ể|ễ|ệ|ì|í|ỉ|ĩ|ị|ò|ó|ỏ|õ|ọ|ồ|ố|ổ|ỗ|ộ|ờ|ớ|ở|ỡ|ợ|ù|ú|ủ|ũ|ụ|ừ|ứ|ử|ữ|ự|ỳ|ý|ỷ|ỹ|ỵ|À|Á|Ả|Ã|Ạ|Ầ|Ấ|Ẩ|Ẫ|Ậ|Ằ|Ắ|Ẳ|Ẵ|Ặ|È|É|Ẻ|Ẽ|Ẹ|Ề|Ế|Ể|Ễ|Ệ|Ì|Í|Ỉ|Ĩ|Ị|Ò|Ó|Ỏ|Õ|Ọ|Ồ|Ố|Ổ|Ỗ|Ộ|Ờ|Ớ|Ở|Ỡ|Ợ|Ù|Ú|Ủ|Ũ|Ụ|Ừ|Ứ|Ử|Ữ|Ự|Ỳ|Ý|Ỷ|Ỹ|Ỵ',

        lambda x: dicchar[x.group()], txt)

* + - * 1. Thực hiện tách từ tiếng Việt: để tăng hiệu quả học của mô hình, thay vì sử dụng những từ đơn thì có thể sử dụng một số thư viện có sẵn để nhận diện từ ghép như pyvi, underthesea, vncorenlp,…

Text

Description automatically generated

Hình 2. 7 Sử dụng Pyvi để tách từ

Text

Description automatically generated

Hình 2. 8 Sử dụng underthesea để tách từ

* + - * 1. Đưa về văn bản chữ viết thường: Việc đưa dữ liệu về chữ viết thường là rất cần thiết. Bởi vì đặc trưng này không có tác dụng ở bài toán phân loại văn bản. Đưa về chữ viết thường giúp giảm số lượng đặc trưng (vì máy tính hiểu hoa thường là 2 từ khác nhau) và tăng độ chính xác hơn cho mô hình.

A screenshot of a computer

Description automatically generated with medium confidence

Hình 2. 9 Đưa văn bản về chữ viết thường

* + - * 1. Xoá bỏ các dữ liệu không cần thiết: ền xử lý bao gồm việc loại bỏ các dữ liệu không có tác dụng cho việc phân loại văn bản. Việc này giúp:
  + Giảm số chiều đặc trưng, tăng tốc độ học và xử lý
  + Tránh làm ảnh hưởng xấu tới kết quả của mô hình

Việc xoá bỏ các dữ liệu không cần thiết gồm có

* Xóa các ký tự đặc biệt: Các dấu ngắt câu, số đếm và các ký tự đặc biệt khác không giúp bạn phân loại một văn bản thuộc chuyên mục nào.

gensim.utils.simple\_preprocess(text)

Text

Description automatically generated

Hình 2. 10 Loại bỏ các ký tự đặc biệt sử dụng gensim module

* Xoá những từ thường xuất hiện nhiều ở tất cả các danh mục cần lọc (stopword) nên chúng thường không mang đặc trưng dùng cho việc phân loại văn bản.

Ý tưởng: Chúng ta sẽ thống kê các từ xuất hiện nhiều trong tất cả các chuyên mục và lấy top đầu trong đó thêm vào bộ từ điển stopword. Việc này cũng khá đơn giản tuy nhiên bộ từ điển stopword phải được xây dựng từ bộ dữ liệu văn bản lớn (tầm vài chục GB)

Một số stopword: là, và, của, có, được, trong, cho với, một, hai, ba,…

def remove\_stopwords(line):

    words = []

    for word in line.strip().split():

        if word not in stopword:

            words.append(word)

    return ' '.join(words)

* + 1. **Xây dựng mô hình**
       1. **Xây dựng tập train/test**

Để biết được mô hình của chúng ta chính xác bao nhiêu, liệu có thể sử dụng được trong thực tế không? Liệu có bị underfitting hay overfitting hay không? Khi nào bắt đầu overfitting để dừng tăng độ phức tạp. Để kiểm thử khả dự đoán khả năng hoạt động hiệu quả của mô hình trên thực tế. Có rất nhiều cách để kiểm thử và đánh giá hiệu năng của mô hình.

Cách đơn giản nhất là đưa ra dùng thử một thời gian. Cách này khá tốn kém thời gian, cũng như khá tốn công để thiết kế các kịch bản kiểm thử phù hợp, hoặc trong nhiều trường hợp cũng khó sử dụng.

Cách thứ hai hợp lý hơn đó là chúng ta sử dụng một tập dữ liệu kiểm thử (testing set) độc lập đối với tập dữ liệu huấn luyện (training set) để đánh giá vầ ước lượng hiệu quả của mô hình. Thông thường training set và testing set được tách ra ngay từ dữ liệu quan sát được cung cấp (chẳng hạn lấy 80% dữ liệu quan sát được cung cấp để huấn luyện và 20% dữ liệu quan sát còn lại không liên quan đến 80% trước để đánh giá).

Trong đa số các trường hợp thì người ta sẽ dùng cách thứ 2. Để sử dụng cách này thì sau khi hoàn thành bước tiền xử lý thì dữ liệu cần phải chuyển dữ liệu đã được xử lý về dạng vector thuộc tính có dạng số học để phù hợp với việc học của mô hình và sau đó là đưa dữ liệu đó và bộ cơ sở dữ liệu (dataset) phù hợp để tiến hành huấn luyện và đánh giá.

Có nhiều cách khác nhau để đưa dữ liệu về dạng phù hợp:

* + Count Vectors as features: Khi sử dụng phương pháp này, chúng ta sẽ thu được một ma trận mà trong đó, mỗi hàng sẽ đại diện cho một văn bản, mỗi cột đại diện cho một từ có trong từ điển, và mỗi ô (cell) sẽ chứa tần suất xuất hiện của từ trong văn bản tương ứng
  + TF-IDF Vectors as features: đây là một phương pháp cực kì phổ biến trong xử lý văn bản. Nó được tính theo công thức dưới đây:
    - TF(t) = (Number of times term t appears in a document) / (Total number of terms in the document)
    - IDF(t) = log\_e(Total number of documents / Number of documents with term t in it)

TF-IDF được phân thành 3 loại cấp độ:

* + - Word level: Thực hiện tính toán dựa trên mỗi thành phần là một từ riêng.
    - N-Gram level: Kết hợp n thành phần (từ) liên tiếp nhau
    - Character level: Dựa trên n-gram của ký tự
  + Word Embeddings as features: Bằng cách sử dụng thuật toán Bag-of-words để chuyển mỗi từ trong từ điển về một vector n chiều.
  + Text / NLP based features
  + Topic Models as features

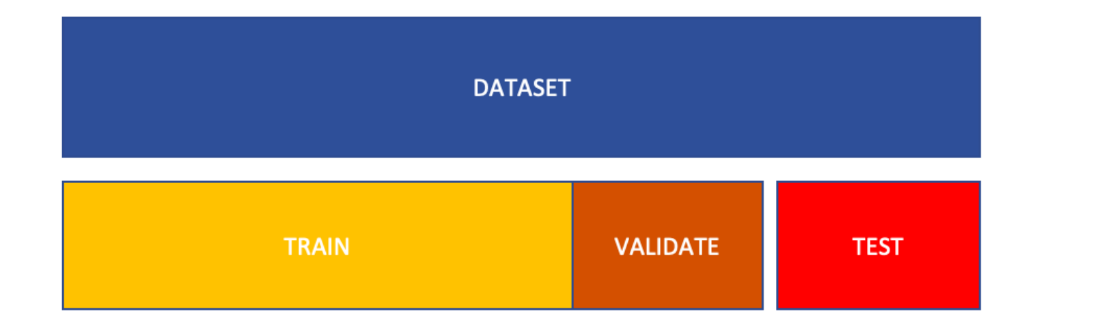
Sau khi xử lý dữ liệu và đưa nó về dạng vector thì chúng ta cần phải sắp xếp chúng vào bộ cơ sở dữ liệu để sử dụng trong quá trình huấn luyện và đánh giá. Về cơ bản, có 3 cách chia tập dữ liệu thường được sử dụng trên thực tế đó là:

Diagram

Description automatically generated with medium confidence

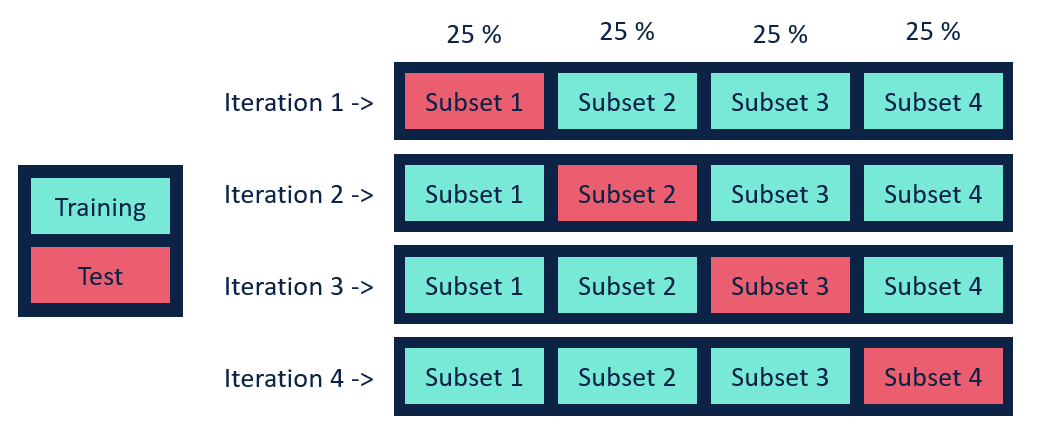
Hình 2. 11 Hình ảnh mô hoạ quá trình Hold-out dataset

* + Hold-out: Đây là phương pháp chỉ chia tập dữ liệu quan sát được cung cấp thành hai tập con, trong đó một tập để huấn luyện (thường chiếm 80% lượng dữ liệu hoặc 70% dữ liệu) và một tập được bỏ ra (Hold-out) ngoài tập huấn luyện đóng vai trò là testing set, là phần còn lại của bộ dữ liệu được cung cấp (thường là 20% lượng dữ liệu hoặc 30% dữ liệu còn lại).



Hình 2. 12 Hình ảnh minh hoạ quá trình Train-Validation-Test Split dataset

* + Train-Validation-Test Split: phương pháp này có cách thức chia tương tự như trường hợp Hold-out, nhưng là để riêng ra một bộ dữ liệu để kiểm thử (testing set), và trong lượng dữ liệu còn lại lại chia nhỏ thành 2 phần, 1 phần để train mô hình (thường khoảng 80% lượng dữ liệu còn lại), và một phần để kiểm chứng và hiệu chỉnh mô hình (validation – 20% dữ liệu còn lại).



Hình 2. 13 Hình ảnh minh hoạ quá trình Cross Validation dataset

* + Cross Validation: Đây là phương pháp chủ yếu được sử dụng để đánh giá thuật toán huấn luyện hoặc dạng của mô hình huấn luyện chứ không phải mô hình huấn luyện. Khi đó, quá trình huấn luyện và kiểm thử sẽ được thực hiện lặp đi lặp lại nhiều lần. Trong mỗi lần tập dữ liệu quan sát sẽ được lần lượt chia thành 2 phần (training và testing), phần training được sử dụng cho việc huấn luyện, còn phần testing được sử dụng cho kiểm thử. Sau đó kết quả kiểm thử tại mỗi lần sẽ được cộng lại và lấy trung bình. Nếu kết quả trung bình tốt có nghĩa là thuật toán huấn luyện hoặc dạng của mô hình được sử dụng tốt và có độ ổn định cao. Cách chia tập training – testing trong cross validation lại khác nhau dựa trên số lượng dữ liệu được chia, trong đó có thể kể đến như: Leave-p-out cross-validation (để ra một lượng dữ liệu là p quan sát làm kiểm thử), Leave-one-out cross-validation (lấy 1 điểm dữ liệu duy nhất để kiểm thử sau mỗi lần), và phổ biến nhất là k-fold validation (chia bộ dữ liệu thành k-phần, trong đó 1 phần đễ kiểm thử và k−1 phần để huấn luyện). Ý tường chính ở đây là chúng ta sẽ thực hiện toàn bộ quá trình huấn luyện - đánh giá k lần và tính trung bình độ chính xác qua các lần. Ví dụ: trong 5 -fold cross validation, chúng ta sẽ lấy 20% của bộ dữ liệu đễ tính toán accuracy, precision, recall, F1Score... Sau đó, chúng ta sẽ thực hiện kiểm thử tiếp 20% dữ liệu thứ hai và tính toán lại các thống kê này. Quy trình này lặp đi lặp lại 5 lần và mỗi lần testing set sẽ là một phân đoạn dữ liệu khác nhau. Cuối cùng chúng ta tính trung bình tất cả các độ chính xác, chúng ta sẽ hiểu rō hơn về cách mô hình của chúng ta hoạt động.
    - * Tuy nhiên cho dù dùng cách nào để đánh giá mô hình chúng ta cần xây dựng một tập kiểm thử (testing set) riêng biệt và không có bất cứ liên hệ gì với tập dữ liệu huấn luyện. Tập này dùng để đánh giá hiệu quả của mô hình được xây dựng.
      1. **Phân loại văn bản với Naive Bayes**

Thuật toán phân loại Naive Bayes được xếp vào loại phương pháp Supervised Learning (Học có giám sát). Thuật toán Naive Bayes tính xác suất cho các yếu tố, sau đó chọn kết quả với xác suất cao nhất.

Thuật toán này được xây dựng dựa trên công thức tính xác suất Bayes

Text

Description automatically generated

Hình 2. 14 Công thức Bayes

Đưa công thức Bayes về dạng 2 biến A và B ta sẽ được

Công thức chỉ ra xác suất của A xảy ra nếu B cũng xảy ra, ta viết là P(A|B). Và nếu ta biết xác suất của B xảy ra khi biết A, ta viết là P(B|A) cũng như xác suất độc lập của A và B.

* + P(A|B) là “xác suất của A khi biết B”
  + P(A) là xác suất xảy ra của A
  + P(B|A) là “xác suất của B khi biết A”
  + P(B) là xác suất xảy ra của B
    - * 1. Một số mô hình sử dụng Naive Bayes:
  + Gaussian Naive Bayes: Mô hình này được sử dụng chủ yếu trong loại dữ liệu mà các thành phần là các biến liên tục.

Diagram

Description automatically generated

Hình 2. 15 Naive Bayes được ứng dụng trong Gaussian Naive Bayes

* + Multinomial Naive Bayes: Mô hình này chủ yếu được sử dụng trong phân loại văn bản mà feature vectors được tính bằng Bags of Words. Lúc này, mỗi văn bản được biểu diễn bởi một vector có độ dài d chính là số từ trong từ điển. Giá trị của thành phần thứ i trong mỗi vector chính là số lần từ thứ i xuất hiện trong văn bản đó.

Diagram

Description automatically generated with medium confidence

Hình 2. 16 Naive Bayes được ứng dụng trong thuật toán Multinomial Naive Bayes

Trong đó

* + - Nyi là tổng số lần từ thứ i xuất hiện trong các văn bản của class c, nó được tính là tổng của tất cả các thành phần thứ i của các feature vectors ứng với class c.
    - Ny là tổng số từ (kể cả lặp) xuất hiện trong class c. Nói cách khác, nó bằng tổng độ dài của toàn bộ các văn bản thuộc vào class c
    - Với α là một số dương, thường bằng 1, để tránh trường hợp tử số bằng 0. Mẫu số được cộng với αn để đảm bảo tổng xác suất = 1
  + Bernoulli Naive Bayes: Mô hình này được áp dụng cho các loại dữ liệu mà mỗi thành phần là một giá trị binary - bẳng 0 hoặc 1.

A picture containing text, clock, watch

Description automatically generated

Hình 2. 17 Naive Bayes được ứng dụng trong thuật toán Bernoulli Naive Bayes

Với với p(xi=1|y) có thể được hiểu là xác suất từ thứ i xuất hiện trong các văn bản của class c.

* + Ngoài ra còn có các mô hình khác như
    - Complement Naive Bayes

Text, letter

Description automatically generated

Hình 2. 18 Naive Bayes trong Complement Naive Bayes

* + - Categorical Naive Bayes

Text

Description automatically generated with medium confidence

Hình 2. 19 Naive Bayes trong Categorical Naive Bayes

* + - * 1. Ưu nhược điểm
  + Ưu điểm
    - Dễ sử dụng và nhanh khi cần đoán nhãn của dữ liệu test. Thực hiện khá tốt trong multi class prediction (test later).
    - Khi giả định rằng các feature của dữ liệu là độc lập với nhau thì Naive Bayes chạy tốt hơn so với các thuật toán khác như logistic regression và cũng cần ít dữ liệu hơn.
  + Nhược điểm
    - Độ chính xác của Naive Bayes nếu so với các thuật toán khác thì không được cao.
    - Trong thế giới thực, hầu như bất khả thi khi các feature của dữ liệu test là độc lập với nhau.
      * 1. Ứng dụng: dựa vào những ưu và nhược điểm được nêu ở trên thì thuật toán này rất thích hợp dùng trong việc phân loại văn bản và lọc spam do phần lớn ở tốc độ của nó.
      1. **Phân loại văn bản với Logistic Regression**

Logistic Regression là 1 thuật toán phân loại được dùng để gán các đối tượng cho 1 tập hợp giá trị rời rạc (như 0, 1, 2, ...).

Một vài ví dụ điển hình là phân loại Email, gồm có email công việc, email gia đình, email spam, ... Giao dịch trực tuyến có là an toàn hay không an toàn, khối u lành tính hay ác tính.

Để hiểu về mô hình này đầu tiên cần phải hiểu về mô hình tuyến tính Linear Regression được xây dựng dưới công thức:

Text

Description automatically generated

Hình 2. 20 Công thức Linear Regression

trong đó f() được gọi là activation function, và x được hiểu là dữ liệu mở rộng với x0=1 được thêm vào để thuận tiện cho việc tính toán

Như tên của nó, thuật toàn Linear Regression chỉ phù hợp với các bài toán Binary Classification. Nếu muốn sử dụng nó cho bài toán phân loại với nhiều classification hơn, thuật toán **Logistic Regression** ra đời để giải quyết vấn đề này. Trong mô hình này, đầu ra có thể được thể hiện dưới dạng xác suất (probability) và nó giống với linear regression ở khía cạnh đầu ra là số thực. Tuy nhiên ở mô hình Logistic Regression thì đầu ra của nó bị chặn trong đoạn [0, 1].

Đầu ra dự đoán của logistic regression thường được viết chung dưới dạng:

Text

Description automatically generated with low confidence

Hình 2. 21 Dạng đầu ra thuật toán Logistic Regression

Theo dạng ở trên thì hàm Sigmoid (hay còn gọi là hàm Logistic) là hàm được sử dụng nhiều nhất vì nó bị chặn trong khoảng [0,1] thoả điều kiện của Logistic regression

Chart, box and whisker chart

Description automatically generated

Hình 2. 22 Hàm Sigmoid

* + - * 1. Ưu nhược điểm của thuật toán Logistic Regression
  + Ưu điểm
    - Dễ thực hiện
    - Hoạt động tốt đối với các trường hợp tập dữ liệu có thể phân tách tuyến tính
  + Nhược điểm
    - Có thể không chính xác nếu kích thước mẫu quá nhỏ
      * 1. Một vài ứng dụng của Logistic Regression
  + Dự đoán email có phải spam hay không
  + Dự đoán giao dịch ngân hàng là gian lận hay không
  + Dự đoán khối u lành hay ác tính
  + Dự đoán khoản vay có trả được không
  + Dự đoán khoản đầu tư vào start-up có sinh lãi hay không.
    - 1. **Phân loại văn bản với Random Forest**

Random forest là thuật toán supervised learning, được sử dụng để giải quyết cả bài toán regression và classification.

Random là ngẫu nhiên, Forest là rừng, nên ở thuật toán Random Forest mình sẽ xây dựng nhiều cây quyết định bằng thuật toán Decision Tree, tuy nhiên mỗi decision tree sẽ khác nhau (có yếu tố random). Sau đó kết quả dự đoán được tổng hợp từ các decision tree.

A picture containing shape

Description automatically generated

Hình 2. 23 Hình minh hoạ Decision tree

Thuật toán này được xây dựng thì nhiều cây nhị phân khác nhau. Ở bước dự đoán, với một dữ liệu mới, thì ở mỗi cây quyết định mình sẽ đi từ trên xuống theo các node điều kiện để được các dự đoán, sau đó kết quả cuối cùng được tổng hợp từ kết quả của các cây quyết định.

Thuật toán Random Forest hoạt động như thế nào:

* + Chọn các mẫu ngẫu nhiên từ tập dữ liệu đã cho.
  + Thiết lập cây quyết định cho từng mẫu và nhận kết quả dự đoán từ mỗi quyết định cây.
  + Hãy bỏ phiếu cho mỗi kết quả dự đoán.
  + Chọn kết quả được dự đoán nhiều nhất là dự đoán cuối cùng.

Giả sử kết quả dự đoán được chỉ ra từ 6 decision tree như sau

A picture containing shape

Description automatically generated

Hình 2. 24 Decision tree đưa ra dự đoán

Có thể thấy là có 6 cây đưa ra dự đoán, trong đó có 5 cây 1 và 1 cây 0 -> Kết quả dự đoán sẽ là 1

* + - * 1. Ưu nhược điểm thuật toán Random Forest:
  + Ưu điểm: Random forests được coi là một phương pháp chính xác và mạnh mẽ vì số cây quyết định tham gia vào quá trình này. Nó không bị vấn đề overfitting. Lý do chính là nó mất trung bình của tất cả các dự đoán, trong đó hủy bỏ những thành kiến. Thuật toán có thể được sử dụng trong cả hai vấn đề phân loại và hồi quy. Random forests cũng có thể xử lý các giá trị còn thiếu. Có hai cách để xử lý các giá trị này: sử dụng các giá trị trung bình để thay thế các biến liên tục và tính toán mức trung bình gần kề của các giá trị bị thiếu.
  + Nhược điểm: Random forests chậm tạo dự đoán bởi vì nó có nhiều cây quyết định. Bất cứ khi nào nó đưa ra dự đoán, tất cả các cây trong rừng phải đưa ra dự đoán cho cùng một đầu vào cho trước và sau đó thực hiện bỏ phiếu trên đó. Toàn bộ quá trình này tốn thời gian. Mô hình khó hiểu hơn so với decision tree, nơi bạn có thể dễ dàng đưa ra quyết định bằng cách đi theo đường dẫn trong cây.
    - * 1. Ứng dụng của thuật toán Random Forest: do tính chất độ chính xác cao nhưng tốc độ chậm nên random forest phù hợp với bài toán đơn giản nhưng yêu cầu độ chính xác cao. Và nó không phù hợp với các bài toán với yêu cầu filter lớn (tốc độ chậm).
      1. **Một số thuật toán phân loại khác**
         1. **Phân loại văn bản với SVM**

SVM (Support Vector Machine) là một thuật toán học máy có giám sát được sử dụng rất phổ biến ngày nay trong các bài toán phân lớp (classification) hay hồi qui (Regression).

SVM được đề xuất bởi Vladimir N. Vapnik và các đồng nhiệp của ông vào năm 1963 tại Nga và sau đó trở nên phổ biến trong những năm 90 nhờ ứng dụng giải quyết các bài toán phi tuyến tính (nonlinear) bằng phương pháp Kernel Trick.

Có 2 loại của thuật toán SVM

* + SVM: dùng cho các bài toán phân lớp
  + SVR (Support Vector Regression): dùng cho các bài toán hồi quy
    - * 1. **Phân loại văn bản với XGBoost**

XGBoost là một trong những thuật toán học máy phổ biến và được sử dụng rộng rãi nhất hiện nay vì đơn giản là nó rất mạnh mẽ. Nó là một giải thuật được base trên gradient boosting, tuy nhiên kèm theo đó là những cải tiến to lớn về mặt tối ưu thuật toán, về sự kết hợp hoàn hảo giữa sức mạnh phần mềm và phần cứng, giúp đạt được những kết quả vượt trội cả về thời gian training cũng như bộ nhớ sử dụng.

XGBoost có thể được sử dụng để giải quyết được tất cả các vấn đề từ hồi quy (regression), phân loại (classification), ranking và giải quyết các vấn đề do người dùng tự định nghĩa.XGBoost hỗ trợ trên Windows, Linux và OS X.Hỗ trợ tất cả các ngôn ngữ lập trình chính bao gồm C ++, Python, R, Java, Scala và Julia.Hỗ trợ các cụm AWS, Azure và Yarn và hoạt động tốt với Flink, Spark và các hệ sinh thái khác....

* + 1. **Đánh giá mô hình**
       1. **Tại sao phải đánh giá mô hình học máy**

Khi bạn đã xây dựng một mô hình machine learning và huấn luyện nó trên một tập dữ liệu, điều tiếp theo bạn nên làm là đánh giá hiệu năng của mô hình trên tập dữ liệu mới.

* + - * 1. Việc đánh giá mô hình giúp chúng ta giải quyết những vấn đề sau:
  + Mô hình đã được huấn luyện thành công hay chưa?
  + Mức độ thành công của mô hình tốt đến đâu?
  + Khi nào nên dừng quá trình huấn luyện?
  + Khi nào nên cập nhật mô hình?
* Trả lời được 4 câu hỏi trên, chúng ta có thể quyết định mô hình này có thực sự phù hợp cho bài toán hay không.

Chúng ta sử dụng dữ liệu mới khi đánh giá mô hình nhằm giảm thiểu khả năng quá khớp (overfitting) đối với tập huấn luyện. Đôi khi sẽ hữu ích khi đánh giá mô hình và cùng lúc huấn luyện nó để tìm ra các chỉ số tốt nhất của một mô hình. Tuy nhiên, chúng ta không thể sử dụng bộ thử nghiệm để thực hiện đánh giá này. Hoặc chúng ta sẽ phải chọn các thông số hoạt động tốt nhất trên dữ liệu thử nghiệm, nhưng có thể không phải là các tham số bao quát nhất.

* + - 1. **Một số cách đánh giá mô hình**

Một mô hình được đánh giá tốt nếu như tỉ lệ: TP (True Positive) và TN (True negative) lớn đồng thời tỉ lệ FP (False Positive) và FN (False Negative) là nhỏ trong đó:

* + TP: số lượng các mẫu thuộc lớp dương được phân chính xác vào lớp dương.
  + FP: số lượng các mẫu không thuộc lớp dương bị phân loại nhầm vào lớp dương.
  + TN: số lượng các mẫu không thuộc lớp dương được phân loại đúng.
  + FN: số lượng các mẫu thuộc lớp dương bị phân nhầm vào các lớp không phải lớp dương.

Độ chính xác là độ gần của phép đo với một giá trị cụ thể, độ chính xác của mô hình được tính bằng công thức sau:

Precision được tính bằng công thức sau:

Recall chính là True Positive Rate (TPR) , còn được gọi là tỷ lệ dương tính thực sự hoặc độ nhạy, là xác suất mà mô hình xác định chính xác sự bất thường được phát hiện.

False Positive Rate (FPR) là tỷ lệ cảnh báo sai, được tính bằng công thức sau:

F1-Score là giá trị trung bình của Precision và Recall. Nó cho phép so sánh duy nhất và cao hơn là tốt hơn.

* 1. **Quy trình xây dựng mô hình Machine learning/Deep Learning phân loại tài liệu gắn nhãn hình ảnh**
     1. **Tổng quan**

Phân loại hình ảnh (Image classification) hay Nhận dạng hình ảnh (Image recognition) là một trong những tác vụ của thị giác máy tính, ở đó thuật toán xem xét và dán nhãn cho hình ảnh từ một tập danh mục được xác định và đào tạo trước.

Ví dụ, với một tập các hình ảnh, mỗi hình ảnh mô tả một con mèo hoặc một con chó, thuật toán sẽ “quan sát” toàn bộ dữ liệu và dựa trên hình dạng, màu sắc để hình thành giả thuyết liên quan đến nội dung của ảnh. Kết quả thu được là từ tập dữ liệu ban đầu, các hình ảnh chó/mèo đã được phân loại một cách tự động.

Thực tế, thị giác góp phần tạo nên 80-85% nhận thức của con người về thế giới. Hàng ngày, mỗi người phải thực hiện phân loại trên bất kỳ dữ liệu hình ảnh nào mà chúng ta bắt gặp.

Do đó, mô phỏng nhiệm vụ phân loại với sự trợ giúp của mạng nơ-ron là một trong những ứng dụng đầu tiên của thị giác máy tính mà các nhà nghiên cứu nghĩ đến.

* + - * 1. Các loại mô hình phân loại hình ảnh
  + Phân loại có giám sát (supervised learning): thuật toán được huấn luyện trên một tập hình ảnh đã được dán nhãn. Từ dữ liệu mẫu này, thuật toán có thể trích xuất thông tin, phục vụ phân loại ngay cả những hình ảnh chưa từng nhìn thấy trước đó.
  + Phân loại không giám sát (unsupervised learning): thuật toán chỉ sử dụng dữ liệu thô để đào tạo.Các nhãn phân loại thường không xuất hiện trong kiểu học này và mô hình học bằng cách nhận dạng các mẫu trong tập dữ liệu huấn luyện.
    - * 1. Mô hình phân loại hình ảnh hoạt động như thế nào

Máy tính xử lý một hình ảnh dưới dạng pixel. Theo đó, hình ảnh chỉ là một mảng ma trận, và kích thước của ma trận phụ thuộc vào độ phân giải hình ảnh.

Do đó, xử lý hình ảnh là tiến hành phân tích dữ liệu toán học với sự trợ giúp của các thuật toán. Các thuật toán này chia nhỏ hình ảnh thành một tập hợp các đặc điểm nổi bật, giúp giảm khối lượng công việc của bộ phân loại cuối cùng.

Quá trình trích xuất đặc điểm là bước quan trọng nhất trong việc phân loại hình ảnh. Phân loại, đặc biệt là phân loại có giám sát, phụ thuộc phần lớn vào dữ liệu được cung cấp cho thuật toán. Một bộ dữ liệu phân loại tốt phải đảm bảo các yêu cầu về sự cân bằng của dữ liệu, chất lượng của ảnh và chú giải kèm theo.

* + 1. **Xử lý dữ liệu**

Ở bước này đầu tiên cần phải làm sau khi tìm được nguồn dữ liệu phù hợp cho mô hình. Xử lý dữ liệu ở đây tuỳ thuộc vào loại mô hình sử dụng nhưng chính vẫn là chuyển dữ liệu về dạng phù hợp, có tình đồng bộ về mặt tính chất (kiểu dữ liệu, kích thước,…). Một trong những công việc chính trong bước này đối với mô hình phân loại tài liệu hình ảnh có thể kể đến như:

* + - * 1. Giải nén dữ liệu

Tuỳ theo loại dữ liệu nén mà sẽ có những cách giải nén dữ liệu khác nhau. Ngoài ra còn có thể giải nèn bằng ứng dụng (giao diện đồ hoạ) hoặc console (bằng lệnh). Một số cách giải nén hình ảnh thông qua console:

* + Rar file: `unrar x -Y <địa chỉ file cần giải nén> <nơi giải nén đến>` (tôi dùng ở trên colab của google)

A picture containing text

Description automatically generated

Hình 2. 25 Giải nén file rar trên google colab

* + Zip file: `unzip <địa chỉ file cần giải nén> -d <nơi cần giải nén đến>`



Hình 2. 26 Giải nén file zip trên google colab

* + - * 1. Thay đổi kích thước hình ảnh: dữ liệu hình ảnh được sử dụng cho quá trình xây dựng mô hình có thể là dữ liệu chưa được đồng bộ về mặt kích thước gây ảnh hưởng đến quá trình xây dựng hoặc những dữ liệu hình ảnh này có kích thước quá lớn khiến quá trình xây dựng trở nên tốn thời gian và tài nguyên phần cứng (tuy nhiên kích thước ảnh lớn có thể giúp có độ chính xác của mô hình tăng lên). Vì vậy cần phải thay đổi kích thước hình ảnh để tăng hiệu quả cho quá trình xây dựng mô hình:
        2. Tạo biến để lưu giá trị hình ảnh: Sau khi xử lý hình ảnh thì cần phải lưu nó vào biến để thuận tiện cho quá trính xây dựng mô hình.

Ngoài ra nều dữ liệu chưa được phân chia sẵn thì cần phải chia dữ liệu ra thành các tập như tập dữ liệu huấn luyện (train set), tập dữ liệu đánh giá (validation), tập dữ liệu kiểm định (test set).

* + 1. **Các bước xây dựng mô hình phân loại tài liệu hình ảnh**

Sau khi có được bộ cơ sở dữ liệu (dataset) và chuẩn hoá dữ liệu của tập này, công việc tiếp theo trong quy trình đó là xây dựng mô hình dự đoán:

* + - * 1. Tạo mô hình (Creating model) : Tạo và nối các layers trong neural network lại với nhau (sử dụng functional or sequential API) hoặc import mô hình đã được train trước đó (transfer learning)
        2. Biên dịch mô hình (Compile model): Định nghĩa hiệu suất mà mô hình nên đo lường (loss/metrics) cũng như làm như thế nào để cải thiện mô hình đó (optimizer)
        3. Cải thiện mô hình (Fit model): Mô hình cố gắng tìm kiếm các khuôn mẫu (các điểm tương đồng hoặc có mối liên hệ) trong dữ liệu (hay nói khác là làm như thế nào để biến các đặc tính X thành y)
    1. **Đánh giá**

Bên cạnh việc quan sát trực quan mô hình phân tích dữ liệu bằng biểu đồ, chúng ta còn có thể thiết lập đánh giá tổng quan cho mô hình bằng các phương pháp toán học khác trong mô hình phân loại (đã được nêu trong phần quy trình cây dựng mô hình Machine learning phân loại tài liệu văn bản chương 2 phần 3)

1. **II. Ứng dụng vào việc xây dựng mô hình**
   1. **Mô hình Machine learning trong việc phân loại văn bản**
      1. **Ý tưởng**

Ý tưởng của việc xây dựng mô hình machine learning là tuân theo quy trình xây dựng mô hình machine learning được mô tả trước đó để tạo nên model phân loại bài báo dựa trên tập dữ liệu được gắn nhãn trước đó. Đầu tiên thì sẽ xử lý dữ liệu tiếng việt để chúng trở nên đồng bộ. Sau đó từ dữ liệu đã được xử lý trước đó để tạo thành các tập train/test phục vụ cho quá trình xây dựng và đánh giá model.

* + 1. **Mô hình**
       1. **Dataset**

Bộ cơ sở dữ liệu (dataset) được lấy từ opensource VNTC (A Large-scale Vietnamese News Text Classification Corpus) của tác giả Vu Hoang Cong Duy trên github của anh tại <https://github.com/duyvuleo/VNTC>

* + - 1. **Tiền xử lý dữ liệu**

Tại đây sử dụng những hàm được tinh chỉnh (được giới thiệu trước đó ở chương 2 lý thuyết tại phần 4.2 tiền xử lý dữ liệu) để xử lý và đồng bộ bộ cơ sở dữ liệu (dataset)

Graphical user interface, text, application

Description automatically generated

Hình 3. 1 Tiền xử lý dữ liệu

Sau khi chạy hàm preprocessing\_doc thì văn bản đầu sẽ bị:

* + Xóa HTML code
  + Chuẩn hóa bảng mã Unicode (đưa về Unicode tổ hợp dựng sẵn)
  + Chuẩn hóa kiểu gõ dấu tiếng Việt (dùng òa úy thay cho oà uý)
  + Thực hiện tách từ tiếng Việt
  + Đưa về văn bản lower (viết thường)
  + Xóa các ký tự đặc biệt: “.”, “,”, “;”, “)”, …
    - 1. **Chuyển dữ liệu sang dạng số**

Tập train/test chính là bộ dataset được xử lý trước đó (ở phần 2.2.3 Tiền xử lý dữ liệu). Ở bước này bộ cơ sở dữ liệu sẽ được xử lý để biến đổi từ dạng chữ ban đầu sang dạng vector để mô hình học máy có thể dễ dàng học được. Tại đây cách tôi sử dụng đó là

* + TF-IDF Vectors as features. Dữ liệu đầu vào sẽ được xử lý thông qua các cấp độ của TF-IDF và được lưu trong các biến:
  + X\_data\_tfidf và X\_test\_tfidf cho word level

Graphical user interface, text, application

Description automatically generated

Hình 3. 2 xử lý dữ liệu ở mức từ

* + X\_data\_tfidf\_ngram và X\_test\_tfidf\_ngram cho N-Gram level

Graphical user interface, text, application

Description automatically generated

Hình 3. 3 xử lý dữ liệu ở mức ngram

* + X\_data\_tfidf\_ngram\_char và X\_test\_tfidf\_ngram\_char cho N-Gram character level.

Graphical user interface, text, application

Description automatically generated

Hình 3. 4 xử lý dữ liệu ở mức ngram character

* Ngoài ra trong bài còn xử dụng SVD - singular value decomposition. Vì dữ liệu được tạo ra sau khi chuyển sang dạng số quá lớn, ảnh hưởng trực tiếp đến bộ nhớ RAM gây tốn kém tài nguyên. Để xử lý vấn đề này, chúng ta sẽ sử dụng thuật toán SVD (singular value decomposition) nhằm mục đích giảm chiều dữ liệu của ma trận mà chúng ta thu được, mà vẫn giữ nguyên được các thuộc tính của ma trận gốc ban đầu.

Graphical user interface, text, application

Description automatically generated

Hình 3. 5 Giảm chiều dữ liệu bằng SVD

* + - 1. **Label Encoder**

Sau khi chuyển dữ liệu sang dạng số thì những label (nhãn phân loại – filter) cũng phải được xử lý để chuyển sang dạng mà máy tính có thể hiểu được.

Graphical user interface, application

Description automatically generated

Hình 3. 6 Label Encoder

* + - 1. **Phân chia bộ cơ sở dữ liệu thành train/test**

Bộ dataset được tác giả chia làm 2 loại đó là train và test. Bộ cơ sở dữ liệu test đã được có sẵn nên sẽ được giữ nguyên. Nhưng bộ train sẽ được chia là 2 là train set và validation set theo tỷ lệ 80% với 20% bằng hàm train\_test\_split



Hình 3. 7 Chia bộ cơ sở dữ liệu thành train set, validation set và test set

Hàm train\_test\_split được sử dụng từ module sklearn.model\_selection của sklearn gồm các giá trị đầu vào được tuỳ chỉnh như sau:

* + X\_data là dữ liệu đầu vào
  + y\_data là dữ liệu nhãn
  + test\_size là tỉ lệ phân chia (ở đây là 20%)
  + random\_state là tham số ngẫu nhiên điều khiển sự ngẫu nhiên trong quá trình tách
    - 1. **Tiến hành train**

Tại đây tôi có xây dựng hàm chung để để training model dựa trên dữ liệu đầu vào và loại thuật toán sẽ được sử dụng (các thuật toán được sử dụng trong bài chính là các python module được build sẵn của Sklearn).

Text, letter

Description automatically generated

Hình 3. 8 training function

Các thuật toán được sử dụng để đánh giá bao gồm

* + naive\_bayes.MultinomialNB(): Mô hình này sử dụng lần lượt dữ liệu đầu vào là 3 cấp độ của TF-IDF
  + naive\_bayes.BernoulliNB(): mô hình này sử dụng dữ liệu đầu vào là TF-IDF cấp độ word sử dụng và không sử dụng thuật toán SVD để giảm chiều dữ liệu
  + linear\_model.LogisticRegression(): tương tự như đối với model naive\_bayes.BernoulliNB()
  + svm.SVC(): chỉ sử dụng TF-IDF đã được SVD
  + ensemble.RandomForestClassifier(): chỉ sử dụng TF-IDF đã được SVD
  + xgboost.XGBClassifier(): chỉ sử dụng TF-IDF đã được SVD
    - 1. **Vẽ chart đánh giá**

Thay vì nhìn thông số được đưa ra thì tôi quyết định sử dụng Chart (sử dụng thư viện matplotlib.pyplot) để biểu diễn giá trị giúp thuận tiện cho quá trình đánh giá.

Hàm vẽ chart: có 3 giá trị đầu vào là train\_value, validation\_value và test\_value chính là độ chính xác lần lượt đối với 3 tập dữ liệu tương ứng

Text

Description automatically generated

Hình 3. 9 chart fuction

Theo như hàm thì sẽ có 3 biểu đồ tròn (max=100%) biểu thị cho độ chính xác của thuật toán

* + 1. **Đánh giá**
       - 1. So sánh độ chính xác đối với 3 mức độ của phương pháp TF-IDF (sử dụng thuật toán Naïve Bayes để đánh giá)
  + Cấp độ từ

Chart, pie chart

Description automatically generated

Hình 3. 10 Sử dụng thuật toán Naive Bayes với TFIDF word

* + Cấp độ Ngram

Chart, pie chart

Description automatically generated

Hình 3. 11 Sử dụng thuật toán Naive Bayes với TFIDF NGram

* + Cấp độ Ngram char

Chart, pie chart

Description automatically generated

Hình 3. 12 Sử dụng thuật toán Naive Bayes với TFIDF NGram char

* Nhìn vào biểu đồ, dễ dàng nhận ra rằng độ chính xác đối với

TF-IDF Ngram > TF-IDF word > TF-IDF Ngram-char

Train accuracy: 89.16 > 88.92 > 81.11

Validation accuracy: 87.92 > 88.74 > 81.50

Test accuracy: 87.28 > 85.23 > 75.60

Có thể rút ra kết luận rằng việc kết hợp các từ lại với nhau với TF-IDF giúp cải thiện độ chính xác của thuật toán lên và việc sử dụng ở cấp độ ký tự thì sẽ mang lại độ chính xác thấp hơn đối với thuật toán sử dụng.

* + - * 1. So sánh độ chính xác đối với việc sử dụng hay không sử dụng thuật toán SVD giúp giảm chiều sâu vector
  + Sử dụng thuật toán: naive\_bayes.BernoulliNB()
    - Không sử dụng SVD

Chart, pie chart

Description automatically generated

Hình 3. 13 Sử dụng thuật toán Bernoulli Naive Bayes với TF-IDF không sử dụng SVD

* + - Có sử dụng SVD

Chart, pie chart

Description automatically generated

Hình 3. 14 Sử dụng thuật toán Bernoulli Naive Bayes với TF-IDF sử dụng SVD

* + Sử dụng thuật toán: linear\_model.LogisticRegression()
    - Không sử dụng SVD

Graphical user interface, chart, text, application, email, pie chart

Description automatically generated

Hình 3. 15 Sử dụng thuật toán Logistic Regression với TF-IDF không sử dụng SVD

* + - Sử dụng SVD

Graphical user interface, chart, text, application, email, pie chart

Description automatically generated

Hình 3. 16 Sử dụng thuật toán Logistic Regression với TF-IDF sử dụng SVD

* Trong cả 2 trường hợp xử dụng thuật toán Bernoulli Naïve Bayes hay Logistic Regression đều cho thấy việc sử dụng SVD làm giảm độ chính xác của thuật toán. Điều này hoàn toàn dễ hiểu bởi vì việc sử dụng SVD làm giảm số chiều của vector biểu diễn dữ liệu gây ảnh hưởng đến kết quả cũng như độ chính xác của thuật toán. Tuy nhiên việc sử dụng SVD giúp cải thiện thời gian chạy và giảm thiểu hao tốn tài nguyên bộ nhớ RAM.
  + - * 1. So sánh độ chính xác của các thuật toán được sử dụng (sử dụng dữ liệu đầu vào đã được xử lý TF-IDF có sử dụng SVD để so sánh)
  + Thuật toán Bernoulli Naïve Bayes

Chart, pie chart

Description automatically generated

Hình 3. 17 Thuật toán Bernoulli Naïve Bayes

* + Thuật toán Logistic Regression

Graphical user interface, chart, text, pie chart

Description automatically generated

Hình 3. 18 Thuật toán Logistic Regression

* + Thuật toán SVM

Graphical user interface, chart, pie chart

Description automatically generated

Hình 3. 19 Thuật toán SVM - Support Vector Machine

* + Thuật toán Random Forest

Chart, pie chart

Description automatically generated

Hình 3. 20 Thuật toán Random Forest

* + Thuật toán XGBoost (XGB)

Chart, pie chart

Description automatically generated

Hình 3. 21 Thuật toán XGB

* Việc sử dụng các thuật toán khác nhau mang đến độ chính xác khác nhau.
  + - * 1. Nếu xét về Train accuracy thì có thể thấy

Random Forest (99.66) > SVM (96.11) > Logistic Regression (92.51) >

XGB(91.31) > Bernoulli Naive Bayes (81.16)

* + - * 1. Nếu xét về Validation accuracy thì

SVM (93.60) > Logistic Regression (92.61) > Random Forest (90.83) >

XGB (89.78) > Bernoulli Naive Bayes (81.97)

* + - * 1. Còn xét về Test accuracy

Random Forest (94.55) > SVM (93.94) > Logistic Regression (91.33) >

XGB (89.63) > Bernoulli Naive Bayes (81.04)

* Từ những kết quả trên có thể rút ra được đánh giá:
  + Thuật toán Bernoulli Naive Bayes tuy có ưu thể về tốc độ nhưng độ chính xác cho ra lại tương đối thấp so với các thuật toán còn lại
  + XGB là thuật toán mới khá được ưa chuộng nhưng kết quả đưa ra lại không ấn tượng
  + Random Forest và SVM là 2 thuật toán chạy khá tốn thời gian nhưng bù lại thì độ chính xác của chúng khá cao
  + Logistic Regression là một thuật toán cân bằng được giữa thời gian và độ chính xác.
  1. **Mô hình Deep learning trong việc phân loại hình ảnh**
     1. **Ý tưởng**

Ý tưởng của mô hình là xây dựng mô hình dựa trên quy trình xây dựng mô hình cơ bản (được nêu ở chương 2 phần 4) và sử dụng convolutional neural network (CNN) – cụ thể ở đây là VGG16 được thiết kế sẵn trong bộ thư viện của tensorflow.keras để xây dựng mô hình.

* + 1. **Mô hình**
       1. **Dataset**

Bộ cơ sở dữ liệu (dataset) được lấy từ trang kaggle.com (“Kaggle” được biết đến là một công ty con, hoạt động dưới sự điều hành, quản lý của tập đoàn Google LLC).

Địa chỉ: <https://www.kaggle.com/competitions/dogs-vs-cats/data>

* + - 1. **Tiền xử lý dữ liệu**

Đầu tiên cần phải giải nén dữ liệu



Hình 3. 22 Giải nén dữ liệu

Dữ liệu sau khi được giải nén sẽ được thay đổi kích thước thành 160\*160 và lưu dữ liệu huấn luyện vào biến train\_dataset, dữ liệu kiểm định vào biến validation\_dataset.

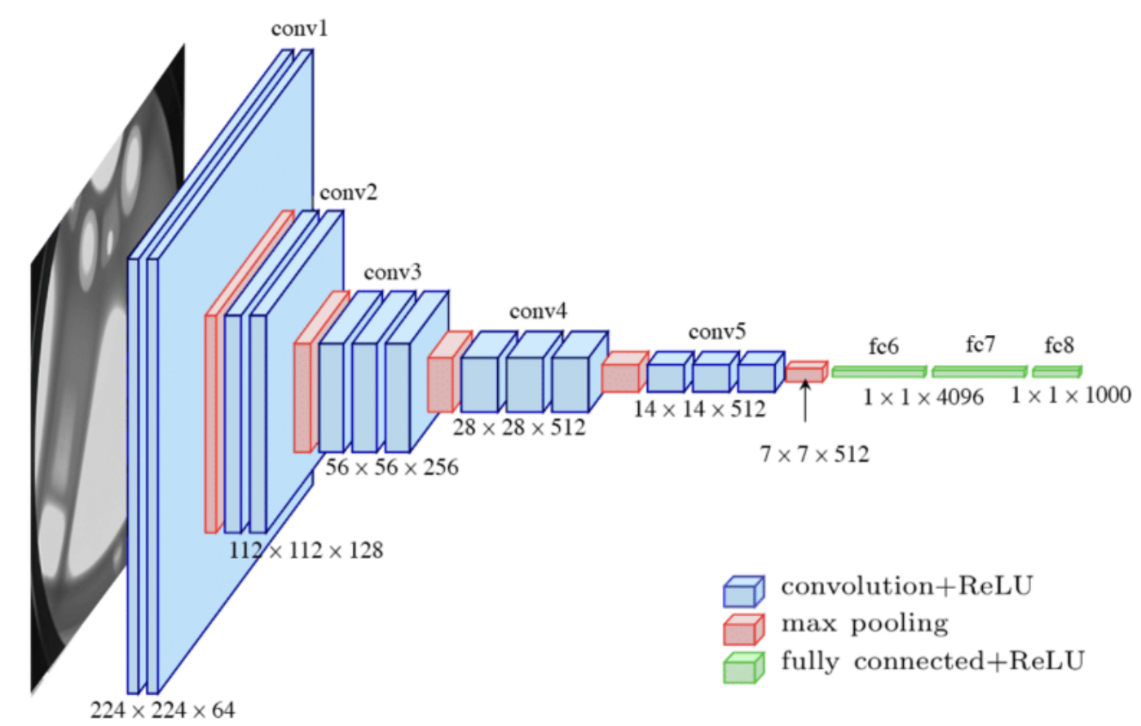
Ảnh có chứa văn bản

Mô tả được tạo tự động

Hình 3. 23 Đồng bộ dữ liệu và lưu dữ liệu vào biến

* + - 1. **Xây dựng mô hình**

Sau khi xử lý dữ liệu sẽ bến bước xây dựng mô hình. Mô hình được sử dụng ở đây là VGG16. VGG16 là một mô hình tiêu biểu của CNN (Convolutional neural network) - một mạng neural được ứng dụng rất nhiều trong deep learning trong computer vision cho classifier (mô hình phân loại đang được đề cập trong bài) và localizer.



Hình 3. 24 Mô hình VGG16

Nhìn vào mô hình của VGG16 có thể thấy mô hình gồm có

* + 16 layer:
    - 13 layer conv (2 layer conv-conv,3 layer conv-conv-conv)
    - 3 layer fully connection
  + Sau mỗi layer conv là maxpooling downsize xuống 0.5

Có thể tạm hiều là VGG16 trải qua 16 layer và 5 lần pooling làm cho hình ảnh từ 3 chiều dạng 224x224x64 thành 1x1x1000

Áp dụng VGG16 vào mô hình

Graphical user interface, text, application

Description automatically generated

Hình 3. 25 Xây dựng mô hình sử dụng VGG16

* + 1. **Đánh giá**
       - 1. Sử dụng matplotlib.pyplot để vẽ đồ thị

Text

Description automatically generated

Hình 3. 26 Vẽ đồ thị đánh giá với matplotlib.pyplot

* + - * 1. Đánh giá mô hình thông qua độ chính xác

Chart, line chart

Description automatically generated

Hình 3. 27 Đồ thị đánh giá độ chính xác của mô hình

* + - * 1. Đánh giá mô hình thông qua hàm mất mát (loss function)

Chart, histogram

Description automatically generated

Hình 3. 28 Đồ thị đánh giá sự mất mát của mô hình

1. **III. Ứng dụng model để phân loại tài liệu**
   1. **Phân loại bài báo**
      1. **Ý tưởng**

Ý tưởng của phần này chính là việc tôi sử dụng lại model được xây dựng ở phần trước (có tổng cộng 10 model) để xây dựng lên ứng dụng giúp phân loại bài báo.

* + 1. **Mô hình**
       - 1. Sử dụng thư viện tkinter trên python để dựng giao diện đồ hoạ cho ứng dụng
         2. Sử dụng Model được tạo ở phần trước để dùng trong việc phân loại
    2. **Xây dựng ứng dụng**
       1. **Giao diện chính**

Graphical user interface, text, application

Description automatically generated

Hình 4. 1 Giao diện chính của ứng dụng

Giao diện ứng dụng được xây dựng gồm 5 phần chính được đánh số

* + 1: danh mục các bài báo có thể phân loại (label)
  + 2: Vùng nhập bài báo
  + 3: Vùng cho phép chọn các model đã được training ở phần trước để đưa ra dự đoán
  + 4: Nơi hiển thị tên model đang được chọn
  + 5: Các nút (function) hổ trợ cho quá trình sử dụng
  + Ngoài ra còn có popup thông báo hiển thị kết quả phân loại
    - 1. **Coding**
         1. Tạo giao diện với Tkinter
  + Hiển thị danh mục phân loại

Text

Description automatically generated

Hình 4. 2 hàm hiển thị label

* + Hiển thị danh sách model

Graphical user interface, text

Description automatically generated

Hình 4. 3 Hiển thị danh sách model

* + Khởi tạo giao diện chính

Text

Description automatically generated

Hình 4. 4 Tạo giao diện chính

* + - * 1. Hàm sử lý giá trị đầu vào (bài báo)

Text

Description automatically generated

Hình 4. 5 Xử lý dữ liệu đầu vào

* + - * 1. Dự đoán và đưa ra kết quả

Text

Description automatically generated

Hình 4. 6 Dự đoán và đưa ra kết quả

* + 1. **Sử dụng**

Tại đây tôi sử dụng bài báo trên trang Vnexpress được viết ngày 25/8/2022 tại địa chỉ <https://vnexpress.net/5-dieu-can-biet-ve-champions-league-2022-2023-4503778.html>

* + - * 1. Kết quả dự đoán khi sử dụng Multinomial Naïve Bayes

Graphical user interface, text, application

Description automatically generated

Hình 4. 7 dự đoán sử dụng thuật toán naive\_bayes MultinomialNB

* + - * 1. Kết quả dự đoán khi sử dụng Random Forest

Graphical user interface, text, application, email

Description automatically generated

Hình 4. 8 Dự đoán thể loại bài báo sử dụng thuật toán Random Forest

* 1. **Phân loại hình ảnh**
     1. **Ý tưởng**

Sử dụng model được xây dựng ở phần trước để phân loại chó mèo thông qua hình ảnh được nhập vào.

* + 1. **Mô hình**

Ứng dụng được xây dựng trên môi trường colab của google nên khó để xây dựng mô hình đồ hoạ. Vì vậy mô hình của ứng dụng sẽ được triển khai như sau: người dùng nhập địa chỉ hình ảnh (trên mạng), ứng dụng sẽ tự lưu hình ảnh lại và sử dụng hình ảnh để đưa ra kết quả dự đoán

* + 1. **Xây dựng ứng dụng và sử dụng**
       - 1. Load model đã được xây dựng trước đó từ google drive

Graphical user interface, text, application, email

Description automatically generated

Hình 4. 9 Load model

* + - * 1. Load hình ảnh từ địa chỉ được nhập. Hình ảnh được chọn ở đây là hình ảnh con mèo có địa chỉ <https://kynguyenlamdep.com/wp-content/uploads/2022/01/hinh-anh-meo-con-sieu-cute-scaled.jpg> và in hình ảnh ra console bằng matplotlib.pyplot.

A cat lying on a blanket

Description automatically generated with low confidence

Hình 4. 10 Hình ảnh con mèo từ internet

Ảnh có chứa văn bản

Mô tả được tạo tự động

Hình 4. 11 Đoạn code xử lý hình ảnh từ mạng internet

* + - * 1. Tiến hành dự đoán và đưa ra kết quả

Graphical user interface, text, application

Description automatically generated

Hình 4. 12 Dự đoán và đưa ra kết quả