TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA CÔNG NGHỆ THÔNG TIN

Nhóm 3

EMOTION DETECTION FROM SPEECH

THỰC HÀNH GIỚI THIỆU NGHÀNH
TRÍ TUỆ NHÂN TẠO
24-TNT

GIÁO VIÊN HƯỚNG DẪN

TS. Lê Ngọc Thành
ThS. Phạm Trọng Nghĩa
ThS. Nguyễn Trần Duy Minh

Tp. Hồ Chí Minh, ngày 3 tháng 12 năm 2024

Thành viên trong nhóm

MSSV	Họ và Tên
24122002	Đinh Như Phát
24122004	Võ Lê Gia Huy
24122019	Từ Văn Khôi
24122021	Lê Bảo Phúc
24122028	Trần Xuân Bách
24122031	Trần Lê Minh Đức

Mục lục

Đánh giá thành viên

Đinh Như Phát

- **Nhiệm vụ:**
  - Chuẩn bị bộ dữ liệu (thu thập, làm sạch, tăng cường).
  - Quay video demo mô hình và kết quả bài toán.
  - Báo cáo kết quả và nhận xét độ chính xác của mô hình.
- **Mức độ hoàn thành: 100%**
- **Đánh giá:**
  - Thực hiện xuất sắc nhiệm vụ được giao, hoàn thành đúng tiến độ.
  - Đóng vai trò quan trọng trong viết báo cáo và chuẩn bị dữ liệu cũng như hoàn thành các mô hình.

Lê Bảo Phúc

- **Nhiệm vụ:**
  - Báo cáo tổng quan mô hình Extremely Randomized Trees.
  - Xây dựng ứng dụng.
  - Thiết kế slide và thuyết trình giới thiệu mô hình Extremely Randomized Trees.
- **Mức độ hoàn thành: 100%**
- **Đánh giá:**
  - Nhiệm vụ được giao hoàn thành đầy đủ và đúng tiến độ.
  - Phần thuyết trình và báo cáo đạt mức chất lượng tốt, rõ ràng.

Võ Lê Gia Huy

- **Nhiệm vụ:**
  - Báo cáo tổng quan mô hình KNN.
  - Xây dựng ứng dụng.
  - Thiết kế slide và thuyết trình giới thiệu mô hình KNN.
- **Mức độ hoàn thành: 100%**
- **Đánh giá:**
  - Hoàn thành tốt nhiệm được giao.
  - Đóng góp nổi bật trong phần video thiết kế slide, thuyết trình tốt, hỗ trợ truyền tải ý tưởng tốt.

Từ Văn Khôi

- **Nhiệm vụ:**
  - Báo cáo tổng quan mô hình Neural Network.
  - Huấn luyện mô hình KNN và Neural Network.
  - Thiết kế slide giải thích mô hình Neural Network.
- **Mức độ hoàn thành: 100%**
- **Đánh giá:**
  - Đóng góp tốt trong các báo cáo và huấn luyện mô hình.
  - Thiết kế slide tốt và hoàn thành đúng tiến độ công việc.

Trần Xuân Bách

- **Nhiệm vụ:**
  - Báo cáo trên Latex, xử lý phần giới thiệu và xử lý dữ liệu.

  - Kiểm tra báo cáo dataset, cấu hình thí nghiệm.
  - Hoàn thiện nội dung Powerpoint phần xử lý dữ liệu.

- **Mức độ hoàn thành: 100%**

- **Đánh giá:**

  - Phần LaTeX được hoàn thiện tốt, đáp ứng yêu cầu dự án.
  - Đảm bảo chất lượng bài báo cáo và nội dung thuyết trình.

Trần Lê Minh Đức

- **Nhiệm vụ:**

  - Báo cáo tổng quan mô hình Decision Tree.
  - Huấn luyện mô hình Decision Tree và Extra Tree.
  - Hoàn thiện nội dung Powerpoint phần tổng quan các mô hình.

- **Mức độ hoàn thành: 100%**

- **Đánh giá:**

  - Hoàn thành tốt công việc huấn luyện mô hình.
  - PowerPoint được điều chỉnh hợp lý, hỗ trợ tốt phần thuyết trình.

Đánh giá mức độ hoàn thành từng danh mục

Yêu cầu ngày họp 28/11: Chuẩn bị bộ dữ liệu và tìm hiểu các mô hình

- **Yêu cầu**
  - Phân công công việc, nhiệm vụ cụ thể cho từng thành viên.
  - Báo cáo tổng quan các mô hình được sử dụng và chuẩn bị bộ dữ liệu.
- **Mức độ hoàn thành: 100%**
  - Tất cả thành viên được giao công việc cụ thể, rõ ràng để thực hiện.
  - Dự án bước đầu được thực hiện tốt với kế hoạch được đề ra.

Yêu cầu ngày họp 05/12: Huấn luyện, đánh giá các mô hình

- **Yêu cầu**
  - Nhận xét đánh giá và báo cáo các mô hình.
  - Huấn luyện các mô hình và xây dựng ứng dụng.
- **Mức độ hoàn thành: 90-100%**
  - Báo cáo và giới thiệu hoàn thành tốt.
  - Huấn luyện và xây dựng ứng dụng đạt hiệu quả khá tốt.

Yêu cầu ngày họp 12/12: Thiết kế slide, quay video demo

- **Yêu cầu**
  - Quay video demo hoạt động.
  - Thiết kế các phần slide.
- **Mức độ hoàn thiện: 90-100%**
  - Quay video hoàn thành .
  - Slide bước đầu hoàn thiện.

Yêu cầu ngày họp 17/12: Kiểm tra nội dung báo cáo, slide và thuyết trình

- **Yêu cầu**
  - Hoàn thành nội dung slide.
  - Hoàn thành việc xử lý video.
  - Hoàn thành báo cáo.
  - Thuyết trình.
- **Mức độ hoàn thành: 100%**
  - Nội dung của phần báo cáo trên Latex đã hoàn thành.
  - Nội dung Powerpoint đã hoàn thành.
  - Video demo đã xong.
  - Buổi thuyết trình đã diễn ra thành công.

Danh sách hình

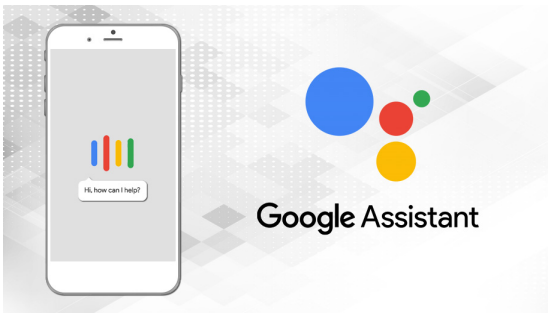
Danh sách bảng

Chương 1

Giới thiệu

Trong thời đại cách mạng công nghiệp 4.0 ngày nay, ứng dụng của trí tuệ nhân tạo *AI* đóng vai quan trọng và mang lại nhiều giá trị hiệu quả trong đa dạng lĩnh vực. Phát hiện cảm xúc từ giọng nói *Emotion Detection from Speech* là một lĩnh vực nghiên cứu đầy triển vọng trong ứng dụng trí tuệ nhân tạo vào đời sống. Khả năng trích xuất trạng thái cảm xúc của người nói từ giọng nói không chỉ cải thiện tương tác giữa con người và máy móc mà còn mở ra nhiều ứng dụng tiềm năng trong các ngành công nghiệp khác nhau. Điển hình là các ứng dụng chăm sóc khách hàng tự động với các trợ lý ảo của các hãng công nghệ lớn như Siri, Google Assitant. Bài toán nhận diện cảm xúc từ giọng nói giúp các hệ thống phản hồi tốt hơn, cá nhân hóa trải nghiệm người dùng, tăng tương tác với khách hàng và cải thiện hiệu quả công việc.

Hình 1.1: Trợ lý ảo Siri của Apple



Hình 1.2: Trợ lý ảo Google Assistant của Google

Mô hình phát hiện cảm xúc từ giọng nói có khả năng phân tích các đặc trưng phổ và tần số của âm thanh. Trong dự án này, hệ thống của chúng ta sẽ phân loại 5 cảm xúc cơ bản: tức giận, sợ hãi, hạnh phúc, buồn bã và trung tính.

Giải mã được cảm xúc từ giọng nói có ý nghĩa rất quan trọng trong ứng dụng thực tiễn. Chính vì vậy, mô hình phát hiện cảm xúc từ giọng nói được tích hợp vào các hệ thống tương tác, hỗ trợ con người:

- Trung tâm cuộc gọi tự động: Giúp xác định mức độ hài lòng hoặc không hài lòng của khách hàng để chuyển tiếp cho nhân viên chăm sóc phù hợp

Hình 1.3: Tổng đài điện thoại ảo VFone

- Giáo dục và y tế: Hỗ trợ người tự kỷ hoặc những người khó khăn trong việc nhận diện cảm xúc xã hội, tự động hóa bệnh viện
- Trò chơi và thực tế ảo: Tăng cường tính tương tác giữa người chơi và các nhân vật ảo

Mặc dù đã có nhiều tiến bộ trong lĩnh vực nhận diện cảm xúc từ giọng nói, bài toán này vẫn tiềm ẩn nhiều thách thức phức tạp, đặc biệt khi ứng dụng vào các tình huống thực tế. Một trong những khó khăn lớn nhất là tính không đồng nhất của dữ liệu. Cảm xúc con người không chỉ phụ thuộc vào nội dung lời nói mà còn thay đổi theo ngữ cảnh, độ tuổi, giới tính và văn hóa của người nói. Ngoài ra, việc phân biệt chính xác giữa các cảm xúc tương tự đòi hỏi các đặc trưng phải đủ chi tiết và mô hình phải đủ nhạy bén để nhận ra những khác biệt nhỏ, điều này không hề dễ dàng trong thực tế. Thêm vào đó, một vấn đề phổ biến khác là tính thực tế của dữ liệu. Nhiều dataset cảm xúc được sử dụng hiện nay chủ yếu dựa trên các đoạn thoại được diễn xuất. Chúng không phản ánh đầy đủ sự phức tạp và tính tự nhiên của cảm xúc con người trong các tình huống thực tế.

Chương 2

Phương pháp nghiên cứu

2.1 Giải thích phần xử lý dữ liệu

Trong bài toán phân loại cảm xúc, một trong những bước quan trọng đầu tiên là xử lý dữ liệu âm thanh. Quá trình này bao gồm việc trích xuất các đặc trưng âm thanh và gán nhãn cho mỗi tệp âm thanh. Các đặc trưng âm thanh quan trọng sẽ được trích xuất từ tín hiệu âm thanh thô, giúp chuyển đổi dữ liệu thành dạng mà các mô hình có thể học và đưa ra dự đoán. Đồng thời, mỗi tệp âm thanh sẽ được gắn nhãn cảm xúc tương ứng kết hợp với các đặc trưng đã trích xuất, tạo thành tập dữ liệu phục vụ cho quá trình huấn luyện cho mô hình.

2.1.1 Trích xuất đặc trưng

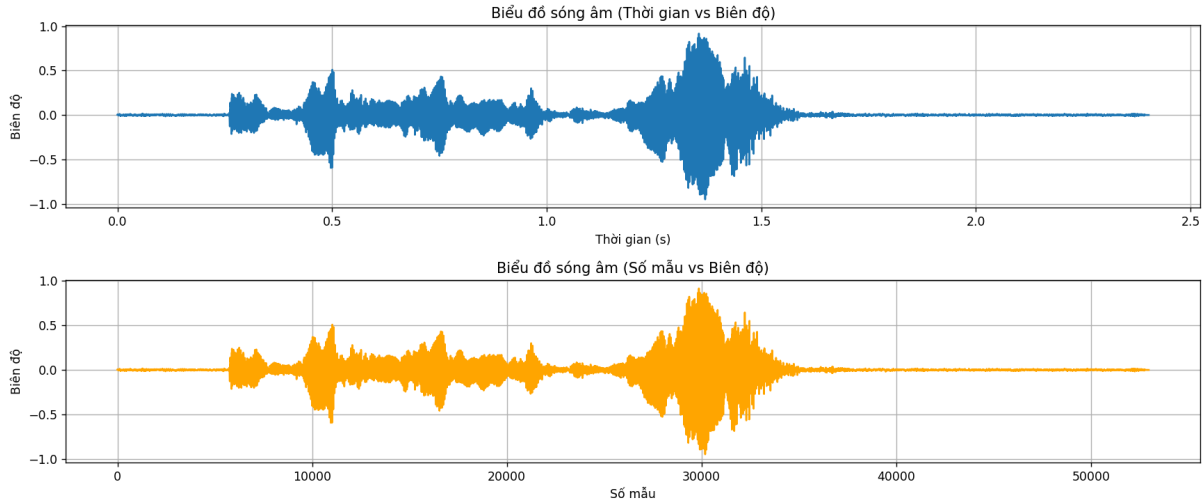
Với sự hỗ trợ mạnh mẽ từ thư viện *Librosa*, một công cụ lý tưởng trong việc xử lý và trích xuất thông tin âm nhạc, chúng ta có thể dễ dàng trích xuất các đặc trưng quan trọng từ các tệp âm thanh. Quá trình này được thực hiện thông qua hàm `extract_feature` dưới đây:

```
def extract_feature(file_path):
    X, sample_rate = librosa.load(file_name)
    stft = np.abs(librosa.stft(X))
    mfccs = np.mean(librosa.feature.mfcc(y=X, sr=sample_rate, n_mfcc=40).T,
    axis=0)
    chroma = np.mean(librosa.feature.chroma_stft(S=stft, sr=sample_rate).T,
    axis=0)
    mel = np.mean(librosa.feature.melspectrogram(y=X, sr=sample_rate).T,
    axis=0)
    contrast = np.mean(librosa.feature.spectral_contrast(S=stft, sr=
    sample_rate).T, axis=0)
    tonnetz = np.mean(librosa.feature.tonnetz(y=librosa.effects.harmonic(X)
    ,sr=sample_rate).T, axis=0)
    return mfccs, chroma, mel, contrast, tonnetz
```

Hàm này nhận vào một đường dẫn tệp âm thanh `file_name` và trả về các đặc trưng âm thanh như MFCCs, Chroma, Mel Spectrogram, Spectral Contrast và Tonnetz. Các đặc trưng này được sử dụng để phân tích tín hiệu âm thanh và phục vụ cho quá trình huấn luyện và đưa ra dự đoán của mô hình.

Đầu tiên, chương trình sẽ đọc tệp âm thanh và chuyển đổi dữ liệu âm thanh thành một mảng số một chiều `X`. Mỗi giá trị trong mảng này đại diện cho biên độ của tín hiệu âm thanh tại một thời điểm cụ thể, phản ánh mức độ dao động của sóng âm tại thời điểm đó. Cùng với mảng tín hiệu, tốc độ lấy mẫu `sample_rate` cũng được lưu lại. Đây là thông số quan trọng, chỉ ra số lượng mẫu âm thanh được lấy trong một giây.

Dưới đây là một hình ảnh minh họa về sự thay đổi biên độ âm thanh theo thời gian và theo số lượng mẫu.

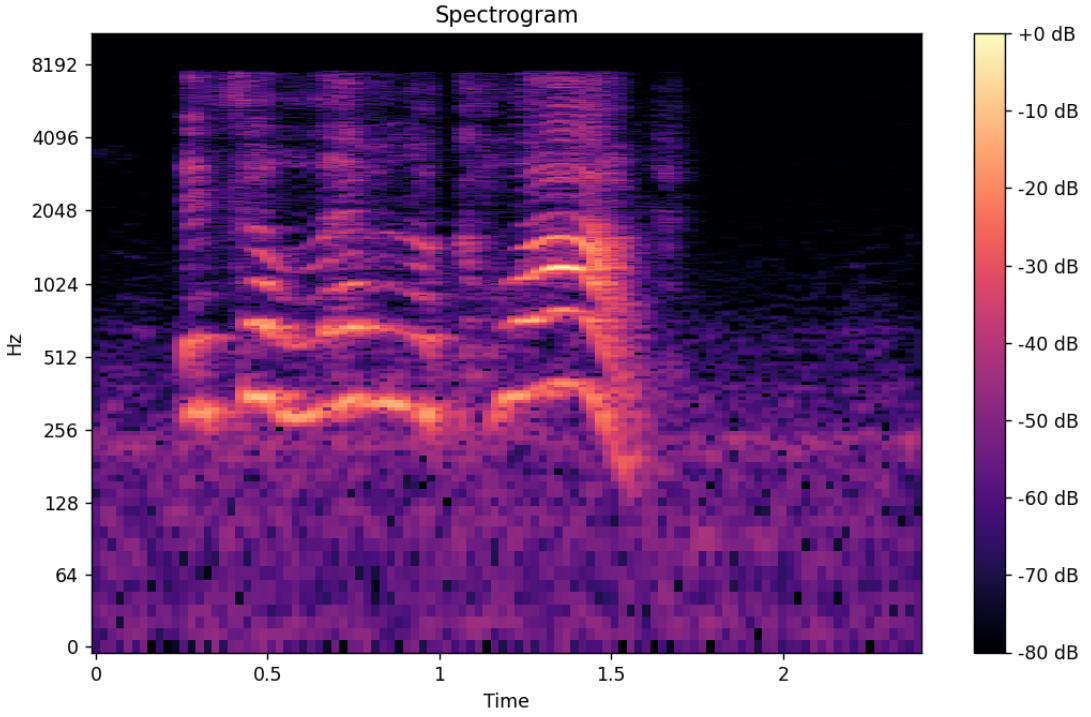


Hình 2.1: Biểu đồ biên độ âm theo thời gian và số mẫu

Tiếp đó, chúng ta sử dụng phương pháp STFT *Short-Time Fourier Transform* để phân tích tín hiệu âm thanh thành các thành phần tần số theo thời gian. STFT chia tín hiệu âm thanh thành các đoạn ngắn và tính toán phổ tần số của từng đoạn đó. Việc này giúp chúng ta biết được trong một khoảng thời gian ngắn, tín hiệu âm thanh có những tần số nào, và những tần số này thay đổi như thế nào.

Để dễ dàng hình dung và phân tích sự thay đổi cường độ tín hiệu theo thời gian và tần số, chúng ta có thể chuyển các giá trị trong ma trận STFT

sang đơn vị decibel (dB). Khi đó, chúng ta có thể vẽ ra một spectrogram - một đồ thị 2D được sử dụng để trực quan hóa sự phân bố năng lượng tín hiệu theo thời gian và tần số.



Hình 2.2: Phổ tần số theo thời gian

Từ những dữ liệu đã tính toán ở trên, chúng ta trích xuất những đặc trưng âm thanh cần thiết cho bài toán phân loại cảm xúc:

- MFCC (Mel-frequency cepstral coefficients): là một trong những đặc trưng phổ biến nhất trong lĩnh vực xử lý tín hiệu âm thanh. Nó giúp trích xuất các đặc điểm âm sắc của âm thanh, dựa trên cách mà con người cảm nhận các tần số qua thang Mel, mô phỏng khả năng phân biệt âm thanh của tai người.

- Chroma: là đặc trưng liên quan đến chất lượng âm nhạc của tín hiệu âm thanh, đặc biệt là trong các ứng dụng phân loại nhạc và phân tích hợp âm. Chroma phân tích các tần số của các nốt nhạc cơ bản và cách chúng thay đổi theo thời gian.

- Mel Spectrogram: là một dạng biểu diễn phổ tần số của tín hiệu âm thanh, nhưng sử dụng thang tần số Mel, giúp mô phỏng cách mà tai người cảm nhận các tần số. Đây là đặc trưng quan trọng trong việc phân tích âm thanh và nhận diện các tính chất âm thanh như giọng nói và nhạc.

- Spectral Contrast: mô tả sự khác biệt giữa các dải tần số trong phổ âm thanh. Các âm thanh có spectral contrast cao thường có âm sắc mạnh mẽ, rõ ràng, trong khi âm thanh có spectral contrast thấp sẽ có âm sắc mờ nhạt hoặc đồng nhất hơn. Đây là đặc trưng hữu ích trong nhận dạng âm thanh và phân loại cảm xúc.

- Tonnetz: là đặc trưng liên quan đến hòa âm và cảm nhận âm nhạc của tín hiệu âm thanh. Tonnetz giúp mô tả các yếu tố như sự chuyển động của các hợp âm và cảm giác tổng thể của âm thanh. Đây là công cụ hữu ích trong phân tích nhạc và nhận diện cảm xúc trong âm thanh.

2.1.2 Trích xuất nhãn dán

Trích xuất nhãn dán là một phần không thể thiếu trong quá trình xây dựng bộ dữ liệu phục vụ cho việc huấn luyện mô hình. Trong bài toán phân loại cảm xúc này, nhãn dán của chúng ta chính là tên các lớp cảm xúc và chúng được lưu trong tên các tệp âm tương ứng. Tên của mỗi tệp âm thanh sẽ có dạng:

<Mã số>_<label>.wav

Để trích xuất nhãn dán, chúng ta sẽ sử dụng hàm `extract_labels` nhận vào tên tệp âm thanh và trả về nhãn cảm xúc tương ứng với tệp đó.

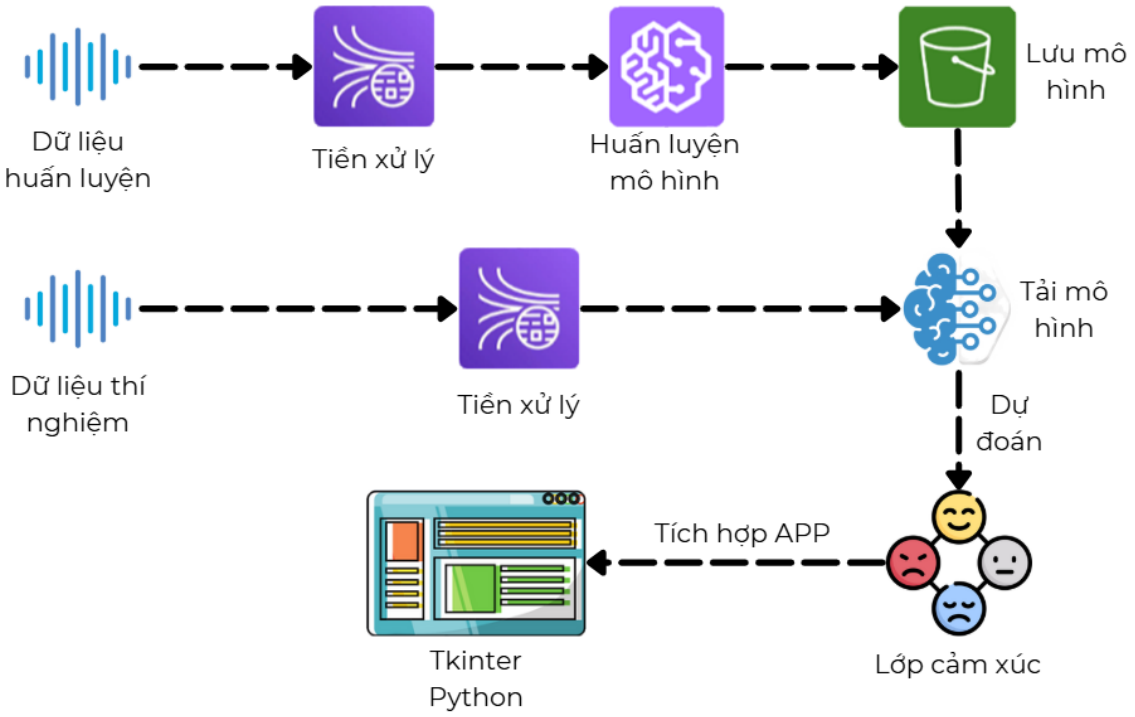
```
def extract_labels(file_path):
    label = file_name.split("_")[-1].split(".")[0]
    return label
```

Sau khi tách tên tệp âm thanh bằng ký tự dấu gạch dưới (_), phần cuối cùng của chuỗi sẽ chứa cả nhãn cảm xúc và phần mở rộng tệp. Để trích xuất nhãn cảm xúc, ta tiếp tục tách chuỗi này bằng dấu chấm (.) và chọn phần đầu tiên, đó chính là nhãn cảm xúc.

2.2 Giới thiệu mô hình

2.2.1 Tổng quan kiến trúc mô hình

Trước tiên, chúng ta phải xem xét kiến trúc của toàn bộ mô hình. Sau đây là sơ đồ thể hiện từng bước liên quan đến quá trình phát hiện cảm xúc từ lời nói:



Hình 2.3: Cấu trúc tổng quản của mô hình

2.2.2 Tổng quan các phương pháp

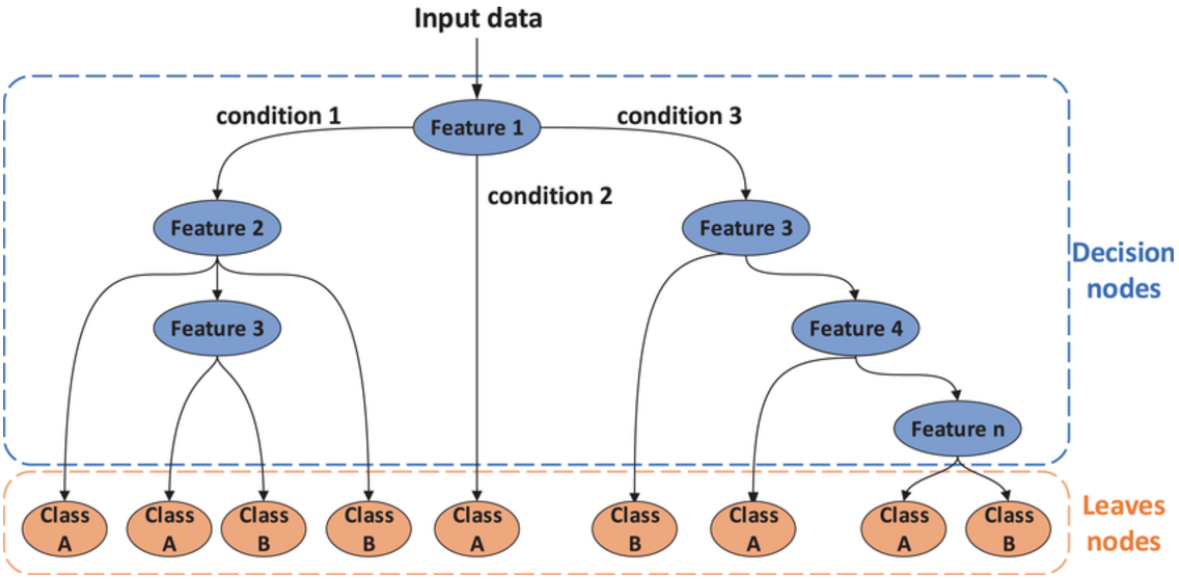
Việc trích xuất đặc trưng và tiền xử lý dữ liệu là những bước quan trọng không thể thiếu trong một dự án AI, tuy nhiên, thuật toán phân loại cũng đóng vai trò không kém phần quan trọng. Thuật toán phân loại giúp quyết định mẫu giọng nói nào thuộc về lớp cảm xúc nào. Chúng ta sẽ sử dụng bốn phương pháp phân loại cho dự án này: Decision Tree, Extra Trees, K-Nearest Neighbors (KNN) và Mạng Nơ-ron (NN).

Decision Tree Classifier

Cây quyết định *Decision Tree* là một thuật toán học máy có giám sát không tham số, được sử dụng phổ biến trong các bài toán phân loại và hồi quy. Mục tiêu của thuật toán này là tạo ra một mô hình dự đoán giá trị của biến mục tiêu dựa trên các quy tắc quyết định đơn giản được rút ra từ các đặc trưng của dữ liệu. Sau đây, chúng ta sẽ tìm hiểu rõ hơn về thuật toán này:

1. Cấu trúc cơ bản

    - Nút gốc (Root Node) : Là điểm khởi đầu của cây. Tại đây, dữ liệu sẽ được chia dựa trên một điều kiện quyết định đầu tiên.
    - Nút quyết định (Decision Nodes) : Các nút này chứa các điều kiện phân chia dữ liệu. Mỗi nút sẽ dựa trên giá trị của một hoặc nhiều đặc trưng để chia dữ liệu thành các nhánh.
    - Nút lá (Leaf Nodes) : Là các nút cuối cùng của cây. Mỗi nút lá đại diện cho một kết quả dự đoán, trong bài toán này là các lớp cảm xúc.



Hình 2.4: Cấu trúc cơ bản của cây quyết định

2. Nguyên tắc hoạt động

   Cây quyết định hoạt động dựa trên việc phân chia dữ liệu theo các điều kiện nhằm tối đa hóa tính đồng nhất của các nhóm được tạo ra. Quy trình này bao gồm các bước sau:

   - Chọn điều kiện phân chia : Tại mỗi nút, cây quyết định tìm một đặc trưng và một ngưỡng giá trị tối ưu để chia dữ liệu.
   - Phân tách dữ liệu : Dữ liệu được chia thành các nhánh dựa trên điều kiện được chọn.
   - Lặp lại quá trình : Quá trình chọn điều kiện và phân tách tiếp tục tại mỗi nút bên trong. Điều này diễn ra cho đến khi đạt một trong các tiêu chí dừng như:
     - Không còn đặc trưng nào để chia tiếp : Khi cây đã sử dụng hết tất cả các đặc trưng có sẵn trong dữ liệu.
     - Mức độ không đồng nhất đạt ngưỡng nhỏ nhất: Khi các mẫu trong một nút trở nên đồng nhất tất cả cùng một lớp.
     - Cây đạt đến độ sâu tối đa cho phép: Khi cây đã đạt độ sâu tối đa mà chúng ta đã đặt trước, quá trình chia nhánh sẽ dừng lại.
   - Kết quả : Khi không thể phân chia thêm, nút đó trở thành nút lá. Nút lá đưa ra dự đoán, thường là nhãn của nhóm dữ liệu chiếm đa số trong nhánh đó.

3. Ưu điểm

   - Dễ hiểu, dễ diễn giải : Cây quyết định có cấu trúc rõ ràng, dễ dàng giải thích cách đưa ra dự đoán.
   - Làm việc tốt với dữ liệu phi tuyến tính : Cây quyết định có thể mô hình hóa các mối quan hệ phi tuyến tính mà không cần biến đổi phức tạp.
   - Tính linh hoạt : Có thể áp dụng cho cả bài toán phân loại và hồi quy với hiệu suất khá tốt. Cây quyết định thường có tốc độ huấn luyện nhanh và dễ triển khai trên dữ liệu không quá lớn.

4. Nhược điểm

    - Quá khớp (overfiting) : Cây quyết định dễ dàng bị quá khớp nếu không giới hạn độ sâu của cây hoặc số lượng mẫu tối thiểu tại một nút.
    - Kém hiệu quả với dữ liệu phức tạp : Với các tập dữ liệu lớn và phức tạp, cây quyết định có thể trở nên kém hiệu quả và không ổn định.
    - Độ nhạy cao với nhiễu : Các thay đổi nhỏ trong dữ liệu có thể dẫn đến sự thay đổi đáng kể trong cấu trúc cây.

Extra Trees Classifier

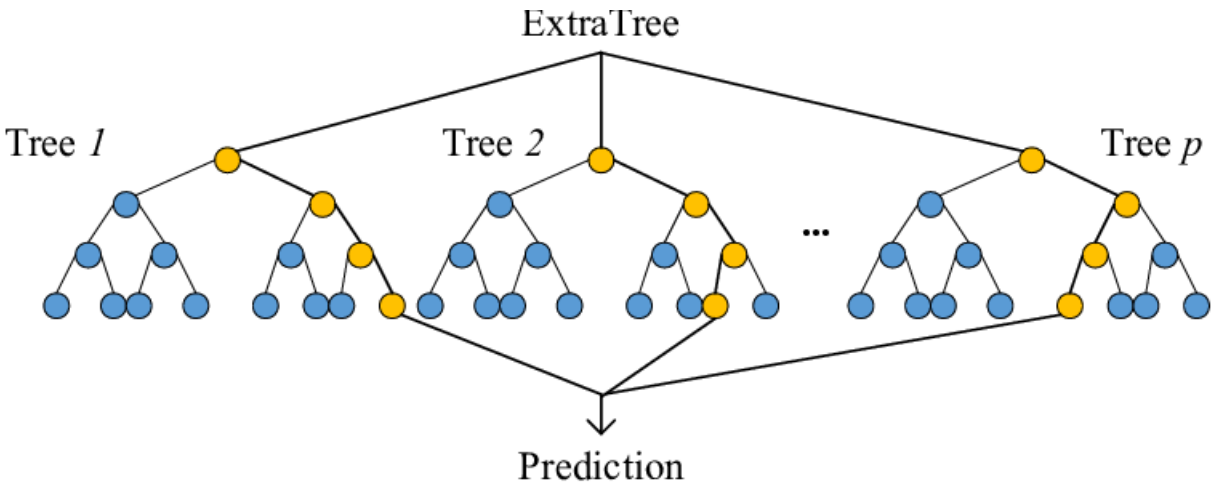
Thuật toán Extra Trees (viết tắt của Extremely Randomized Trees) là một phương pháp ensemble thuộc họ các thuật toán cây quyết định, được đề xuất bởi Geurts và cộng sự. Khác với các phương pháp như Random Forest, điểm đặc trưng của Extra Trees nằm ở mức độ ngẫu nhiên hóa cao trong quá trình xây dựng cây.

Cụ thể, thuật toán này thực hiện ngẫu nhiên hóa ở hai cấp độ: thuộc tính được chọn ngẫu nhiên thay vì dựa trên tiêu chí tối ưu, và điểm chia trên các thuộc tính được quyết định hoàn toàn ngẫu nhiên thay vì tính toán tối ưu như thông thường.

Chính sự ngẫu nhiên hóa này không chỉ giảm thiểu hiện tượng overfitting mà còn cải thiện khả năng khái quát hóa của mô hình, đồng thời giảm chi phí tính toán do không cần sắp xếp dữ liệu để tìm kiếm điểm chia tốt nhất. Dưới đây là một số đặc điểm của thuật toán này:

1. Cấu trúc cơ bản

    ExtraTree Classifier bao gồm một tập hợp các cây quyết định độc lập. Mỗi cây trong ExtraTree có cấu trúc cơ bản tương tự cây quyết định thông thường, nhưng quá trình xây dựng cây được thêm tính ngẫu nhiên hóa.



Hình 2.5: Cấu trúc cơ bản của ExtraTree

2. Nguyên tắc hoạt động

   Quá trình xây dựng cây trong ExtraTree khác biệt với các cây quyết định truyền thống ở một số điểm sau:

   - Chọn ngẫu nhiên đặc trưng : Thuật toán này không sử dụng toàn bộ các đặc trưng có sẵn mà thay vào đó chọn ngẫu nhiên một tập con các đặc trưng. Số lượng đặc trưng này có thể được điều chỉnh bằng tham số `max_features`.
   - Chọn ngẫu nhiêu điểm chia : Với mỗi đặc trưng được chọn thuật toán sẽ xác định một khoảng giá trị đặc trưng và chọn một điểm chia ngẫu nhiên trong khoảng giá trị này.
   - Phân chia dữ liệu : Dữ liệu được phân chia thành hai nhánh dựa trên điểm chia ngẫu nhiên đã chọn ở bước trước. Quá trình này sẽ tiếp tục được lặp lại tại mỗi nút của cây, và cây sẽ phân chia dữ liệu cho đến khi đạt được các điều kiện dừng dưới đây:
     - Độ sâu tối đa của cây (Max Depth): Xác định số mức tối đa từ gốc đến lá. Khi cây đạt độ sâu này, quá trình chia nhánh dừng lại, giúp ngăn cây quá phức tạp và giảm overfitting.
     - Số mẫu tối thiểu tại một nút (Min Samples per Node): Là số mẫu tối thiểu để một nút được xem xét chia nhánh. Nếu không đủ, nút sẽ trở thành nút lá, giúp giảm nhiễu, tránh overfitting và cải thiện khả năng khái quát hóa.

- Số mẫu tối thiểu để chia (Min Samples for Split): Là số mẫu tối thiểu cần thiết để thực hiện phép chia. Nếu không đủ, nút trở thành nút lá. Điều này giúp giảm chia nhánh không cần thiết, tối ưu hóa hiệu suất và tăng độ chính xác của mô hình.

- Tổng hợp kết quả : Mỗi cây trong mô hình sẽ đưa ra một dự đoán cho một mẫu dữ liệu. Kết quả cuối cùng sẽ được tính bằng cách kết hợp tất cả các dự đoán từ các cây, thường là bằng cách bầu chọn (cho bài toán phân loại) hoặc tính trung bình (cho bài toán hồi quy).

3. Ưu điểm:

- Khả năng tổng quát tốt : Extra Trees tạo ra nhiều cây quyết định với cách chia nhánh ngẫu nhiên, giúp giảm thiểu overfitting.
- Tốc độ huấn luyện nhanh : Do việc chia nhánh được thực hiện ngẫu nhiên nên không phải tìm kiếm tối ưu cho mỗi đặc trưng.

4. Nhược điểm :

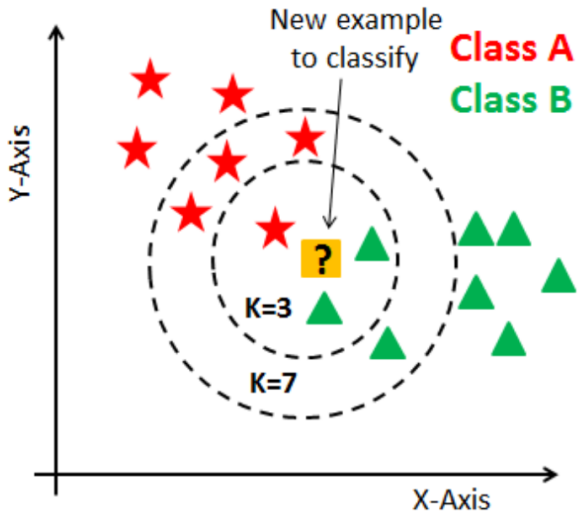
- Yêu cầu bộ nhớ lớn : Do Extra Trees tạo ra nhiều cây quyết định, mô hình này có thể yêu cầu một lượng bộ nhớ đáng kể, đặc biệt với dữ liệu lớn hoặc số lượng cây lớn.
- Hiệu suất không ổn định trên các dữ liệu nhỏ : Extra Trees có thể không hoạt động tốt trên dữ liệu nhỏ vì sự ngẫu nhiên trong việc chia nhánh có thể khiến mô hình khó đạt được độ chính xác cao.

K-Nearest Neighbors

KNN (K-Nearest Neighbors) là một thuật toán học máy thuộc nhóm học không tham số, cũng giống với các thuật toán cây quyết định, KNN được dùng để phân loại hoặc hồi quy. Trong trường hợp phân loại, thuật toán dựa trên nguyên tắc tìm K điểm lân cận gần nhất để quyết định nhãn của đối tượng cần dự đoán.

1. Cấu trúc cơ bản

Thuật toán KNN cho rằng các dữ liệu tương tự nhau là các điểm được đặt gần nhau trong một không gian. Từ đó, để dự đoán nhãn hoặc giá trị của một điểm mới, ta sẽ tìm kiếm K điểm lân cận gần nhất (hàng xóm) so với điểm mà ta muốn tìm. Cuối cùng đưa ra dự đoán dựa trên nhãn hoặc giá trị của những điểm lân cận đó.



Hình 2.6: Cấu trúc cơ bản của thuật toán KNN

2. Nguyên tắc hoạt động

Như đã nói ở trên, KNN hoạt động dựa trên việc tìm các điểm lân cận để đưa ra dự đoán. Quy trình này bao gồm các bước:

- Chọn tham số K phù hợp với bài toán : Ta có thể thử nghiệm nhiều lần với tham số K khác nhau để chọn ra kết quả tối ưu nhất cho bài toán.
- Tính khoảng cách giữa mẫu cần dự đoán và tất cả các mẫu trong tập huấn luyện : Có nhiều công thức phổ biến để đo khoảng cách, chẳng hạn như khoảng cách Euclide, khoảng cách Manhattan, khoảng cách Cosine,... Ở đây chúng ta sẽ đề cập đến khoảng cách Euclide:

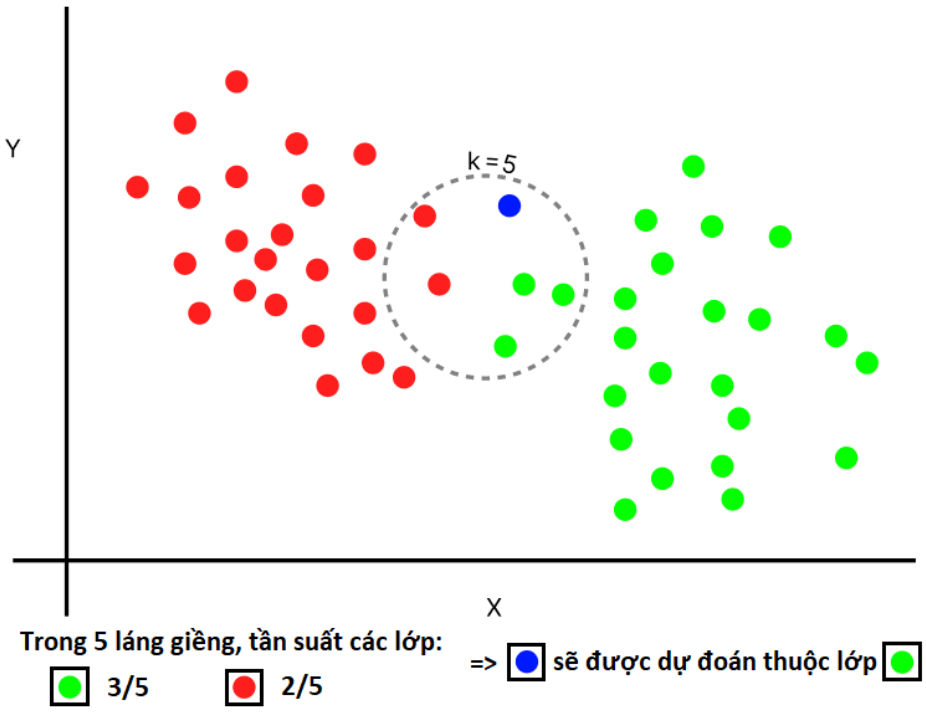
$$d = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$$

Trong đó:

- $x = (x_1, x_2, \ldots, x_n)$: Vector tọa độ của mẫu cần dự đoán.
- $y = (y_1, y_2, \ldots, y_n)$: Vector tọa độ của một mẫu trong tập huấn luyện.

Sử dụng thuật toán để tính khoảng cách từ mẫu dữ liệu mới cần dự đoán đến tất cả các mẫu trong tập huấn luyện.

- Xác định K láng giềng gần nhất : Từ kết quả ở trên, chọn ra K mẫu có khoảng cách nhỏ nhất.
- Dự đoán nhãn cho mẫu mới : Xác định tần suất các nhãn xuất hiện trong K láng giềng vừa được chọn. Cuối cùng lấy nhãn xuất hiện nhiều nhất trong K láng giềng để làm nhãn dự đoán.



Hình 2.7: Minh họa nguyên tắc hoạt động của thuật toán KNN

3. Ưu điểm

- Đơn giản, dễ hiểu và dễ triển khai : Không yêu cầu xây dựng mô hình phức tạp.
- Không tham số : Không cần giả định trước về phân phối của dữ liệu.

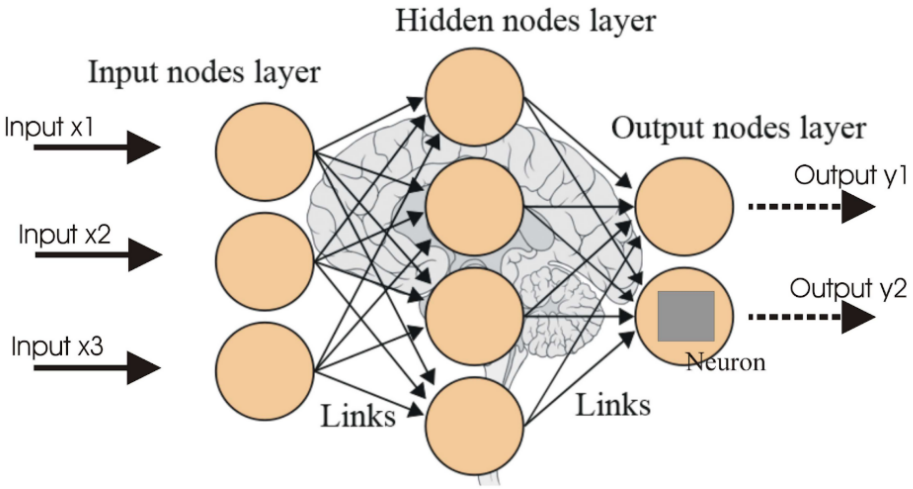
- Hiệu quả với dữ liệu nhỏ hoặc trung bình : Hoạt động tốt khi tập dữ liệu có kích thước nhỏ và số chiều không quá cao.

4. Nhược điểm

   - Tính toán tốn kém : Khi tập dữ liệu lớn, việc tính toán khoảng cách cho từng mẫu mất nhiều thời gian.
   - Nhạy cảm với dữ liệu nhiễu và các đặc trưng không quan trọng. Các mẫu nhiễu hoặc phân bố không đồng đều có thể gây ảnh hưởng lớn đến kết quả.
   - Phụ thuộc nhiều vào cách lựa chọn K : Phải chọn tham số K sao cho phù hợp. K quá nhỏ sẽ nhạy cảm với nhiễu, K quá lớn sẽ làm mất đi tính chính xác.

Neural Network

Neural Network (NN), lấy cảm hứng từ cơ chế hoạt động của não bộ con người, đã trở thành công cụ mạnh mẽ để giải quyết các bài toán phức tạp như nhận diện hình ảnh, xử lý ngôn ngữ tự nhiên và dự đoán dữ liệu. Mạng nơ-ron bao gồm các đơn vị cơ bản là các nơ-ron nhân tạo, được liên kết với nhau thông qua các lớp (layer) để hình thành một mạng lưới thông minh, có khả năng học hỏi và tối ưu hóa từ dữ liệu.



Hình 2.8: Cấu trúc cơ bản của mạng nơ-ron

1. Cấu trúc cơ bản

   Neural Network thường có 3 thành phần chính:

   - Lớp đầu vào (Input Layer): Nhận dữ liệu đầu vào từ môi trường bên ngoài. Số nơ-ron trong lớp này thường tương ứng với số đặc trưng (features) của dữ liệu.
   - Lớp ẩn (Hidden Layers): các lớp nằm giữa lớp đầu vào và lớp đầu ra, giúp xử lý và học các đặc trưng phức tạp từ dữ liệu.

     Các nơ-ron trong lớp ẩn thực hiện các phép biến đổi phi tuyến tính, sử dụng các hàm kích hoạt như ReLU, Sigmoid, hoặc Tanh. Lớp ẩn giúp mạng nơ-ron học các mối quan hệ phức tạp và tối ưu hóa kết quả đầu ra.
   - Lớp đầu ra (Output Layer): Trả về kết quả cuối cùng như dự đoán nhãn phân loại hoặc giá trị liên tục trong bài toán hồi quy.

2. Nguyên tắc hoạt động

   Neural Network học từ dữ liệu thông qua quá trình huấn luyện, bao gồm các bước:

   - Khởi tạo trọng số và độ lệch: Trọng số và độ lệch của các kết nối giữa các nơ-ron được khởi tạo ngẫu nhiên trước khi bắt đầu huấn luyện.
   - Lan truyền dữ liệu: Dữ liệu đầu vào được đưa qua lớp đầu vào, sau đó qua các lớp ẩn, và cuối cùng là lớp đầu ra để tính toán dự đoán.
   - Tính toán lỗi: Kết quả dự đoán được so sánh với giá trị thực tế, từ đó tính toán sai số bằng một hàm mất mát.
   - Lan truyền ngược: Lỗi được lan truyền ngược qua các lớp, tính toán đạo hàm của lỗi đối với trọng số và độ lệch để xác định cách thức các tham số cần được điều chỉnh.
   - Cập nhật trọng số: Các trọng số được cập nhật qua nhiều epoch (vòng lặp), cải thiện dần hiệu suất mô hình.

Sau khi huấn luyện, mạng nơ-ron có thể sử dụng trọng số và độ lệch đã tối ưu để đưa ra dự đoán cho các dữ liệu mới.

3. Ưu điểm

    - Khả năng học phi tuyến tính: Neural Network có thể học được các mối quan hệ phi tuyến tính phức tạp trong dữ liệu.
    - Khả năng tổng quát hóa tốt: Khi được huấn luyện đúng cách, Neural Network có thể tạo ra các dự đoán chính xác trên dữ liệu chưa từng thấy.
    - Tính linh hoạt: Neural Network có thể áp dụng cho nhiều loại dữ liệu khác nhau, từ dữ liệu dạng chuỗi, hình ảnh đến âm thanh. Được tích hợp với các công cụ mạnh mẽ như GPU và TPU để huấn luyện nhanh hơn.

4. Nhược điểm

    - Cần nhiều dữ liệu: Neural Network yêu cầu lượng lớn dữ liệu để đạt hiệu quả tốt, đặc biệt với các mô hình sâu.
    - Tính toán phức tạp: Huấn luyện và dự đoán đòi hỏi tài nguyên tính toán lớn, tiêu tốn thời gian và năng lượng.
    - Khó giải thích: Neural Network hoạt động như một "hộp đen", khó hiểu được các quyết định bên trong.

2.3 Giải thích mô hình

2.3.1 Tiền xử lý

Trong quá trình huấn luyện

Để huấn luyện mô hình nhận diện cảm xúc từ giọng nói, chúng ta cần chuẩn bị một bộ dữ liệu bao gồm các đặc trưng của âm thành và nhãn tương ứng. Chúng ta sử dụng hàm `parse_audio_files`, giúp xử lý tất cả các tệp âm thanh trong một thư mục nhất định và trả về hai mảng: một mảng chứa các đặc trưng của các tệp âm và một mảng chứa nhãn tương ứng.

```python
def parse_audio_files(path):
    features_list = []
    labels_list = []
    for fn in glob.glob(path  + "/*.wav"):
        mfccs, chroma, mel, contrast, tonnetz = extract_feature(fn)
        ext_features = np.hstack([mfccs, chroma, mel, contrast, tonnetz])
        features_list.append(ext_features)
        label = extract_labels(fn)
        labels_list.append(label)
    features = np.array(features_list)
    labels = np.array(labels_list)
    return features, labels
```

Hàm sẽ duyệt qua tất cả các tệp âm thanh với đuôi `.wav` trong thư mục được chỉ định thông qua đường dẫn path. Đối với mỗi tệp âm thanh, các đặc trưng sẽ được trích xuất thông qua hàm `extract_feature`, sau đó các đặc trưng này sẽ được kết hợp lại thành một vecto (mảng 1D) và lưu vào danh sách `features_list`. Đồng thời, nhãn của mỗi tệp âm thanh sẽ được xác định bằng cách gọi hàm `extract_labels`, và nhãn này sẽ được thêm vào danh sách `labels_list`.

Hai danh sách này sau đó sẽ được chuyển đổi thành các mảng NumPy và trả về dưới tên là `features` và `labels`.

- `features`: là một mảng 2D có kích thước (N, 193), trong đó N là số lượng mẫu và 193 là số lượng đặc trưng được trích xuất từ mỗi tệp âm thanh.
- `labels`: là một mảng 1D có độ dài N, chứa nhãn tương ứng với mỗi tệp âm thanh trong bộ dữ liệu.

Sau đó, chúng ta xáo trộn ngẫu nhiên các mảng `features` và `labels` sao cho các phần tử trong hai mảng này vẫn giữ đúng mối quan hệ tương ứng.

```python
def shuffle_data(features, labels):
    indices = np.random.permutation(len(features))
    shuffled_features = features[indices]
    shuffled_labels = labels[indices]
    return shuffled_features, shuffled_labels
```

Việc xáo trộn dữ liệu giúp mô hình tránh học thuộc vào trật tự của dữ liệu, đảm bảo rằng mô hình học được từ một tập hợp mẫu ngẫu nhiên và

đa dạng. Những dữ liệu này được vào các file `.npy` để tiện cho việc tái sử dụng:

```
np.save('features.npy', features)
np.save('labels.npy', labels)
```

Trong quá trình huấn luyện mô hình, khi cần sử dụng những dữ liệu này chúng ta sẽ sử dụng lệnh `np.load()` để tải dữ liệu từ các file đã lưu ở trên:

```
features = np.load("features.npy", allow_pickle=True)
labels = np.load("labels.npy", allow_pickle=True)
```

Đối với các mô hình học máy truyền thống, chỉ cần tải dữ liệu từ file và đưa vào mô hình. Tuy nhiên, đối với các mạng nơ-ron (Neural Networks), chúng ta cần chuẩn hoá để đảm bảo rằng dữ liệu đầu vào có thể được sử dụng đúng cách trong quá trình huấn luyện.

```
X = np.array(features, dtype=np.float32)
Y = np.array(labels, dtype=str)
encoder = LabelEncoder()
encoder.fit(Y)
encoded_Y = encoder.transform(Y)
dummy_y = to_categorical(encoded_Y)
```

- Chuyển đổi đặc trưng và nhãn thành dạng mảng NumPy

  Các giá trị đặc trưng được chuyển về kiểu dữ liệu `float32` và các nhãn được chuyển về kiểu `str`.

- Chuẩn hoá nhãn với LabelEncoder

  Khởi tạo một đối tượng `LabelEncoder` để chuẩn hóa nhãn, sau đó đối tượng này sẽ học từ dữ liệu nhãn và chuyển đổi các nhãn văn bản thành dạng số nguyên.

- Chuyển nhãn thành one-hot encoding

  Chuyển đổi nhãn số nguyên thành dạng one-hot encoding, mỗi nhãn sẽ được biểu diễn dưới dạng một vector với một phần tử duy nhất bằng 1 và các phần tử còn lại bằng 0.

Trong quá trình dự đoán

Trong dự án này, các mô hình sẽ được huấn luyện trên những đoạn âm thanh ngắn. Tuy nhiên khi trải khai mô hình, chúng ta sẽ dự đoán trên những đoạn âm thanh dài hơn. Nếu không chia nhỏ tệp âm thanh, kết quả dự đoán sẽ không thực sự chính xác. Để giải quyết vấn đề này, chúng ta chia các tệp âm thanh dài thành những đoạn ngắn hơn, phù hợp với khả năng của mô hình. Quá trình trên được thực hiện như sau:

1. Xác định thời lượng của tệp âm thanh:

```
y, sr = librosa.load(path)
secs = librosa.get_duration(y=y, sr=sr)
```

   Trong đó:

   - *path* : Đường dẫn đến tệp âm thanh cần được xử lý.
   - *sr* : Tần số lấy mẫu của tệp âm thanh, biểu thị số mẫu được lấy trong 1 giây.
   - *y* : Mảng Numpy chứa dữ liệu của âm thanh.
   - *secs* : Thời lượng của đoạn âm thanh.

2. Chia nhỏ âm thanh nếu thời lượng quá lớn:

   Ta sẽ chia nhỏ âm thanh thành các đoạn có thời lượng khoảng 2 giây chồng lấp lên nhau.

```
if secs <= 1:
    pass
else:
    for i in range(int((secs-1)//0.5)+3):
        cat = y[max(0, int(sr*(i*0.5-1))):min(int((i*0.5+1)*sr), int(
    secs*sr))]
```

   Đoạn mã này sử dụng biến i để chia tệp âm thanh thành các đoạn ngắn, với độ chồng lấp từ 1 đến 1.5 giây giữa các đoạn. Biến i chạy qua các giá trị từ 0 đến số lượng đoạn cần tạo, mỗi lần lặp tính toán chỉ số mẫu tương ứng dựa trên tần số mẫu sr, sau đó cắt ra các đoạn âm thanh xung quanh vị trí đó. Các đoạn dữ liệu này sẽ được trích xuất các đặc trưng tương tự như ở mục 2.1.1.

2.3.2 Huấn luyện và lưu mô hình

Sau khi xử lý dữ liệu về định dạng phù hợp, chúng ta bắt đầu khởi tạo và huấn luyện các mô hình:

Decision Tree Classifier

```
clf = tree.DecisionTreeClassifier(class_weight="balanced",criterion='
    entropy', max_depth=30, random_state=42, min_samples_split=30,
    min_samples_leaf=20)
```

Chúng ta khởi tạo mô hình với các thông số như:

- `class_weight="balanced"` : Tham số này giúp cây quyết định tự động điều chỉnh trọng số của các lớp khi có sự mất cân bằng lớp.
- `criterion='entropy'` : Quyết định tiêu chí được sử dụng để chia các nút trong cây, `entropy` dùng để đánh giá mức độ hỗn loạn hoặc không chắc chắn trong dữ liệu.
- `max_depth=30` : Giới hạn độ sâu tối đa của cây quyết định.
- `min_samples_split=30` : Quy định số lượng mẫu tối thiểu phải có tại một nút để tiếp tục phân chia.
- `min_samples_leaf=20` : Đảm bảo rằng mỗi lá trong cây chứa ít nhất 20 mẫu.
- `random_state=42` : cho phép mô hình tái tạo được kết quả khi chạy lại với cùng một dữ liệu. Đây là một cách để đảm bảo tính nhất quán trong các lần chạy mô hình, giúp người dùng kiểm tra kết quả một cách lặp lại và ổn định.

Extra Tree Classifier

```
exported_pipeline = ExtraTreesClassifier(bootstrap=True,criterion="entropy"
    ,max_features=20,min_samples_leaf=5,min_samples_split=8,max_depth=12,
    n_estimators=1000)
exported_pipeline.fit(tr_features, tr_labels)
```

Mô hình được khởi tạo với các thông số quan trọng như:

- `max_features=20` : Số lượng đặc trưng tối đa được sử dụng tại mỗi phân nhánh của cây.
- `min_samples_leaf=5` : Số lượng mẫu tối thiểu tại mỗi lá của cây. Điều này giúp tránh việc mô hình học quá chi tiết.
- `min_samples_split=8` : Số lượng mẫu tối thiểu trong một node để có thể chia thành các node con.
- `n_estimators=1000` : Số lượng cây quyết định trong mô hình.

K-Nearest Neighbors

```
neigh = KNeighborsClassifier(algorithm='auto', n_neighbors=7,p=2,weights='
    uniform')
```

Mô hình được khởi tạo với các thông số quan trọng như:

- `algorithm='auto'` : Thuật toán được sử dụng để tính toán các k-neighbors gần nhất, ở đây chúng ta cho phép mô hình tự động chọn thuật toán phù hợp nhất dựa trên số lượng điểm dữ liệu.
- `n_neighbors=7` : Số lượng láng giềng gần nhất mà mô hình sẽ sử dụng để phân loại một điểm dữ liệu.
- `p=2` : Tham số này xác định loại khoảng cách được sử dụng để tính toán khoảng cách giữa các điểm dữ liệu ở đây chúng ta chọn khoảng cách *Euclidean*.
- `weights='uniform'` : Tham số này xác định cách tính trọng số cho các láng giềng trong quá trình phân loại, ở đây chúng ta chọn tất cả các láng giềng đều có trọng số như nhau.

Neural Network

Với sự hỗ trợ của thư viện Keras, chúng ta xây dựng mạng nơ-ron với cấu trúc đơn giản như sau:

```
def baseline_model():
    model = Sequential()
```

```
    model.add(Dense(256, input_dim=193, activation="relu",
    kernel_initializer="he_normal"))
    model.add(Dropout(0.3))
    model.add(Dense(128, activation="relu",kernel_initializer="he_normal"))
    model.add(Dropout(0.3))
    model.add(Dense(64, activation="relu", kernel_initializer="he_normal"))
    model.add(Dropout(0.3))
    model.add(Dense(5, activation="softmax", kernel_initializer="
    glorot_uniform"))
    optimizer = Adam(learning_rate=0.0001)
    model.compile(loss='categorical_crossentropy', optimizer=optimizer,
    metrics=['accuracy'])
    return model
```

- Đầu tiên chúng ta khởi tạo một mô hình mạng nơ-ron theo kiểu tuần tự (sequential), nghĩa là các lớp sẽ được xếp chồng lên nhau từ đầu đến cuối.
- Các lớp Dense với số nơ-ron lần lượt là 256, 128, 64 và lớp cuối cùng là 5 tương ứng với 5 lớp cảm xúc.
- Xen giữa các là các lớp Dropout để giảm overfitting, tăng khả năng tổng quát trong quá trình huấn luyện mô hình.
- Cuối cùng, mô hình được biên dịch với hàm mất mát là categorical_crossentropy phù hợp cho bài toán phân loại đa lớp và tối ưu hoá bằng Adam với tỷ lệ học là 0.0001 đảm bảo tính ổn định.

Mô hình sau khi được khởi tạo sẽ nhận dữ liệu và huấn luyện với các thông số dưới đây:

```
result = model.fit(X, dummy_y, validation_split=1/9, batch_size=64, epochs
    =200, verbose=1)
```

- `validation_split=1/9` : chỉ định phần trăm dữ liệu huấn luyện sẽ được tách ra để kiểm tra mô hình trong quá trình huấn luyện.
- `batch_size=64` : mỗi lần huấn luyện sẽ cập nhật trọng số sau khi xử lý 64 mẫu.
- `epochs=200` : mô hình sẽ được huấn luyện 200 lần trên toàn bộ tập dữ liệu huấn luyện.

Sau khi quá trình huấn luyện mô hình hoàn tất, chúng ta sẽ tiến hành lưu lại các mô hình đã được huấn luyện. Các mô hình học máy sẽ được lưu trữ dưới định dạng `.sav`, trong khi đó mô hình học sâu sẽ được lưu dưới định dạng `.h5` theo chuẩn của Keras. Việc lưu mô hình giúp chúng ta dễ dãng quản lý và triển khai sau này.

2.3.3 Phân loại cảm xúc

Từ những mô hình đã được huấn luyện ở trên, ta có thể sử dụng chúng để đưa ra dự đoán cảm xúc từ giọng nói. Đối với các mô hình khác nhau thì quá trình để đưa ra dự đoán cũng khác nhau. Với các mô hình học máy như KNN, Decision Tree và Extra Tree thì mô hình sẽ đưa ra kết quả dự đoán trực tiếp, còn đối với mô hình học sâu như Neural Network thì mô hình sẽ trả về xác suất của các lớp, từ đó ta sẽ chọn ra lớp có xác suất lớn nhất làm kết quả cuối cùng.

Cụ thể, đối với mô hình học máy, ta sẽ sử dụng thư viện *_pickle* để tải mô hình lên và đưa ra dự đoán:

```
def predict(audio_path, modelpath):
    ts_features= np.hstack(extract_feature(audio_path))
    ts_features = np.array(ts_features, dtype=pd.Series).reshape(1,-1)
    model = pickle.load(open(modelpath, 'rb'))
    prediction = model.predict(ts_features)[0]
    return prediction
```

Mô hình học máy của chúng ta nhận vào một ma trận `ts_features`, trong đó số hàng là số lượng mẫu cần dự đoán (ở đây mỗi lần chỉ dự đoán một mẫu), và số cột là số lượng đặc trưng đã được trích xuất từ tệp âm thanh. Sau khi mô hình nhận đầu vào này, kết quả đầu ra `prediction` sẽ là một trong các lớp cảm xúc mà mô hình đã được huấn luyện trước, bao gồm: ANG, HAP, FEA, NEU, và SAD.

Còn đối với mô hình học sâu, viêc tải mô hình sẽ được hỗ trở bởi thư viện *keras*.

```
def predict(audio_path, model_path):
    ts_features= np.hstack(extract_feature(audio_path))
    ts_features = np.array(ts_features, dtype=float).reshape(1,-1)
    model = load_model(model_path)
    labels_map = ("ANG", "FEA", "HAP", "NEU", "SAD")
    prediction = model.predict(ts_features)
```

```
    predicted_class = np.argmax(prediction[0])
    prediction = labels_map[predicted_class]
    return prediction
```

Đầu vào của mô hình này cũng là một ma trận giống như ở trên. Tuy nhiên, khác với các mô hình học máy, mô hình học sâu sẽ trả về một mảng chứa xác suất tương ứng với từng lớp cảm xúc. Dựa trên mảng xác suất này, chúng ta sẽ chọn lớp có xác suất cao nhất và gán nhãn lớp đó làm kết quả dự đoán cuối cùng.

Khi các đoạn âm thanh dài được chia thành các phần nhỏ, mô hình sẽ thực hiện dự đoán cho từng phần và trả về một danh sách các nhãn tương ứng. Từ danh sách này, chúng ta sẽ chọn nhãn có tần suất xuất hiện cao nhất làm kết quả cuối cùng.

```
def max_appear(list_emo):
    dic = {}
    for i in list_emo:
        dic[i] = dic.get(i,0) + 1
    return max(dic.items(), key=lambda a: a[1])[0]
```

Khi có nhiều nhãn có cùng tần suất, chúng ta sẽ ưu tiên chọn nhãn xuất hiện đầu tiên.

2.3.4 Tích hợp ứng dụng

Bước cuối cùng, chúng ta sẽ xây dựng một ứng dụng với chức năng nhận diện cảm xúc từ giọng nói. Chúng ta sẽ sử dụng thư viện Tkinter trong Python, một công cụ mạnh mẽ giúp xây dựng giao diện người dùng (GUI). Giao diện sẽ bao gồm những tính năng chính sau:

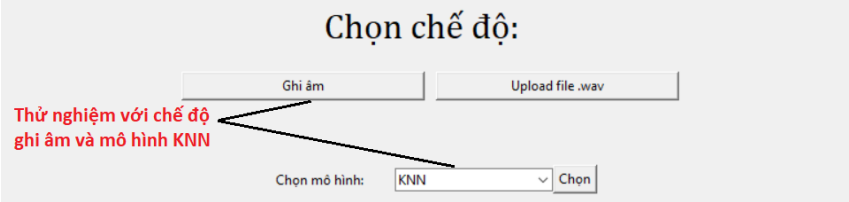
1. Chọn mô hình dự đoán

   Người dùng có thể lựa chọn một trong bốn mô hình dự đoán cảm xúc từ giọng nói. Mỗi mô hình có những đặc điểm riêng, từ mô hình đơn giản như Decision Tree và KNN đến các mô hình phức tạp hơn như Extra Tree và Neural Network, giúp người dùng dễ dàng thử nghiệm và so sánh hiệu quả dự đoán giữa các phương pháp khác nhau.

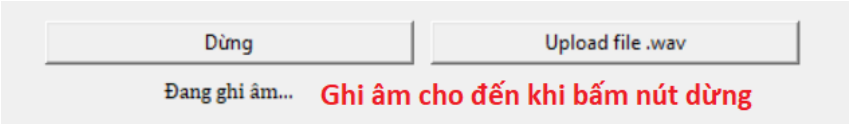
Hình 2.9: Giao diện của ứng dụng

2. Chọn phương thức nhập liệu

- Ghi âm: Người dùng có thể ghi âm một đoạn thoại bất kỳ. Đoạn âm thanh này sẽ được chia nhỏ thành nhiều phần, và mô hình sẽ thực hiện dự đoán cảm xúc cho từng phần âm thanh đó.

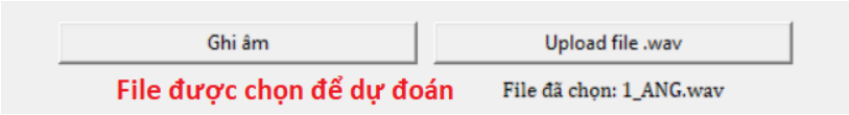


Hình 2.10: Chọn chức năng ghi âm



Hình 2.11: Quá trình ghi âm

- Tải tệp âm thanh: Ngoài việc ghi âm trực tiếp, người dùng còn có thể tải lên một tệp âm thanh có sẵn và yêu cầu ứng dụng đưa ra dự đoán về cảm xúc trong đoạn âm đó.



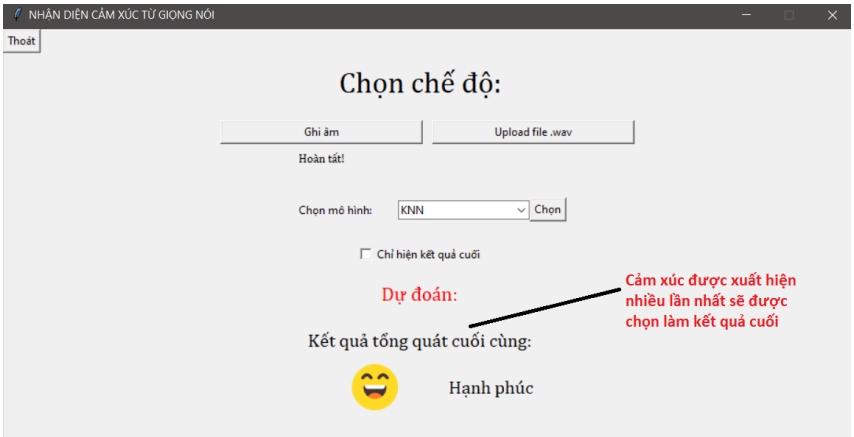
Hình 2.12: Chọn chức năng tải tệp âm thanh

3. Chọn chế độ hiện thị kết quả

   Kết quả dự đoán từ từng đoạn âm thanh nhỏ sẽ được hiển thị lần lượt trên giao diện, kèm theo thanh tiến trình để người dùng có thể theo dõi vị trí của đoạn âm.

Hình 2.13: Quá trình hiển thị kết quả

Sau khi tất cả các đoạn âm thanh được xử lý, kết quả tổng hợp cuối cùng sẽ được hiển thị màn hình.



Hình 2.14: Kết quả cuối cùng

Người dùng có thể yêu cầu chỉ hiện kết quả tổng hợp cuối cùng vằng cách tick và nút "chỉ hiện kết quả cuối".

Chương 3

Thí nghiệm

3.1 Dataset

3.1.1 Giới thiệu Dataset

Dataset đóng vai trò then chốt trong việc huấn luyện và kiểm thử mô hình nhận diện cảm xúc từ giọng nói. Để xây dựng dataset này, chúng ta tổng hợp từ nhiều bộ dữ liệu khác nhau chẳng hạn RAVDESS [1], TESS [2], CREMA-D [3],... Đây là những bộ dữ liệu phong phú, đa dạng với chất lượng cao, đáp ứng các yêu cầu khắt khe của bài toán.

Dataset được tổng hợp và tổ chức, phân chia lại một cách khoa học. Để đảm bảo tính hiệu quả, mỗi mẫu giọng nói được gán một mã số duy nhất, giúp dễ dàng quản lý, chỉnh sửa, và theo dõi trong quá trình huấn luyện và kiểm thử.

Mỗi tệp âm thanh là một mẫu giọng nói tiếng Anh mô phỏng, với nội dung tập trung vào việc thể hiện cảm xúc trong từng câu lệnh ngắn. Mục tiêu của chúng ta là xây dựng một bộ dữ liệu đủ lớn và đa dạng, đảm bảo tính cân bằng giữa các lớp cảm xúc để tối ưu hóa hiệu năng của mô hình nhận diện.

3.1.2 Mô tả dataset

Bộ dữ liệu này bao gồm các đoạn âm thanh được ghi lại từ nhiều người, với sự đa dạng về độ tuổi và giới tính, bao gồm cả nam và nữ, người trẻ và người già. Điều này giúp bộ dữ liệu phản ánh được sự phong phú và đa dạng trong các giọng nói, từ đó hỗ trợ việc phát triển các mô hình phân loại cảm xúc chính xác và linh hoạt hơn.

Từ những dữ liệu thu thập được, chúng ta tiến hành loại bỏ các cảm xúc không cần thiết và thực hiện việc đánh nhãn lại theo các quy tắc sau:

Nhãn dán	Cảm xúc
ANG	Giận dữ
FEA	Sợ hãi
HAP	Vui vẻ
SAD	Buồn
NEU	Trung lập

Bảng 3.1: Bảng quy tắc nhãn dán

Mỗi tệp âm thanh được lưu dưới định dạnh `.wav` với độ dài từ 1 đến 3 giây. Tổng số mẫu thu thập được là 9,086 mẫu. Sau đó, thông qua việc áp dụng các kỹ thuật tăng cường dữ liệu như thay đổi cao độ, thêm nhiễu trắng và tiếng vang, bộ dữ liệu đã được mở rộng lên 23,750 mẫu.

Việc áp dụng kỹ thuật tăng cường dữ liệu không chỉ làm tăng số lượng mẫu mà còn nâng cao chất lượng và độ tổng quát của mô hình, giúp mô hình hoạt động hiệu quả hơn trong các tình huống thực tế.

Bộ dữ liệu này sau đó được chia thành ba tập: Tập huấn luyện (Train), Tập xác thực (Validation) và Tập kiểm tra (Test) với tỉ lệ phân chia là 8:1:1 với số lượng các lớp cảm xúc là như nhau. Dưới đây là thống kê số lượng mẫu theo từng lớp cảm xúc:

Lớp cảm xúc	**Train**	**Validation**	**Test**	**Tổng**
ANG	3800	475	475	4750
FEA	3800	475	475	4750
HAP	3800	475	475	4750
SAD	3800	475	475	4750
NEU	3800	475	475	4750

Bảng 3.2: Phân phối mẫu dữ liệu theo lớp cảm xúc

3.2 Cấu hình chạy thí nghiệm

3.2.1 Phần cứng

Để đảm bảo quá trình thí nghiệm nhận diện cảm xúc bằng giọng nói được thực hiện một cách hiệu quả và đạt kết quả tối ưu, chúng ta phải sử dụng cấu hình phần cứng phù hợp với yêu cầu của cả mô hình học máy và học sâu.

- CPU: 11th Gen Intel(R) Core(TM) i7-1165G7 @ 2.8GHz

  Đây là vi xử lý mạnh mẽ đảm bảo tốc độ xử lý tốt cho các tác vụ tiền xử lý dữ liệu và tính toán học máy.

- GPU 0: Intel(R) Iris(R) Xe Graphics

  GPU tích hợp, hỗ trợ các tác vụ đồ họa cơ bản và tính toán thông thường, nhưng không tối ưu cho các mô hình học sâu phức tạp.

- GPU 1: NVIDIA GeForce MX350

  Có khả năng xử lý các tác vụ học sâu cơ bản và sẽ được sử dụng trong các mô hình học sâu, giúp tăng tốc độ huấn luyện đáng kể so với GPU tích hợp.

- RAM: 8GB

  Đáp ứng các tác vụ xử lý dữ liệu và huấn luyện mô hình cơ bản.

- Hệ điều hành: WIN 11

  Hỗ trợ tốt các ứng dụng cần thiết cho học máy và học sâu.

3.2.2 Phần mềm

Bên cạnh đó, phần mềm và các thư viện chuyên dụng đóng vai trò quan trọng. Dưới đây là một số phần mềm và thư viện được sử dụng trong dự án này:

- Librosa

  Dùng để trích xuất đặc trưng âm thanh từ các tệp âm phục vụ cho quá trình huấn luyện. Thư viện này hỗ trợ các tác vụ như trích xuất đặc trưng từ âm thanh, phân tích tần số, và trực quan hóa.

- Soundfile

  Dùng để đọc và ghi các tệp âm thanh, đặc biệt là định dạng WAV.

- Scikit-learn (sklearn)

  Thư viện mạnh mẽ cho học máy, bao gồm các thuật toán phân loại, hồi quy, đánh giá mô hình, và phân tích dữ liệu. Dùng để gọi ra những mô hình học máy dùng cho việc nhận diện cảm xúc.

- Keras và TensorFlow:

  Là những thư viện học sâu mạnh mẽ, dùng để xây dựng và huấn luyện các mô hình học sâu như mạng nơ-ron.

- Sounddevice

  Dùng để thu âm giọng nói từ microphone để đưa vào mô hình nhận diện cảm xúc.

- Tkinter

  Thư viện xây dựng giao diện người dùng (GUI) cho ứng dụng desktop. Dùng để tạo giao diện người dùng cho việc tải tệp âm thanh, huấn luyện mô hình, và hiển thị kết quả.

3.3 Kết quả chạy thí nghiệm

3.3.1 Kết quả quá trình huấn luyện

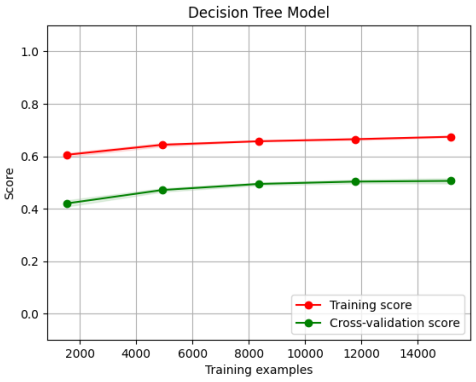
Để đáng giá và giám sát quá trình huấn luyện của mô hình học máy, chúng ta sử dụng đường cong học (Learning Curve). Chúng ta sẽ sử dụng phương pháp K-fold cross-validation, một kỹ thuật đánh giá mô hình phổ biến được sử dụng để phân tích hiệu suất của mô hình học máy khi kích thước của dữ liệu huấn luyện thay đổi. K-fold cross-validation chia tập dữ liệu thành k phần bằng cách phân chia ngẫu nhiên. Sau đó, quá trình huấn luyện và kiểm tra sẽ diễn ra k lần, trong đó mỗi phần sẽ lần lượt được sử dụng làm tập kiểm tra, còn lại sẽ là tập huấn luyện.

Tiếp theo chúng ta phân tích các đường cong học Learning Curve lần lượt của 3 mô hình học máy được sử dụng trong bài toán này:

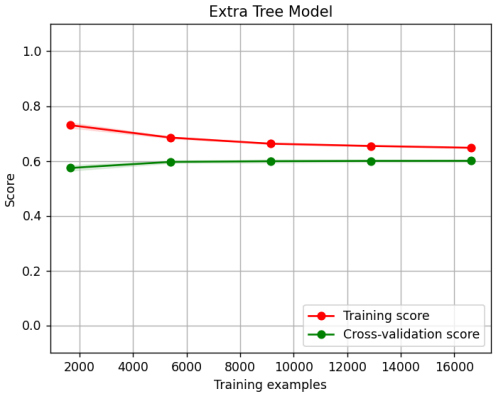
1. Decision Tree Classifier

    - Điểm huấn luyện: Cao (trên 0.6), tăng nhẹ khi tăng số lượng dữ liệu huấn luyện.
    - Điểm kiểm tra chéo: Thấp hơn đáng kể so với điểm huấn luyện (khoảng 0.4-0.5), tăng nhẹ nhưng không cải thiện đáng kể khi tăng dữ liệu.

    Mô hình Decision Tree có xu hướng bị overfitting, mô hình đạt hiệu suất cao trên dữ liệu huấn luyện nhưng lại kém hiệu quả khi áp dụng vào dữ liệu kiểm tra.



Hình 3.1: Đường cong học tập mô hình Decision Tree



Hình 3.2: Đường cong học tập mô hình Extra Tree

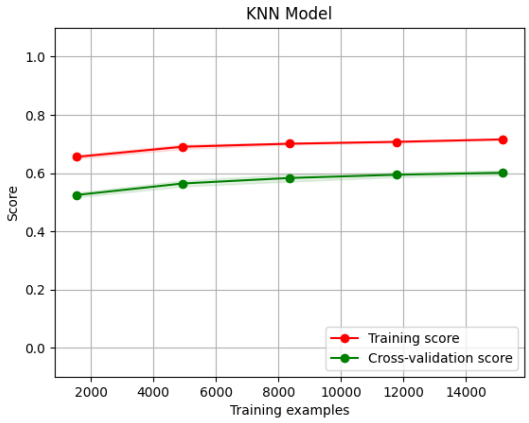
2. Extra Tree Classifier

    - Điểm huấn luyện: Điểm huấn luyện ban đầu cao 0.8, giảm dần và ổn định ở mức 0.6 khi số lượng dữ liệu huấn luyện tăng.
    - Điểm kiểm tra chéo: Điểm kiểm tra chéo tăng nhẹ ở giai đoạn đầu và nhanh chóng ổn định ở mức tương đương với điểm huấn luyện (0.6). Đường kiểm tra chéo rất gần đường huấn luyện, điều này cho thấy mô hình có khả năng tổng quát hóa tốt.

    Extra Tree sử dụng tập hợp các cây ngẫu nhiên, giảm khả năng overfitting bằng cách chia dữ liệu và chọn các đặc trưng ngẫu nhiên. Điều này giúp mô hình ổn định hơn so với Decision Tree.

Tuy nhiên, điểm số của Extra Trees bị giới hạn ở mức khoảng 0.6, nghĩa là nó có thể gặp khó khăn trong việc nắm bắt các đặc trưng phức tạp hơn của dữ liệu cảm xúc.

3. K-Nearest Neighbors



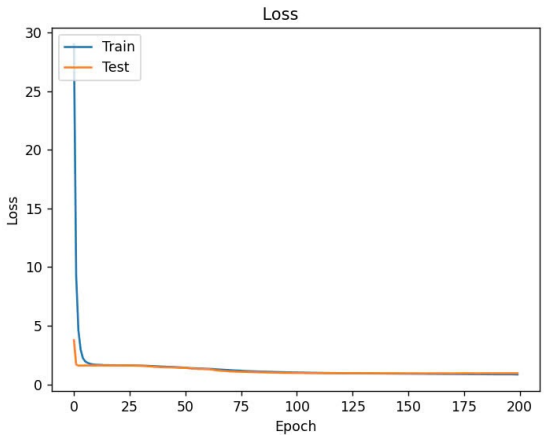
Hình 3.3: Đường cao học tập mô hình KNN

- Điểm huấn luyện: Điểm huấn luyện cao (0.8) và tăng nhẹ khi số lượng dữ liệu tăng.
- Điểm kiểm tra chéo khởi đầu thấp hơn điểm huấn luyện (0.5) nhưng tăng dần khi dữ liệu huấn luyện tăng, đạt mức 0.6 với 15,000 mẫu. Khoảng cách giữa hai đường giảm dần, điều này cho thấy KNN cần nhiều dữ liệu hơn để tổng quát hóa tốt.

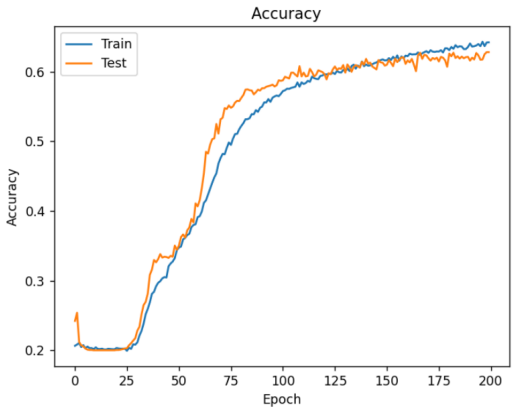
KNN là mô hình phi tham số, hoạt động tốt hơn khi số lượng dữ liệu tăng lên. Điều này lý giải vì sao điểm kiểm tra chéo cải thiện dần khi có nhiều dữ liệu huấn luyện hơn.

Tuy nhiên, nhược điểm chính của KNN là chi phí tính toán cao khi số lượng dữ liệu lớn, do nó tính toán khoảng cách với tất cả các điểm trong tập dữ liệu.

Bên cạnh đó, chúng ta đánh giá mô hình học sâu (Neural Netwwork) bằng đồ thị độ chính xác (Accuracy) và đồ thị mất mát (Loss).



Hình 3.4: Đồ thị mất mát của mô hình học sâu



Hình 3.5: Đồ thị độ chính xác của mô hình học sâu

1. Biểu đồ Loss

    - Đường Train và Test giảm mạnh trong giai đoạn đầu, sau đó giảm dần và ổn định ở mức thấp, không có dấu hiệu chênh lệch quá lớn giữa hai đường.
    - Loss giảm và ổn định cho thấy mô hình đang học tốt và hội tụ sau một số epoch.
    - Quá trình huấn luyện diễn ra hiệu quả với dữ liệu và kiến trúc mô hình phù hợp, không có dấu hiệu overfiting.

2. Biểu đồ Accuracy

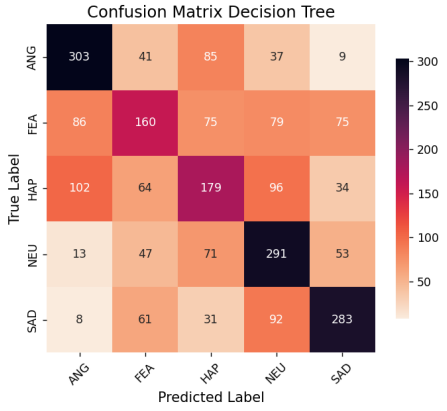
    - Độ chính xác của tập Train và Test tăng đều từ khoảng epoch 25 đến 75.
    - Sau epoch 75, cả hai đường tiếp tục tăng nhưng với tốc độ chậm hơn và cuối cùng đều hội tụ ở mức trên 0.6.
    - Mô hình có độ chính xác tăng dần qua các epoch, chứng tỏ mô hình đang học cách phân loại cảm xúc tốt hơn.

Mô hình có hiệu suất học tốt, không xảy ra hiện tượng overfitting hoặc underfitting. Tuy nhiên, độ chính xác (khoảng 0.6) vẫn chưa hiệu quả đối với bài toán nhận diện cảm xúc.

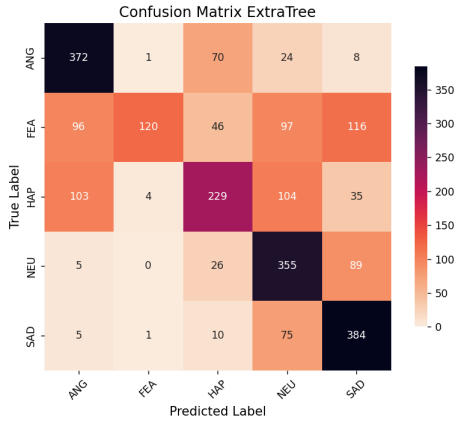
3.3.2 Kết quả quá trình thí nghiệm

Ở bước này, chúng ta sẽ sử dụng ma trận nhầm lẫn (Confusion Matrix) để đánh giá hiệu suất của một mô hình phân loại. Mục đích chính của biểu đồ này là cung cấp một cái nhìn chi tiết về cách mô hình phân loại các nhãn, bao gồm cả dự đoán đúng và nhầm lẫn giữa các lớp.

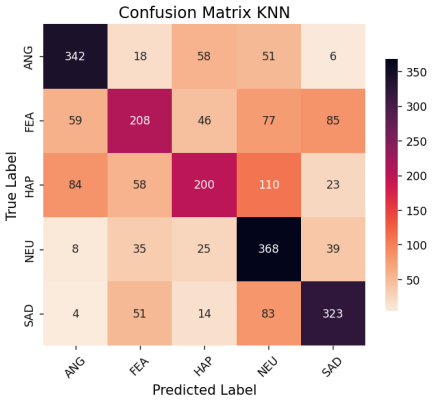
Cấu trúc của ma trận nhầm lẫn bao gồm các hàng và cột đại diện cho các nhãn thực tế và nhãn dự đoán. Mỗi ô trong ma trận biểu thị số lượng mẫu mà mô hình phân loại chính xác hoặc sai cho từng lớp. Các ô chéo trên ma trận đại diện cho số lượng dự đoán đúng, trong khi các ô ngoài chéo thể hiện các dự đoán sai (nhầm lẫn giữa các lớp).



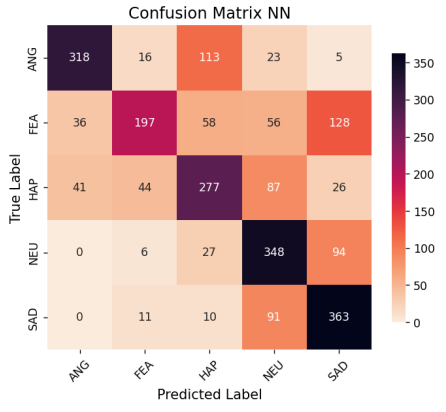
Hình 3.6: Ma trận nhầm lẫn mô hình Decision Tree



Hình 3.7: Ma trận nhầm lẫn mô hình Extra Tree



Hình 3.8: Ma trận nhầm lẫn mô hình KNN



Hình 3.9: Ma trận nhầm lẫn mô hình học sâu NN

Mô hình Decision Tree nhầm lẫn nhiều giữa các lớp như FEA và HAP. Tuy nhiên, các lớp ANG và SAD được phân loại khá chính xác. Hiệu suất của Decision Tree ở mức trung bình.

So với Decision Tree, Extra Tree cải thiện rõ rệt ở các lớp NEU và SAD, với ít nhầm lẫn hơn. Tuy nhiên, lớp FEA vẫn dễ bị nhầm lẫn với các lớp khác. Hiệu suất tốt hơn nhờ cơ chế phân vùng ngẫu nhiên và sử dụng nhiều cây trong mô hình.

Mô hình KNN hoạt động tốt đối với các lớp NEU và ANG, nhưng vẫn có nhầm lẫn giữa FEA, HAP và SAD. Hiệu suất của KNN phụ thuộc vào số lượng láng giềng và phân bố dữ liệu, nhưng chưa đủ mạnh để phân biệt các lớp gần nhau trong không gian đặc trưng.

Neural Network thể hiện hiệu suất so với các mô hình trước đó. Tuy nhiên, vẫn còn một số nhầm lẫn ở các lớp HAP và FEA.

Dưới đây là bảng so sánh độ chính xác của các mô hình khác nhau trong việc phân loại các lớp cảm xúc.

Mô hình	Độ chính xác
Decision Tree	0.512
Extra Tree	0.615
KNN	0.607
Neural Network	0.633

Bảng 3.3: Bảng biểu thị độ chính xác của các mô hình

Decision Tree chỉ đạt được độ chính xác 51,2%, cho thấy hiệu suất còn khá khiêm tốn và tồn tại nhiều hạn chế, các phương pháp như KNN và Extra Trees có phần cải thiện với độ chính xác lần lượt là 60,7% và 61,5%. Tuy nhiên, các phương pháp này vẫn chưa thực sự tối ưu. Trong khi đó, mạng nơ-ron thể hiện sự vượt trội với độ chính xác đạt 63,3%, cho thấy tiềm năng mạnh mẽ của nó trong việc giải quyết bài toán này.

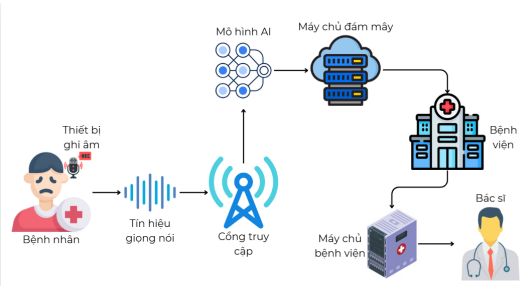
Chương 4

Kết luận

Qua nghiên cứu, nhóm đã xây dựng hệ thống nhận diện cảm xúc từ giọng nói bằng cách kết hợp các phương pháp xử lý tín hiệu âm thanh hiện đại với các mô hình học máy như Decision Tree, Extra Trees, KNN, và mạng nơ-ron nhân tạo (Neural Network).

Quá trình thử nghiệm cho thấy mô hình đạt hiệu suất khá tốt trong việc phân loại các cảm xúc cơ bản như vui, buồn, giận dữ, sợ hãi và trung tính. Mô hình cũng đã được tích hợp giao diện thân thiện với người dùng, cho phép ghi âm hoặc tải lên các tệp âm thanh để dự đoán cảm xúc. Điều này khẳng định tiềm năng ứng dụng trong các lĩnh vực thực tiễn như chăm sóc khách hàng, giáo dục, y tế, an ninh mạng, và trò chơi thực tế ảo.

Hình 4.1: Hệ thống nhận diện cảm xúc giọng nói ngăn chặn các sự số an ninh trong an ninh mạng



Hình 4.2: Hệ thống nhận diện cảm xúc qua giọng nói trong y tế

Mặc dù đạt được nhiều kết quả tích cực trong quá trình kiểm thử, hệ thống vẫn gặp một vài mặt hạn chế như hiệu suất chưa cao khi phân loại các cảm xúc tương đồng. Để giải quyết các hạn chế hiện tại và nâng cao chất lượng hệ thống, nhóm sẽ tập trung vào các hướng phát triển sau:

- Nâng cấp Dataset: Bổ sung thêm dữ liệu từ môi trường thực tế, bao gồm các tình huống phức tạp như giọng nói có nhiều nhiễu, giọng nói bị méo, hoặc cảm xúc bị che lấp bởi ngữ cảnh.

Hình 4.3: Nâng cấp Dataset

- Cải tiến mô hình: Áp dụng các thuật toán tiên tiến như các mô hình dựa trên Transformer Based Models hoặc thực hiện tinh chỉnh *fine-tuning* trên các mô hình tiền huấn luyện *pre-trained models* để nâng cao độ chính xác và tối ưu hóa thời gian xử lý của mô hình.

Hình 4.4: Pre-training và Fine-Tuning

Nhận biết cảm xúc từ giọng nói là một lĩnh vực nghiên cứu đầy tiềm năng, mở ra nhiều cơ hội ứng dụng trí tuệ nhân tạo vào cuộc sống, hứa hẹn mang lại những giải pháp đột phá giúp đời sống con người ngày càng cải thiện.

Tài liệu tham khảo

[1] UWRF Kaggle. *RAVDESS Emotional Speech Audio*. Accessed: 2024-12-01. 2018. URL: https://www.kaggle.com/datasets/uwrfkaggler/ravdess-emotional-speech-audio.

[2] Ejlok1. *Toronto Emotional Speech Set (TESS)*. Accessed: 2024-12-08. 2018. URL: https://www.kaggle.com/datasets/ejlok1/toronto-emotional-speech-set-tess.

[3] Ejlok1. *CREMA-D: Crowdsourced Emotional Multimodal Actors Dataset*. Accessed: 2024-12-08. 2018. URL: https://www.kaggle.com/datasets/ejlok1/cremad.

Báo cáo làm việc hàng tuần

1. Ngày họp 28/11

- **Thành viên tham dự:** 6/6
- **Nội dung:** Chuẩn bị bộ dữ liệu và tìm hiểu các mô hình

Thành viên	**Công việc được giao**	**Tiến độ hoàn thành tuần trước**
Phát	Chuẩn bị bộ dữ liệu	
Phúc	Báo cáo tổng quan mô hình Extra Tree	
Huy	Báo cáo tổng quan mô hình KNN	
Khôi	Báo cáo tổng quan mô hình Neural Network	
Bách	Demo Latex, viết phần giới thiệu, xử lý dữ liệu	
Đức	Báo các tổng quan mô hình Decision Tree	

Bảng 4.1: Phân công công việc ngày 28/11

2. Ngày họp 05/12

- **Thành viên tham dự:** 6/6
- **Nội dung:** Huấn luyện, đáng giá các mô hình.

Thành viên	**Công việc được giao**	**Tiến độ hoàn thành tuần trước**
Phát	Nhận xét đánh giá các mô hình	100%
Phúc	Xây dựng ứng dụng	90%
Huy	Xây dựng ứng dụng	90%
Khôi	Huấn luyện mô hình KNN và Neural Network	95%
Bách	Báo cáo dataset, cấu hình thí nghiệm	100%
Đức	Huấn luyện mô hình Decision Tree và Extra Tree	95%

Bảng 4.2: Phân công công việc ngày 05/12

3. Ngày họp 12/12

- **Thành viên tham dự:** 6/6
- **Nội dung:** Thiết kế slide, quay video demo

Thành viên	**Công việc được giao**	**Tiến độ hoàn thành tuần trước**
Phát	Quay video demo	90%
Phúc	Thiết kế slide phần tổng quan các mô hình	90%
Huy	Thiết kế silde phần giải thích mô hình	90%
Khôi	Thiết kế slide phần giải thích mô hình	100%
Bách	Thiết kế slide phần xử lý dữ liệu	95%
Đức	Thiết kế slide phần tổng quan các mô hình	100%

Bảng 4.3: Phân công công việc ngày 12/12

4. Ngày họp 17/12

- **Thành viên tham dự:** 6/6
- **Nội dung:** Kiểm tra nội dung báo cáo, slide và thuyết trình

Thành viên	**Công việc được giao**	**Tiến độ hoàn thành tuần trước**
Phát	Kiểm tra nội dung báo cáo	100%
Phúc	Thuyết trình phần đáng giá mô hình	100%
Huy	Kiểm tra nội dung code	100%
Khôi	Thuyết trình phần giải thích mô hình	100%
Bách	Thuyết trình phần xử lý dữ liệu, tổng quan mô hình	100%
Đức	Kiểm tra nội dung slide	100%

Bảng 4.4: Phân công công việc ngày 17/12