

All-in-One Course 2023

Music Genre Classification

(Time-series Data Project)

2023

Các bạn có thể tham khảo Code chi tiết tại link [Colab](#).

Phần I: Giới thiệu

Trong lĩnh vực **Truy xuất Thông tin Âm nhạc (MIR)**, **Phân loại Thể loại Âm nhạc** đóng một vai trò quan trọng. Bộ dữ liệu **GTZAN**, thường được xem như "MNIST của âm thanh", bao gồm 1.000 bản nhạc được phân loại theo 10 thể loại khác nhau, từ Blues đến Rock. Mỗi bản nhạc kéo dài 30 giây và đi kèm với biểu diễn hình ảnh và dữ liệu đặc trưng đã được trích xuất.

Xử lý âm thanh là một lĩnh vực nghiên cứu phức tạp, đòi hỏi sự hiểu biết về xử lý tín hiệu thời gian, chuỗi thời gian và trích xuất đặc trưng âm thanh. Để xây dựng một hệ thống đề xuất âm nhạc hiệu quả, việc phân loại chính xác thể loại âm nhạc là bước quan trọng, và GTZAN cung cấp một nguồn dữ liệu quý giá cho mục tiêu này.

Để tải bộ data GTZAN các bạn hãy chạy dòng lệnh sau

```
1 !gdown 1MGhyeMngD6P9Kz9zJpL68y1QaIQvW7Zx
2 !unzip GTZAN.zip -d /content/GTZAN
3 !rm GTZAN.zip
```

Tuy nhiên, mặc dù GTZAN đã trích xuất sẵn một số đặc trưng, để có thể hiểu sâu hơn về những đặc trưng này chúng ta sẽ bắt đầu lại từ đầu, trực tiếp từ các tệp âm thanh, để hiểu rõ hơn về quá trình trích xuất và phân tích đặc trưng tín hiệu nói chung và âm nhạc nói riêng.

Chạy dòng lệnh này để xoá các feature có sẵn trong bộ GTZAN dataset

```
1 !rm -r /content/GTZAN/images_original
2 !rm /content/GTZAN/features_30_sec.csv
3 !rm /content/GTZAN/features_3_sec.csv
```

Phần II: Xử lý tín hiệu - Nền tảng

Chúng ta sẽ dùng một bản nhạc để làm ví dụ cho nội dung này.

```
1 data_path = '/content/GTZAN/genres_original/blues/blues.00000.wav'
```

Tệp này sử dụng định dạng .wav - "Waveform Audio File Format", một định dạng không nén do Microsoft và IBM phát triển. Để phân tích tệp .wav một cách chính xác, chúng ta có thể sử dụng thư viện **librosa** trong Python, thư viện này cung cấp rất nhiều công cụ xử lý âm thanh dựa trên lý thuyết khoa học.

Để đọc được tệp .wav các bạn thực hiện câu lệnh sau

```
1 import librosa
2
3 # Load a .wav file using librosa
4 data, sr = librosa.load(data_path) # Sampling rate = 22050
```

Theo mặc định, thư viện librosa sẽ tải bản nhạc của chúng ta với **sampling rate = 22.050 Hz**. Chúng ta sẽ tìm hiểu về ý nghĩa của thông số này sau. **Các bạn có thể nghe trực tiếp bản nhạc đã load trên Colab bằng câu lệnh sau**

```
1 # Playing audio file
2 import IPython
3
4 IPython.display.Audio(data, rate=sr)
```

1. Biểu đồ sóng (Waveform)

Âm thanh là một dạng năng lượng do sự rung động tạo ra. Khi bất kỳ vật thể nào rung động, nó tạo ra sự di chuyển của các hạt không khí. Những hạt này va chạm vào các hạt xung quanh, khiến chúng cũng rung động và va chạm vào nhiều hạt không khí khác. Sự di chuyển này, được gọi là sóng âm, tiếp tục cho đến khi nó mất hết năng lượng. Khi những sóng âm này đến tai chúng ta, chúng làm cho màng nhĩ của chúng ta cũng rung động, và não bộ giải mã điều này thành âm thanh.

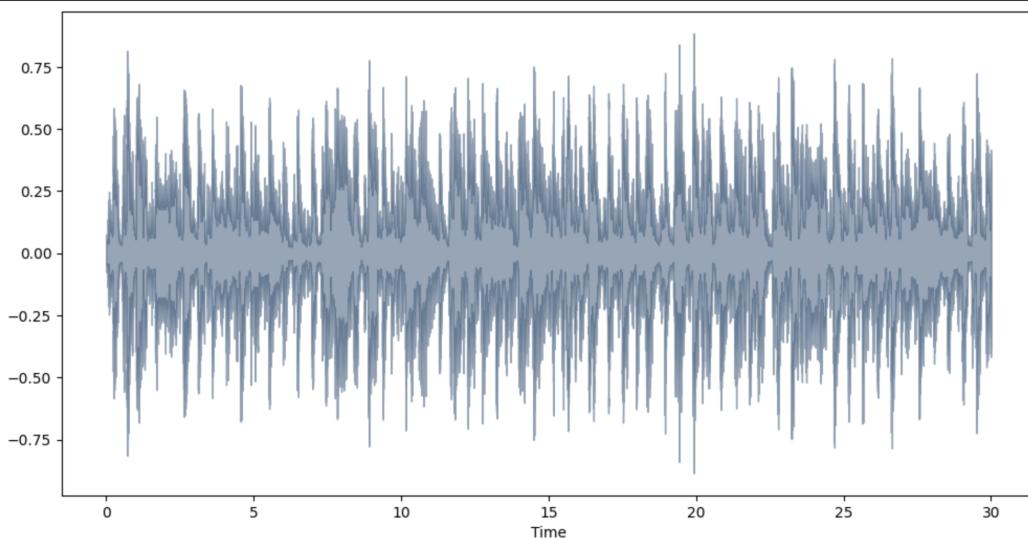
Biểu đồ sóng là biểu diễn hình dạng và cấu trúc của một tín hiệu (như âm thanh) qua biến đổi biên độ theo thời gian. Trong ngữ cảnh của âm thanh, biểu đồ sóng mô tả đồ họa sự biến đổi áp lực (hoặc biên độ) của sóng âm so với thời gian.

Biểu đồ sóng là biểu diễn hình ảnh của một tín hiệu âm thanh. Nó hiển thị cách biên độ của tín hiệu âm thanh thay đổi theo thời gian. Trực đọc biểu diễn biên độ, có thể được coi là độ lớn hoặc cường độ của âm thanh. Trục ngang biểu diễn thời gian. Những đỉnh trên biểu đồ sóng biểu diễn những khoảnh khắc có biên độ cao (âm thanh lớn), trong khi những đáy biểu diễn những khoảnh khắc có biên độ thấp (âm thanh nhẹ).

Cơ bản, biểu đồ sóng cung cấp cho chúng ta một cái nhìn trực quan về động lực của âm thanh. Nó cho phép chúng ta thấy độ lớn và độ nhẹ của tín hiệu âm thanh trong suốt thời gian diễn ra, giúp ta hiểu rõ hơn về đặc điểm và cấu trúc của nó.

Để vẽ biểu đồ sóng, các bạn chạy dòng lệnh sau

```
1 import matplotlib.pyplot as plt
2
3 # Wave form of the audio
4 plt.figure(figsize=(12,6))
5 librosa.display.waveplot(data, color="#2B4F72", alpha = 0.5)
6 plt.show()
```



Hình 1: Biểu đồ sóng - Waveform

2. Tốc độ lấy mẫu (Sampling Rate)

Khi chúng ta mở một tệp .wav bằng librosa, tốc độ lấy mẫu mặc định là 22.050Hz. Vậy giá trị này có ý nghĩa gì?

Âm thanh, trong dạng tự nhiên, là một tín hiệu liên tục. Để xử lý số hóa tín hiệu này, chúng cần được chuyển đổi thành định dạng số rồi rạc thông qua quá trình 'lấy mẫu'. 'Tốc độ lấy mẫu', được đo bằng Hertz (Hz), biểu thị số lượng mẫu được lấy mỗi giây từ tín hiệu liên tục.

Tốc độ lấy mẫu có vai trò quan trọng trong xử lý tín hiệu. Nó đảm bảo tín hiệu số được biểu diễn chính xác từ tín hiệu analog. Theo Định lý Lấy mẫu Nyquist-Shannon, để tránh mất thông tin, tín hiệu cần được lấy mẫu ít nhất gấp đôi tần số cao nhất của nó.

Tuy nhiên, việc chọn tốc độ lấy mẫu cũng có hậu quả thực tế. Tốc độ lấy mẫu cao hơn dẫn đến dữ liệu lớn và yêu cầu tính toán cao, trong khi tốc độ thấp có thể làm mất chi tiết tín hiệu.

Trong Trích xuất Thông tin Âm nhạc (MIR), tốc độ lấy mẫu phù hợp là rất quan trọng. MIR liên quan đến việc trích xuất đặc trưng từ tín hiệu âm thanh cho các nhiệm vụ như phát hiện cao độ hoặc phân loại thể loại. Tốc độ lấy mẫu tối ưu đảm bảo trích xuất đặc trưng chính xác.

Thông thường, việc chọn tốc độ lấy mẫu phụ thuộc vào ứng dụng. Đối với việc phát lại âm thanh tiêu chuẩn, tốc độ 44,1 kHz được sử dụng, bắt đầu toàn bộ dải nghe của con người, khoảng 20 Hz đến 20 kHz. Các ứng dụng âm thanh chuyên nghiệp có thể sử dụng tốc độ 48 kHz, 96 kHz hoặc thậm chí 192 kHz. Đối với ứng dụng dành riêng cho giọng nói, tốc độ 8 kHz hoặc 16 kHz là có thể đáp ứng đủ.

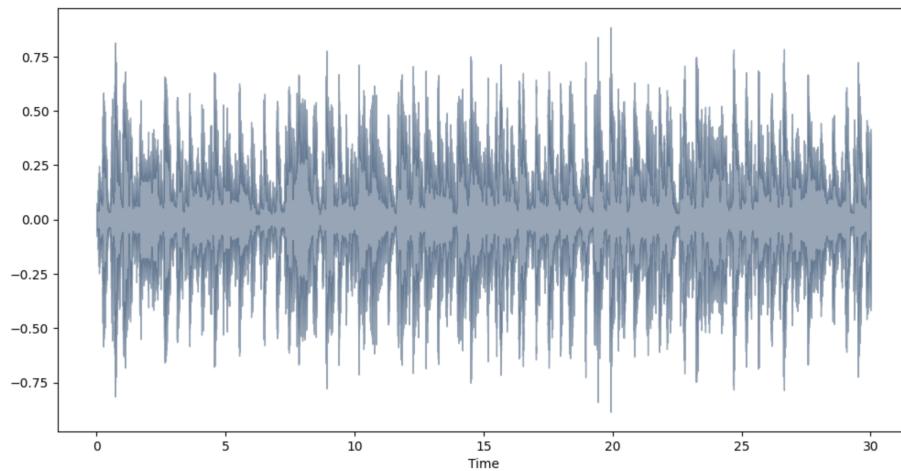
Trong ngữ cảnh của MIR, 44,1 kHz thường được sử dụng do cân bằng giữa việc bắt đầu toàn bộ dải nghe của con người và hiệu quả tính toán. Tuy nhiên, nhiệm vụ hoặc nguồn lực tính toán cụ thể có thể yêu cầu tốc độ khác nhau.

Tóm lại, tốc độ lấy mẫu là một tham số quan trọng trong xử lý tín hiệu số, đảm bảo cả việc biểu diễn chính xác của tín hiệu analog và khả năng tính toán.

So sánh ảnh hưởng của các tốc độ lấy mẫu khác nhau của cùng một bài nhạc

- Sampling rate = 22.050 Hz

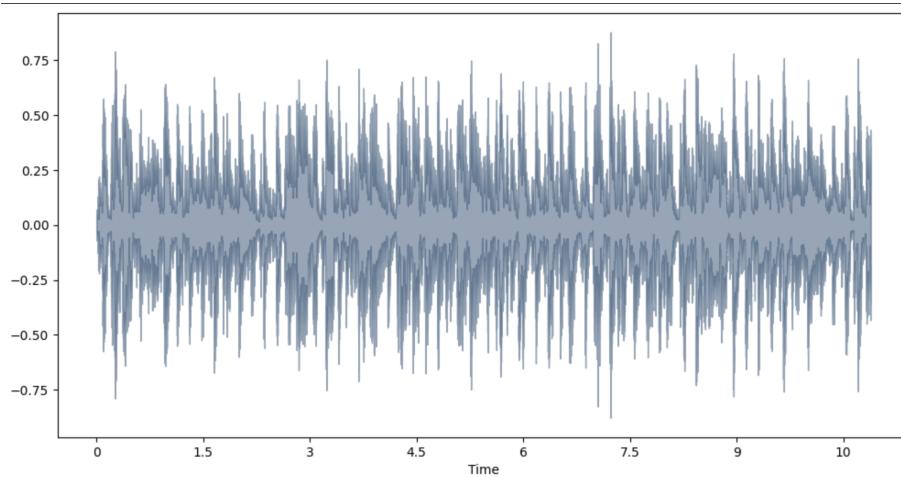
```
1 data, sr = librosa.load(data_path)
2
3 plt.figure(figsize=(12,6))
4 librosa.display.waveplot(data, color="#2B4F72", alpha = 0.5)
5 plt.show()
```



Hình 2: Biểu đồ sóng - Sampling rate = 22.050 Hz

- Sampling rate = 8000 Hz

```
1 data, sr = librosa.load(data_path, sr=8000)
2
3 plt.figure(figsize=(12,6))
4 librosa.display.waveplot(data, color="#2B4F72", alpha = 0.5)
5 plt.show()
```



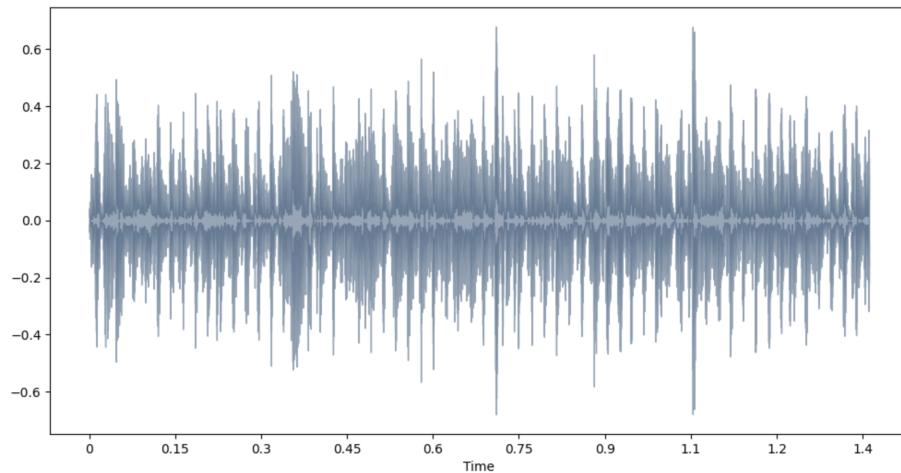
Hình 3: Biểu đồ sóng - Sampling rate = 8000 Hz

- Sampling rate = 1000 Hz

```

1 data, sr = librosa.load(data_path, sr=1000)
2
3 plt.figure(figsize=(12,6))
4 librosa.display.waveplot(data, color="#2B4F72", alpha = 0.5)
5 plt.show()

```



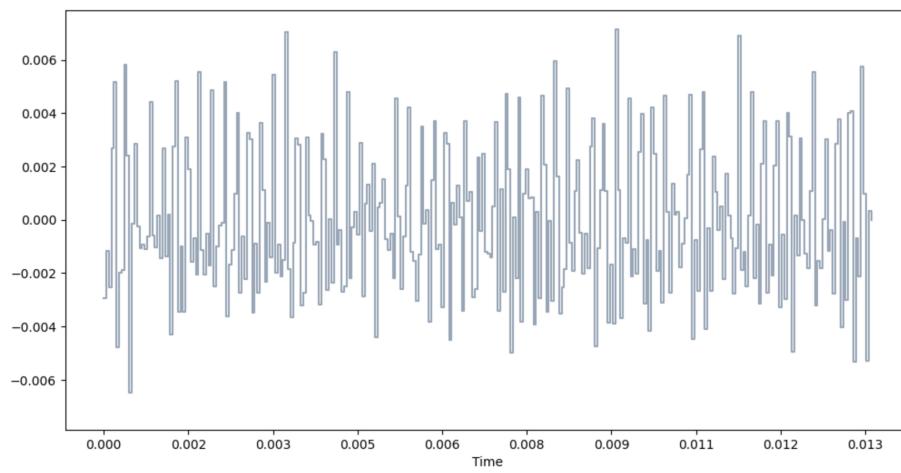
Hình 4: Biểu đồ sóng - Sampling rate = 1000 Hz

- Sampling rate = 10 Hz

```

1 data, sr = librosa.load(data_path, sr=10)
2
3 plt.figure(figsize=(12,6))
4 librosa.display.waveplot(data, color="#2B4F72", alpha = 0.5)
5 plt.show()

```



Hình 5: Biểu đồ sóng - Sampling rate = 10 Hz

Các bạn có thể nghe sự khác biệt trong âm thanh của bốn giá trị này trong Colab

3. Biểu đồ phổ (Spectrogram)

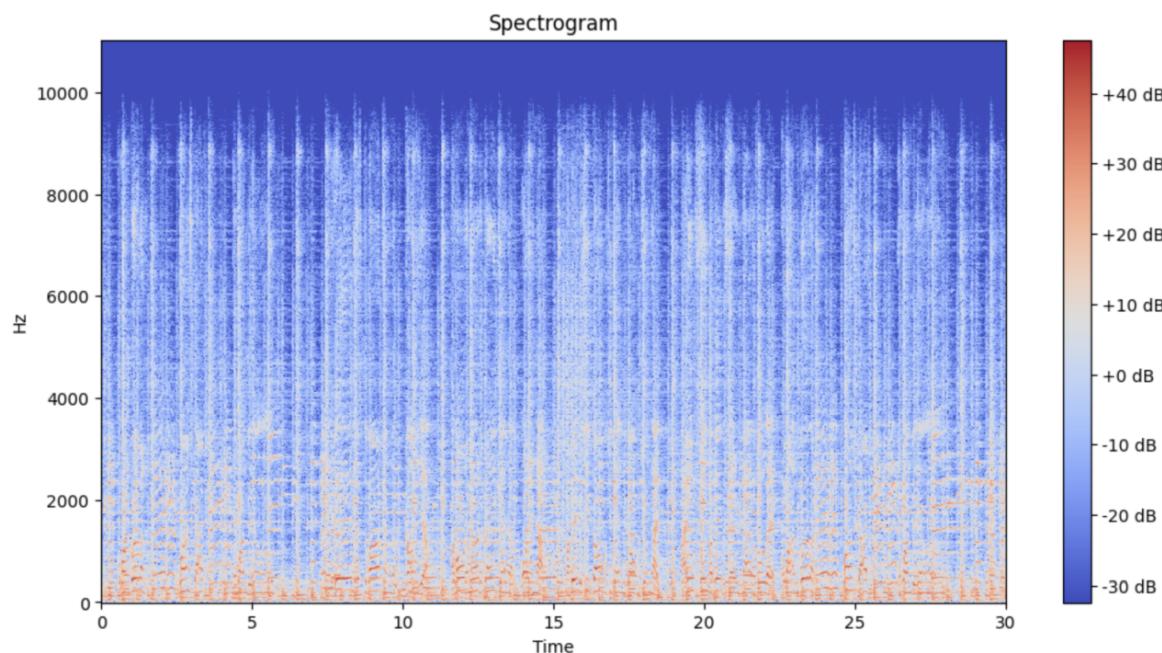
Khác với biểu đồ sóng, biểu đồ phổ phân tách một tín hiệu âm thanh thành các tần số thành phần riêng lẻ. Sau đó, nó vẽ biểu đồ biên độ của từng thành phần tần số theo thời gian. So với phân tích biểu đồ sóng, biểu đồ phổ thích hợp hơn để nắm bắt các đặc điểm liên quan đến tần số, cao độ và âm sắc của âm nhạc. Nó cung cấp cái nhìn chi tiết về cách các tần số khác nhau phân phối theo thời gian, giúp rất nhiều trong các nhiệm vụ như nhận dạng giọng nói, phân tích âm nhạc và hiểu biết về các bức tranh âm thanh phức tạp. Tuy nhiên, mặc dù nó xuất sắc trong việc biểu diễn trong miền tần số, nó có thể không cung cấp nhiều thông tin về động lực thời gian và nhịp điệu của âm nhạc.

Để vẽ biểu đồ phổ các bạn chạy dòng lệnh sau

```

1 # Let load .wav file with default sampling rate of 22,050Hz
2 data, sr = librosa.load(data_path)
3
4 # Spectrogram of the audio
5 stft=librosa.stft(data)
6 stft_db=librosa.amplitude_to_db(abs(stft))
7 plt.figure(figsize=(12,6))
8 librosa.display.specshow(stft_db,sr=sr,x_axis='time',y_axis='hz')
9 plt.title('Spectrogram')
10 plt.colorbar(format='%.2f dB')

```



Hình 6: Biểu đồ phổ - Sampling rate = 22.050 Hz

Nói một cách đơn giản, biểu đồ phổ cho thấy tần số khác nhau xuất hiện hoặc vắng mặt tại các khoảnh khắc khác nhau trong tín hiệu âm thanh.

4. Phổ Mel (Mel-Spectrogram)

Mặc dù biểu đồ phổ cung cấp cái nhìn chi tiết về nội dung tần số của tín hiệu theo thời gian, chúng có thể không luôn phù hợp với cách thị giác âm thanh của con người hoạt động. Tai của chúng ta không nhận biết tất cả các tần số một cách bình đẳng; chúng ta nhạy hơn đối với một số khoảng tần số, đặc biệt là các khoảng tần số tương ứng với giọng nói và nhiều công cụ nhạc cụ. Sự tỷ lệ tần số tuyến tính của biểu đồ phổ có thể không phải lúc nào cũng hiệu quả hoặc cung cấp thông tin đầy đủ khi phân tích âm thanh dưới góc độ cách con người cảm nhận chúng.

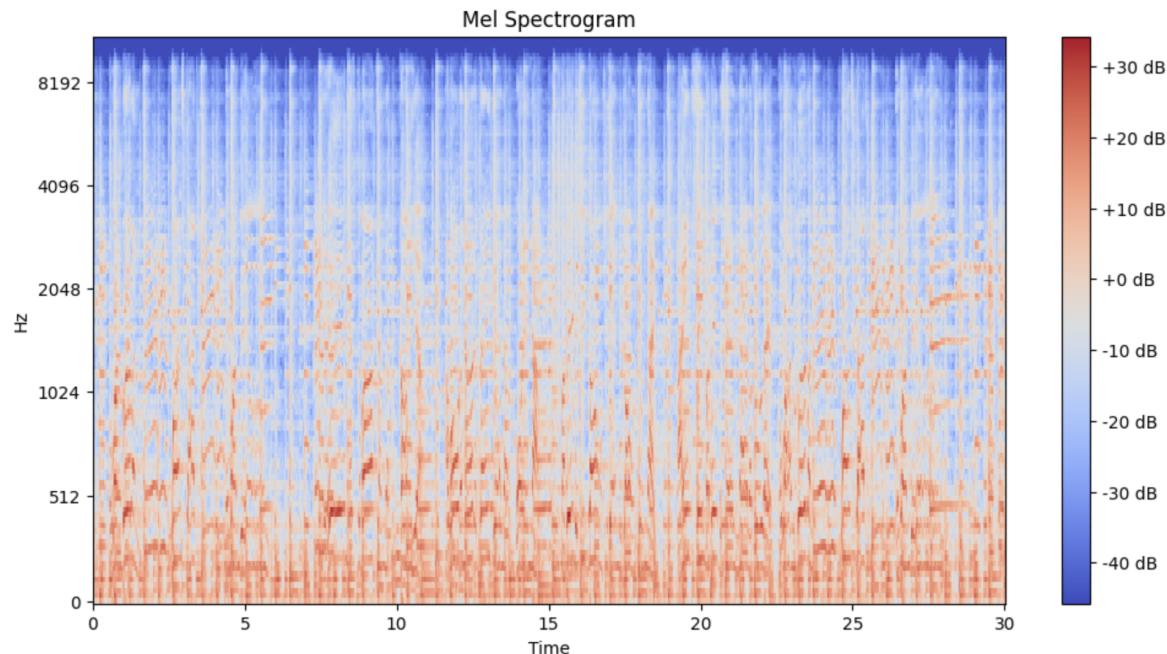
Để giải quyết được giới hạn này, chúng ta có Phổ Mel. Phổ Mel dựa trên thang Mel, một thang nhạc lý được người nghe đánh giá có khoảng cách bằng nhau giữa chúng. Thang này gần hơn với phản ứng của tai người đối với các tần số khác nhau, làm cho nó đặc biệt quý báu trong các nhiệm vụ như phân tích giọng nói và âm nhạc.

Để vẽ Phổ Mel các bạn chạy dòng lệnh sau

```

1 # Let load .wav file with default sampling rate of 22,050Hz
2 data, sr = librosa.load(data_path)
3
4 # Creating log mel spectrogram
5 plt.figure(figsize=(12, 6))
6 spectrogram = librosa.feature.melspectrogram(y=data, sr=sr, n_mels=128, fmax=sr
    //2)
7 spectrogram = librosa.power_to_db(spectrogram)
8 librosa.display.specshow(spectrogram, y_axis='mel', fmax=sr//2, x_axis='time');
9 plt.title('Mel Spectrogram')
10 plt.colorbar(format='%+2.0f dB')

```



Hình 7: Phổ Mel - Sampling rate = 22.050 Hz

Phổ Mel là biểu diễn của phổ công suất ngắn hạn của âm thanh, tương tự như biểu đồ phổ, nhưng với trục tần số bị bóp méo thành thang Mel. Việc bóp méo trục tần số này làm cho phổ Mel trở nên đại diện hơn về cảm nhận thị giác âm thanh của con người.

Một cách thực tế, phổ Mel nhấn mạnh các dải tần số quan trọng nhất cho việc nghe và hiểu của con người. Ví dụ, nó cung cấp độ phân giải cao hơn cho các dải tần số mà tai người cảm nhận nhiều nhất (như tần số tương ứng với giọng nói) và độ phân giải thấp hơn ở nơi tai người cảm nhận ít hơn (như tần số rất cao).

5. Năng lượng hiệu dụng (RMS-E)

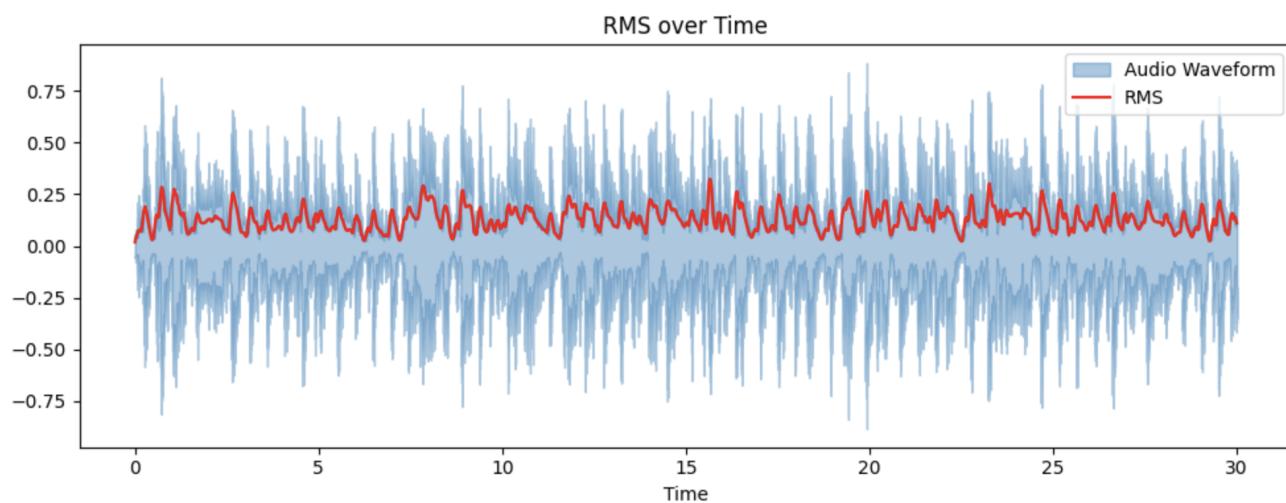
Trong ngữ cảnh xử lý âm thanh và tín hiệu, RMS-E thường được sử dụng để đo "âm lượng cảm nhận" của tín hiệu âm thanh.

RMS-E là một số liệu quan trọng trong âm thanh bởi vì cách chúng ta cảm nhận độ ồn liên quan mật thiết đến mức độ RMS của âm thanh hơn là mức đỉnh của nó. Hai âm thanh có cùng mức đỉnh có thể có độ ồn cảm nhận khác nhau nếu một trong số chúng có mức RMS-E cao hơn. Đây là lý do tại sao RMS thường được sử dụng trong các thuật toán chuẩn hóa và nén âm thanh để đảm bảo độ ồn cảm nhận được đồng nhất trên các tập âm thanh.

Để hình dung được RMS-E các bạn chạy dòng lệnh sau

```

1 # Let load .wav file with default sampling rate of 22,050Hz
2 data, sr = librosa.load(data_path)
3
4 # Compute RMS
5 rms = librosa.feature.rms(y=data)
6
7 # Plot RMS
8 plt.figure(figsize=(10, 4))
9 librosa.display.waveshow(data, sr=sr, alpha=0.4, label='Audio Waveform')
10 plt.plot(librosa.times_like(rms[0], sr=sr), rms[0], color='r', label='RMS')
11 plt.legend(loc='upper right')
12 plt.title('RMS over Time')
13 plt.tight_layout()
14 plt.show()
```



Hình 8: RMS-E

6. Tần số cắt không (Zero-Crossing Rate)

Tần số cắt không (Zero Crossing Rate - ZCR) là một đặc trưng cơ bản trong xử lý và phân tích tín hiệu âm thanh. Nó biểu thị tốc độ mà tín hiệu thay đổi dấu, tức là đi từ dương sang âm hoặc ngược lại.

Tần số cắt không cao thường cho thấy sự hiện diện của các tần số cao hơn trong tín hiệu, trong khi tần số cắt không thấp gợi ý các tần số thấp hơn. Điều này hợp lý vì các thành phần tần số cao hơn sẽ làm cho tín hiệu dao động (băng qua zero) thường xuyên hơn.

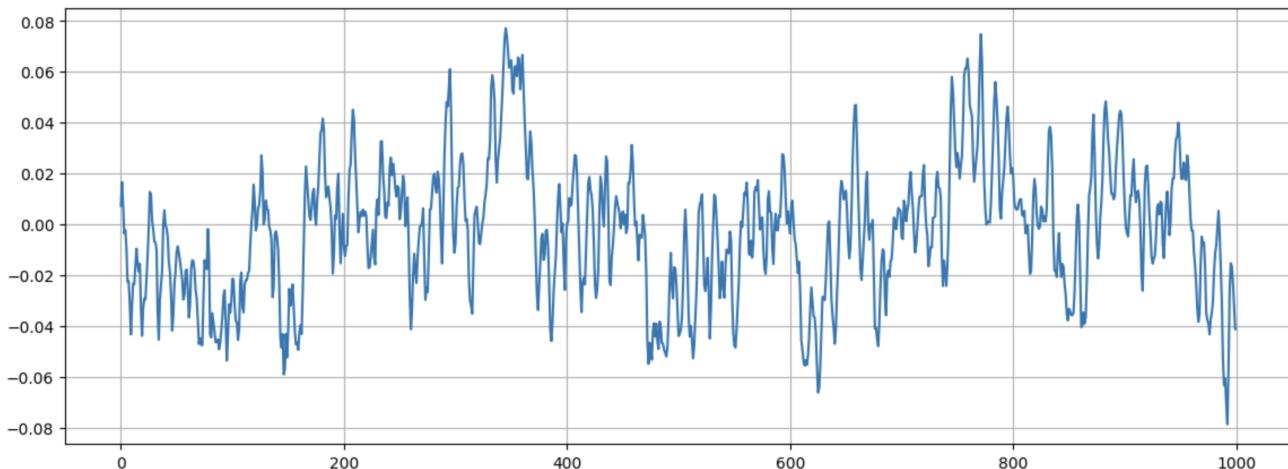
Tiếng ồn, đặc biệt là tiếng ồn có tần số cao, có thể dẫn đến sự gia tăng của các sự băng qua zero. Do đó, một tần số cắt không cao không mong đợi đôi khi có thể là một chỉ báo của tiếng ồn.

Để hình dung được ZCR các bạn chạy dòng lệnh sau

```

1 # Let load .wav file with default sampling rate of 22,050Hz
2 data, sr = librosa.load(data_path)
3
4 # Calculate ZRC of the first 1000 data point of our song
5 n0 = 0
6 n1 = 1000
7 plt.figure(figsize=(14, 5))
8 plt.plot(data[n0:n1])
9 plt.grid()
10
11 zero_crossings = librosa.zero_crossings(data[n0:n1], pad=False)
12 print(f'ZRC = {sum(zero_crossings)}')
13
14 >>> ZRC = 131

```



Hình 9: Tần số cắt không

Trong xử lý giọng nói, tần số cắt không có thể giúp phân biệt giữa các đoạn có âm hữu thanh và âm vô thanh. Các đoạn hữu thanh, như nguyên âm, thường có tần số cắt không thấp hơn, trong khi các đoạn vô thanh, như các phụ âm như "s" hoặc "f", thường có tần số cắt không cao hơn. Đặc trưng này đã được sử dụng rộng rãi trong cả việc nhận dạng giọng nói và trích xuất thông tin âm nhạc. Thông thường, nó có giá trị cao hơn cho các âm thanh có tính xúc tác cao như trong nhạc và rock.

7. Spectral roll-off

Spectral roll-off là một mô tả ngắn gọn về hình dạng phẳng của tín hiệu âm thanh, cung cấp thông tin về phân phôi tần số và nội dung harmonics, là một đặc trưng được sử dụng trong xử lý và phân tích tín hiệu âm thanh. Spectral roll-off đại diện cho một ngưỡng mà dưới đó chứa một phần trăm cụ thể (thường là từ 85% đến 95%) của tổng năng lượng phẳng.

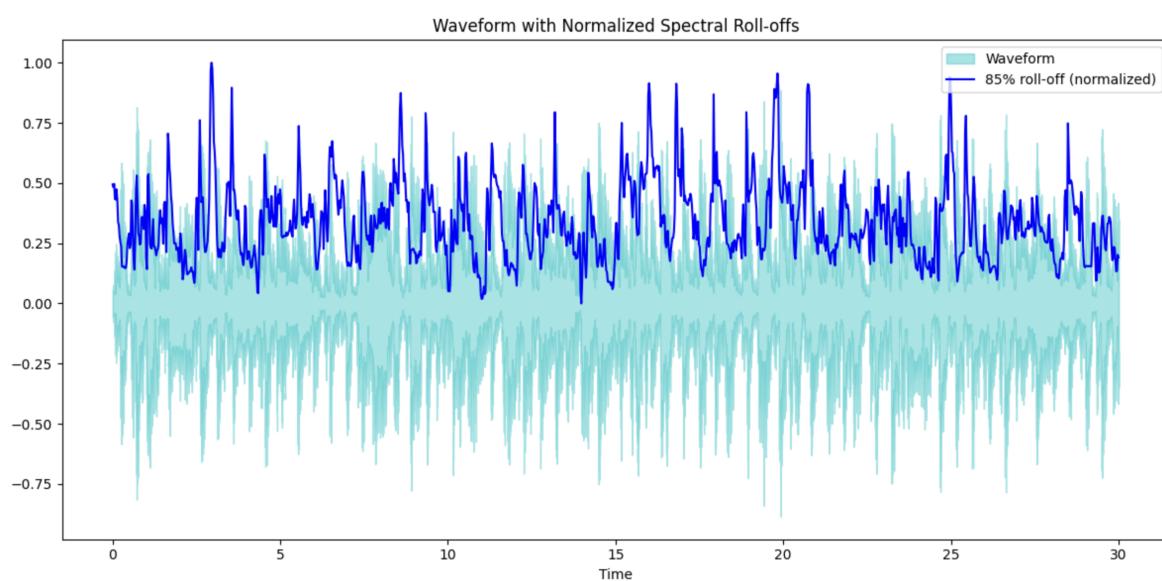
Nói cách khác, nếu bạn tổng hợp biên độ của các thành phần phẳng từ tần số thấp nhất lên tần số roll-off, nó sẽ chiếm, ví dụ, 85% tổng năng lượng phẳng của tín hiệu.

Để hình dung được Spectral roll-off các bạn chạy dòng lệnh sau

```

1 import sklearn.preprocessing
2
3 # Function to normalize an array
4 def normalize(x, axis=0):
5     return sklearn.preprocessing.minmax_scale(x, axis=axis)
6
7 # Load the audio file with default sampling rate of 22,050Hz
8 data, sr = librosa.load(data_path)
9
10 # Compute and norm the spectral roll-off
11 rolloff_85 = librosa.feature.spectral_rolloff(y=data, sr=sr, roll_percent=0.85)
12 [0]\n12 rolloff_85_norm = normalize(rolloff_85)
13
14 # Plot the waveform and normalized spectral roll-offs
15 plt.figure(figsize=(12, 6))
16 librosa.display.waveshow(data, sr=sr, alpha=0.4, label='Waveform')
17 times = librosa.times_like(rolloff_85)
18 plt.plot(times, rolloff_85_norm, color='r', label='85% roll-off (normalized)')
19
20 plt.title('Waveform with Normalized Spectral Roll-offs')
21 plt.legend(loc='upper right')
22 plt.tight_layout()
23 plt.show()

```



Hình 10: Spectral Roll-offs

Spectral roll-off cung cấp thông tin về hình dạng của phô. Một tần số roll-off thấp có thể chỉ ra một tín hiệu tập trung hơn vào tần số thấp hơn, trong khi tần số roll-off cao có thể gợi ý một phô rộng hoặc phẳng hơn.

Trong tín hiệu âm nhạc, tần số roll-off thấp có thể chỉ ra sự ưu thế của các harmonics thấp hơn, trong khi tần số roll-off cao có thể gợi ý sự hiện diện của các harmonics cao hơn hoặc tiếng ồn.

Spectral roll-off có thể giúp phân biệt giữa các nhạc cụ giàu harmonics và những nguyên âm cao hoặc âm thanh có nhiều bước sóng bất thường.

8. Spectral Centroid

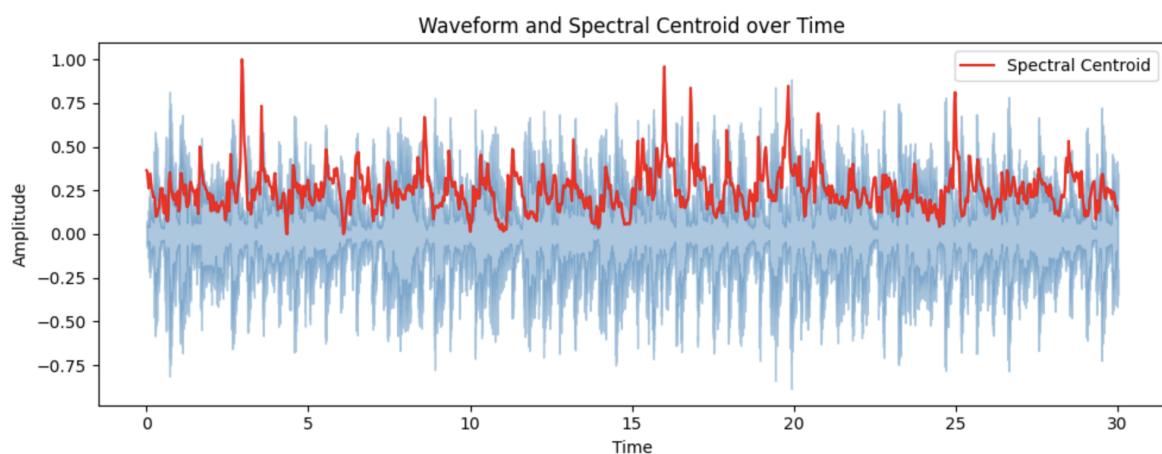
Spectral centroid là một đặc trưng được sử dụng trong xử lý tín hiệu kỹ thuật số cho thấy mức độ "sáng" hay "sắc nét" của âm thanh. Nó chỉ ra nơi "trọng tâm" phô của tín hiệu âm thanh nằm ở đâu.

Spectral centroid đại diện cho điểm cân đối của phô của âm thanh. Nếu hầu hết năng lượng của âm thanh tập trung vào các tần số thấp, Spectral centroid sẽ thấp, chỉ ra một âm thanh "tối" hoặc "đục". Ngược lại, nếu năng lượng tập trung vào các tần số cao, Spectral centroid sẽ cao, chỉ ra một âm thanh "sáng" hoặc "sắc nét".

Để hình dung được Spectral centroid các bạn chạy dòng lệnh sau

```

1 # Load the audio file with default sampling rate of 22,050Hz
2 data, sr = librosa.load(data_path)
3 # Compute the spectral centroid
4 spectral_centroids = librosa.feature.spectral_centroid(y=data, sr=sr)
5 # Plotting
6 plt.figure(figsize=(10, 4))
7 librosa.display.waveform(data, sr=sr, alpha=0.4)
8 frames = range(len(spectral_centroids[0]))
9 t = librosa.frames_to_time(frames)
10 plt.plot(t, normalize(spectral_centroids[0]), color='r', label='Spectral Centroid')
11 plt.title('Waveform and Spectral Centroid over Time')
12 plt.xlabel('Time')
13 plt.ylabel('Amplitude')
14 plt.legend(loc='upper right')
15 plt.tight_layout()
16 plt.show()
```



Hình 11: Spectral Centroid

Spectral centroid thường được sử dụng trong âm nhạc và phân tích âm thanh để mô tả timbre hay "sắc thái của âm thanh". Các nhạc cụ hoặc âm thanh có Spectral centroid cao thường được cảm nhận là sáng sủa, trong khi những cái có Spectral centroid thấp được cảm nhận là tối. Nó có thể được sử dụng để phân biệt các nhạc cụ, phân loại thể loại âm nhạc, hoặc thậm chí phát hiện tâm trạng của một bài hát.

Mặc dù Spectral centroid cung cấp thông tin quý báu về nội dung phổ của âm thanh, nó chỉ là một giá trị đơn và không thể hiện được toàn bộ độ phức tạp sắc thái của âm thanh. Thường được sử dụng kết hợp với các đặc trưng phổ khác để thực hiện phân tích toàn diện hơn.

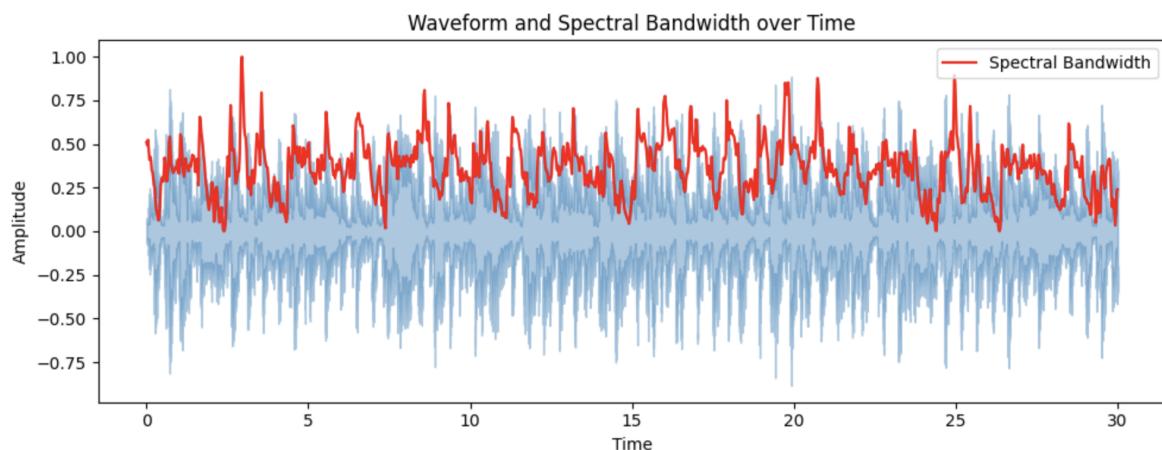
9. Spectral Bandwidth

Spectral bandwidth là một đặc trưng được sử dụng trong lĩnh vực xử lý tín hiệu và phân tích âm thanh, nó mô tả độ rộng phổ của một tín hiệu âm thanh. Nói một cách đơn giản, nó đo lường mức độ rộng của phân phối tần số trong âm thanh.

Để hình dung được Spectral bandwidth các bạn chạy dòng lệnh sau

```

1 # Let load .wav file with default sampling rate of 22,050Hz
2 data, sr = librosa.load(data_path)
3
4 # Compute the spectral bandwidth
5 spectral_bandwidth = librosa.feature.spectral_bandwidth(y=data, sr=sr)
6
7 # Plotting
8 plt.figure(figsize=(10, 4))
9 librosa.display.waveform(data, sr=sr, alpha=0.4)
10 frames = range(len(spectral_bandwidth[0]))
11 t = librosa.frames_to_time(frames)
12 plt.plot(t, normalize(spectral_bandwidth[0]), color='r', label='Spectral
    Bandwidth')
13
14 # Set labels and title
15 plt.title('Waveform and Spectral Bandwidth over Time')
16 plt.xlabel('Time')
17 plt.ylabel('Amplitude')
18 plt.legend(loc='upper right')
19 plt.tight_layout()
20 plt.show()
```



Hình 12: Spectral Bandwidth

Nếu một âm thanh có Spectral bandwidth nhỏ, điều đó có nghĩa là hầu hết năng lượng của nó tập trung vào một khoảng tần số cụ thể. Ví dụ, sóng hình sin thuần có băng thông rất hẹp vì năng lượng của nó tập trung vào một tần số duy nhất.

Một âm thanh có Spectral bandwidth lớn có năng lượng được phân bố rộng hơn trên một loạt tần số. Nhiều trống, ví dụ, có Spectral bandwidth rất rộng vì nó chứa tất cả các tần số với năng lượng bằng nhau.

Thông thường Spectral bandwidth thường đi kèm với Spectral centroid. Spectral centroid đại diện cho "trọng tâm" của phổ. Spectral bandwidth sau đó đo lường sự lan trải của phổ xung quanh trọng tâm này.

10. Chroma

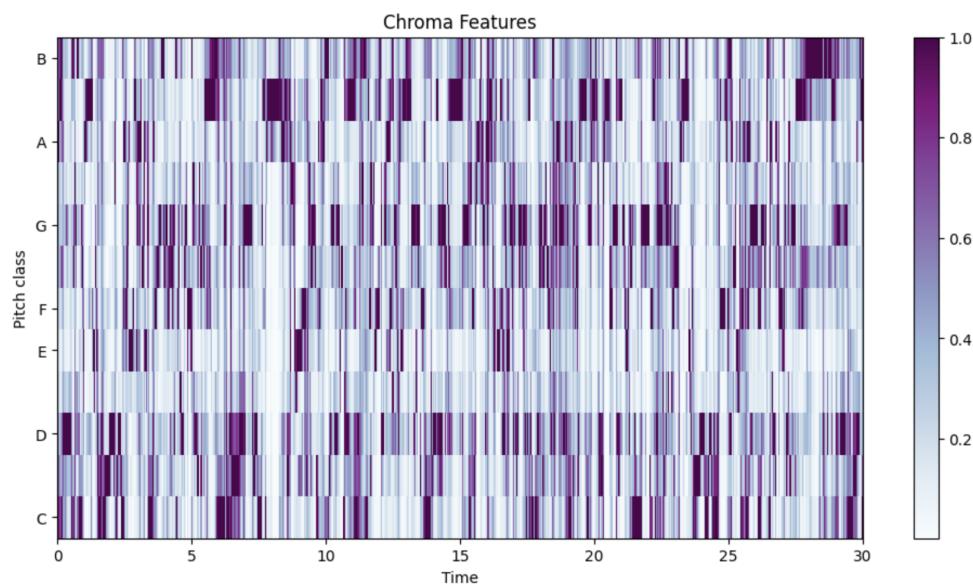
Đặc trưng Chroma là một công cụ mạnh mẽ trong xử lý tín hiệu âm thanh, đặc biệt trong bối cảnh âm nhạc. Chúng liên quan đến mười hai lớp tần số khác nhau và thường được sử dụng để mô tả âm hòa và hợp âm trong âm nhạc.

Nhạc lý phương Tây được chia thành mười hai lớp tần số, tương ứng với mươi hai nốt trong một quãng (ví dụ, C, C#, D, D#, ..., B). Đặc trưng Chroma bắt lượng năng lượng trong mỗi một trong những lớp tần số này, không phân biệt quãng cụ thể. Điều này làm cho đặc trưng Chroma ít nhạy cảm đối với nốt cụ thể và tập trung hơn vào nội dung hòa âm.

Để hình dung được Chroma các bạn chạy dòng lệnh sau

```

1 # Let load .wav file with default sampling rate of 22,050Hz
2 data, sr = librosa.load(data_path)
3
4 chroma = librosa.feature.chroma_stft(y=data, sr=sr)
5 plt.figure(figsize=(12,6))
6 librosa.display.specshow(chroma, sr=sr, x_axis="time", y_axis="chroma", cmap="BuPu")
7 plt.colorbar()
8 plt.title("Chroma Features")
9 plt.show()
```



Hình 13: Chroma

Bởi vì đặc trưng Chroma nắm bắt được nội dung hòa âm của âm nhạc, chúng thường được sử dụng trong các thuật toán để tự động nhận diện hợp âm, tìm các phân đoạn có sự tiến triển hòa âm tương tự, trích xuất bài hát hoặc xác định các bản phối.

Đặc trưng Chroma thường được tạo ra từ Biến đổi Fourier ngắn thời gian (STFT) của tín hiệu. Cường độ của STFT được ánh xạ vào mười hai lớp tần số, thường sử dụng một kỹ thuật gọi là "bin folding." Quá trình này bao gồm việc lấy toàn bộ năng lượng từ một nốt cụ thể trong tất cả các quãng và tổng hợp chúng vào một trong mười hai bin Chroma.

Tóm lại, đặc trưng Chroma cung cấp một biểu diễn ngắn gọn về nội dung hòa âm trong âm nhạc, trừu tượng hóa khỏi các khía cạnh như timbre và nhịp, và tập trung vào phân phối năng lượng của các lớp tần số. Chúng là một phần quan trọng trong nhiều nhiệm vụ truy xuất thông tin âm nhạc nhờ khả năng bắt lấy bản chất hòa âm trong âm nhạc.

11. Harmonic/Percussive Source Separation (HPSS)

- **Thành phần Hài hòa (Harmonic Components):**

Đây là các thành phần của tín hiệu thay đổi chậm theo thời gian. Chúng tương ứng với giai điệu và hòa âm trong âm nhạc. Thành phần hài hòa thường được tạo ra bởi các nhạc cụ sản xuất âm thanh kéo dài, chẳng hạn như dây (như violin hoặc guitar), tiết tấu (như sáo), và nhạc cụ đồng.

Trong miền tần số, thành phần hài hòa xuất hiện dưới dạng các đỉnh rõ ràng, chúng chủ yếu là tần số cơ bản của nốt nhạc đang được chơi và các âm phụ của nó (bội số nguyên của tần số cơ bản).

- **Thành phần Gõ (Percussive Components):**

Thành phần gõ của một tín hiệu thường là tạm thời và thay đổi nhanh chóng theo thời gian. Chúng tương ứng với nhịp điệu trong âm nhạc. Chúng được tạo ra bởi các nhạc cụ gõ như trống, tambourine, và conga. Nhưng không phải tất cả âm thanh gõ đều đến từ các nhạc cụ gõ truyền thống; việc gảy dây của một cây guitar hoặc sự tấn công của một nút piano cũng có thể có yếu tố gõ mạnh.

Trong miền tần số, thành phần gõ phân bố qua một loạt các tần số mà không có đỉnh rõ ràng. Chúng xuất hiện như những cơn nổ đột ngột trong biểu diễn miền thời gian.

- **Tách nguồn Hài hòa/Gõ (HPSS):**

Đây là một khái niệm quan trọng trong MIR. Đó là một kỹ thuật được sử dụng để tách các thành phần hài hòa và gõ của tín hiệu âm nhạc. Động lực cho sự tách biệt này bao gồm các ứng dụng như:

- Điều chỉnh các yếu tố nhịp điệu so với yếu tố giai điệu một cách độc lập.
- Đơn giản hóa công việc phiên âm các nhạc cụ nhịp điệu riêng biệt từ các nhạc cụ giai điệu.
- Cho phép nghiên cứu nhịp điệu và giai điệu một cách riêng biệt.

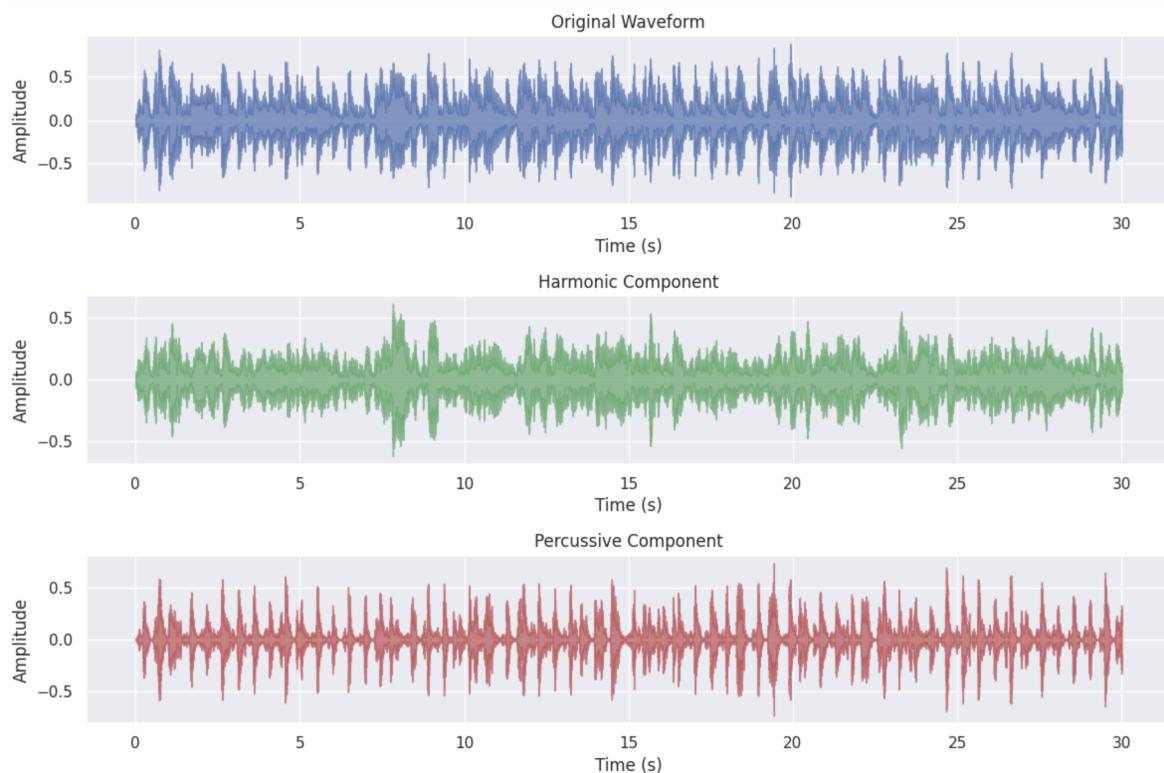
HPSS thường được thực hiện bằng các phương pháp lọc trung bình trong miền thời gian-tần số. Một phương pháp, chẳng hạn, bao gồm việc lấy biểu đồ phổ của một tín hiệu và áp dụng các bộ lọc trung bình dọc theo thời gian (để bảo tồn hài hòa) và tần số (để bảo tồn gõ). Điều này cho phép tách biệt các thành phần hài hòa và gõ dựa trên đặc tính tự nhiên của chúng trong biểu diễn thời gian-tần số.

Để hình dung được HPSS các bạn chạy dòng lệnh sau

```

1 # Load .wav file with default sampling rate of 22,050Hz
2 data, sr = librosa.load(data_path)
3 # Separate the harmonic and percussive components
4 data_harmonic, data_percussive = librosa.effects.hpss(data)
5
6 # Plot the original, harmonic, and percussive waveforms
7 plt.figure(figsize=(12, 8))
8 # Original waveform
9 plt.subplot(3, 1, 1)
10 librosa.display.waveshow(data, sr=sr, alpha=0.7)
11 plt.title('Original Waveform')
12 plt.xlabel('Time (s)')
13 plt.ylabel('Amplitude')
14 # Harmonic component
15 plt.subplot(3, 1, 2)
16 librosa.display.waveshow(data_harmonic, sr=sr, alpha=0.7, color='g')
17 plt.title('Harmonic Component')
18 plt.xlabel('Time (s)')
19 plt.ylabel('Amplitude')
20 # Percussive component
21 plt.subplot(3, 1, 3)
22 librosa.display.waveshow(data_percussive, sr=sr, alpha=0.7, color='r')
23 plt.title('Percussive Component')
24 plt.xlabel('Time (s)')
25 plt.ylabel('Amplitude')
26 plt.tight_layout()
27 plt.show()

```



Hình 14: Harmonic/Percussive Source Separation

12. Hệ số Cepstral Tần số Mel (MFCC)

MFCC là một tập các đặc trưng thường được sử dụng trong xử lý tín hiệu âm thanh, đặc biệt trong ngữ cảnh của nhận dạng giọng nói và tiếng nói. Chúng cung cấp một biểu diễn gọn gàng của phổ công suất ngắn hạn của âm thanh và dựa trên nhận thức về thính giác của con người.

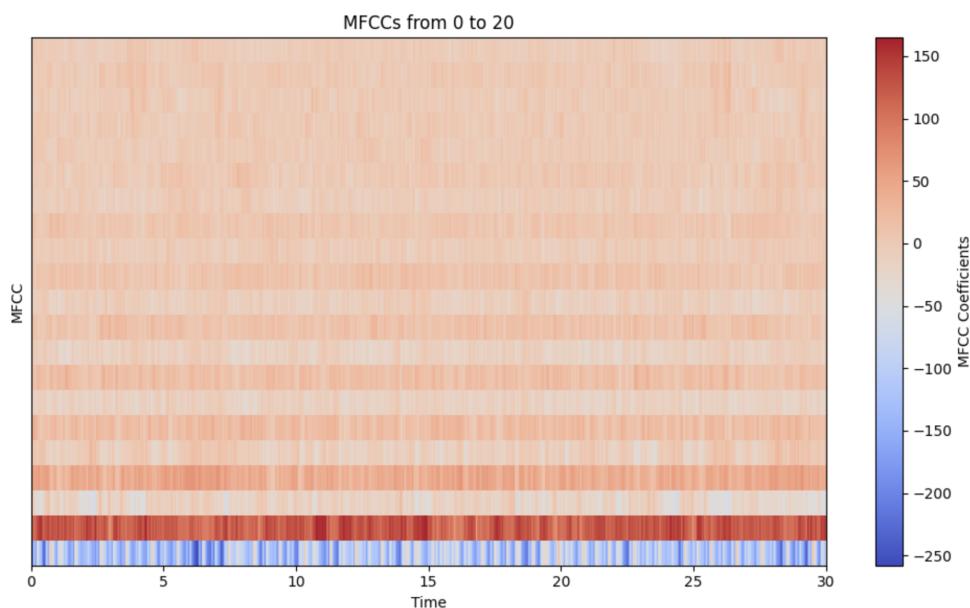
MFCC là một trong những đặc trưng phổ biến nhất được sử dụng trong sinh trắc học giọng nói để xác thực người dùng dựa trên đặc điểm giọng nói đặc đáo của họ. Chúng có thể nắm bắt được nội dung ngữ âm của tín hiệu giọng nói.

Để hình dung được MFCC các bạn chạy dòng lệnh sau

```

1 # Let load .wav file with default sampling rate of 22,050Hz
2 data, sr = librosa.load(data_path)
3
4 # Compute the MFCCs
5 mfccs = librosa.feature.mfcc(y=data, sr=sr, n_mfcc=21) # Compute 21 MFCCs to
   include 0 to 20
6
7 # Visualize the MFCCs
8 plt.figure(figsize=(10, 6))
9 librosa.display.specshow(mfccs, x_axis='time', sr=sr)
10 plt.colorbar(label='MFCC Coefficients')
11 plt.ylabel('MFCC')
12 plt.title('MFCCs from 0 to 20')
13 plt.tight_layout()
14 plt.show()

```



Hình 15: Hệ số Cepstral Tần số Mel

Mặc dù phổ biến hơn trong xử lý giọng nói, MFCC cũng được sử dụng trong một số nhiệm vụ trích xuất thông tin âm nhạc, như phân loại thể loại hoặc nhận dạng nhạc cụ.

Tóm lại, MFCC cung cấp một biểu diễn của hình dạng phổ ngắn hạn của âm thanh. Chúng được thiết kế để mô phỏng hệ thống thính giác của con người, làm cho chúng đặc biệt hiệu quả cho các nhiệm vụ liên quan đến giọng nói và tiếng nói của con người.

Phần III: Music Genre Classification - Project

Các bạn có thể tham khảo Code chi tiết tại link [Colab](#).

1. Trích xuất đặc trưng âm thanh - Tái tạo bộ dữ liệu GTZAN

Bộ dữ liệu GTZAN tách mỗi bản nhạc 30 giây thành 10 phân đoạn 3 giây, điều này giúp số lượng data của chúng ta tăng từ 1.000 bài nhạc lên 10.000 đoạn nhạc. Tập **feature_3_sec.csv** của bộ dữ liệu này chứa mean và variant của toàn bộ những thuộc tính mà chúng ta đã tìm hiểu ở trên cho mỗi đoạn.

Để tái tạo lại tập CSV này các bạn làm theo những bước sau.

```

1 from glob import glob
2 import pandas as pd
3
4 num_segment=10
5 num_mfcc=20
6 sample_rate=22050
7 n_fft=2048
8 hop_length=512
9
10 my_csv={"filename":[] , "chroma_stft_mean": [] , "chroma_stft_var": [] ,
11     "rms_mean": [] , "rms_var": [] , "spectral_centroid_mean": [] ,
12     "spectral_centroid_var": [] , "spectral_bandwidth_mean": [] ,
13     "spectral_bandwidth_var": [] , "rolloff_mean": [] ,
14     "rolloff_var": [] , "zero_crossing_rate_mean": [] ,
15     "zero_crossing_rate_var": [] , "harmony_mean": [] ,
16     "harmony_var": [] , "perceptr_mean": [] ,
17     "perceptr_var": [] , "tempo": [] , "mfcc1_mean": [] ,
18     "mfcc1_var": [] , "mfcc2_mean": [] , "mfcc2_var": [] ,
19     "mfcc3_mean": [] , "mfcc3_var": [] , "mfcc4_mean": [] ,
20     "mfcc4_var": [] , "mfcc5_mean": [] , "mfcc5_var": [] ,
21     "mfcc6_mean": [] , "mfcc6_var": [] , "mfcc7_mean": [] ,
22     "mfcc7_var": [] , "mfcc8_mean": [] , "mfcc8_var": [] ,
23     "mfcc9_mean": [] , "mfcc9_var": [] , "mfcc10_mean": [] ,
24     "mfcc10_var": [] , "mfcc11_mean": [] , "mfcc11_var": [] ,
25     "mfcc12_mean": [] , "mfcc12_var": [] , "mfcc13_mean": [] ,
26     "mfcc13_var": [] , "mfcc14_mean": [] , "mfcc14_var": [] ,
27     "mfcc15_mean": [] , "mfcc15_var": [] , "mfcc16_mean": [] ,
28     "mfcc16_var": [] , "mfcc17_mean": [] , "mfcc17_var": [] ,
29     "mfcc18_mean": [] , "mfcc18_var": [] , "mfcc19_mean": [] ,
30     "mfcc19_var": [] , "mfcc20_mean": [] , "mfcc20_var":[] ,
31     "label":[]}

```

Trong đó:

- **num_segment** là số đoạn chia nhỏ của bài hát
- **num_mfcc** là số lượng Hệ số MFCC cần tính toán, trong bộ GTZAN là 20
- **sample_rate, n_fft** và **hop_length** là các giá trị mặc định của thư viện librosa
- **my_csv** là một dictionary chứa những thuộc tính có trong tập GTZAN mà chúng ta cần tính toán lại.

Tính toán mean và var của các thuộc tính và chèn vào my_csv

```

1 dataset_path="/content/GTZAN/genres_original"
2 audio_files = glob(dataset_path + "/*/*")
3
4 samples_per_segment = int(sample_rate*30/num_segment)
5
6 genre=""
7 for f in sorted(audio_files):
8     if genre!=f.split('/')[-2]:
9         genre=f.split('/')[-2]
10        print("Processing " + genre + "...")
11    fname=f.split('/')[-1]
12    try:
13        y, sr = librosa.load(f, sr=sample_rate)
14    except:
15        continue
16
17    for n in range(num_segment):
18        y_seg = y[samples_per_segment*n: samples_per_segment*(n+1)]
19        #Chromagram
20        chroma_hop_length = 512
21        chromagram = librosa.feature.chroma_stft(y=y_seg, sr=sample_rate,
22        hop_length=chroma_hop_length)
23        my_csv["chroma_stft_mean"].append(chromagram.mean())
24        my_csv["chroma_stft_var"].append(chromagram.var())
25
26        #Root Mean Square Energy
27        RMSEn= librosa.feature.rms(y=y_seg)
28        my_csv["rms_mean"].append(RMSEn.mean())
29        my_csv["rms_var"].append(RMSEn.var())
30
31        #Spectral Centroid
32        spec_cent=librosa.feature.spectral_centroid(y=y_seg)
33        my_csv["spectral_centroid_mean"].append(spec_cent.mean())
34        my_csv["spectral_centroid_var"].append(spec_cent.var())
35
36        #Spectral Bandwith
37        spec_band=librosa.feature.spectral_bandwidth(y=y_seg,sr=sample_rate)
38        my_csv["spectral_bandwidth_mean"].append(spec_band.mean())
39        my_csv["spectral_bandwidth_var"].append(spec_band.var())
40
41        #Rolloff
42        spec_roll=librosa.feature.spectral_rolloff(y=y_seg,sr=sample_rate)
43        my_csv["rolloff_mean"].append(spec_roll.mean())
44        my_csv["rolloff_var"].append(spec_roll.var())
45
46        #Zero Crossing Rate
47        zero_crossing=librosa.feature.zero_crossing_rate(y=y_seg)
48        my_csv["zero_crossing_rate_mean"].append(zero_crossing.mean())
49        my_csv["zero_crossing_rate_var"].append(zero_crossing.var())
50
51        #Harmonics and Percussive
52        harmony, percep = librosa.effects.hpss(y=y_seg)
53        my_csv["harmony_mean"].append(harmony.mean())
54        my_csv["harmony_var"].append(harmony.var())
55        my_csv["percep_mean"].append(percep.mean())
56        my_csv["percep_var"].append(percep.var())
57
58        #Tempo
59        tempo, _ = librosa.beat.beat_track(y=y_seg, sr=sample_rate)

```

```

59     my_csv["tempo"].append(tempo)
60
61     #MFCC
62     mfcc=librosa.feature.mfcc(y=y_seg,sr=sample_rate, n_mfcc=num_mfcc, n_fft=
63     n_fft, hop_length=hop_length)
64     mfcc=mfcc.T
65
66     fseg_name='.'.join(fname.split('.')[2:])+f'{n}.wav'
67     my_csv["filename"].append(fseg_name)
68     my_csv["label"].append(genre)
69     for x in range(20):
70         feat1 = "mfcc" + str(x+1) + "_mean"
71         feat2 = "mfcc" + str(x+1) + "_var"
72         my_csv[feat1].append(mfcc[:,x].mean())
73         my_csv[feat2].append(mfcc[:,x].var())
74     print(fname)
75
76 df = pd.DataFrame(my_csv)
77 df.to_csv('/content/GTZAN/features_3_sec.csv', index=False)

```

2. Music Genre Classification - Machine Learning Solution

Tiến hành đọc file feature_3_sec.csv vừa tạo bằng thư viện Pandas

```

1 # Reading the csv file
2 df = pd.read_csv("/content/GTZAN/features_3_sec.csv")
3 df.head()

```

Chúng ta sẽ xoá cột "filename" vì nó không cần thiết trong quá trình huấn luyện

```

1 # Drop the column filename as it is no longer required for training
2 df=df.drop(labels="filename",axis=1)

```

Tiếp theo chúng ta sẽ tách data và label, đồng thời tiến hành scale data và label encoding

```

1 X, y = df.iloc[:, :-1], df.iloc[:, -1]
2
3 # Label Encoding
4
5 # Blues - 0
6 # Classical - 1
7 # Country - 2
8 # Disco - 3
9 # Hip-hop - 4
10 # Jazz - 5
11 # Metal - 6
12 # Pop - 7
13 # Reggae - 8
14 # Rock - 9
15
16 encoder=LabelEncoder()
17 y=encoder.fit_transform(y)
18
19 # Data scaling
20 from sklearn.preprocessing import StandardScaler
21 scaler=StandardScaler()
22 X=scaler.fit_transform(X)

```

Chia bộ data-label thành hai tập train và test

```

1 from sklearn.model_selection import train_test_split
2
3 # splitting 70% data into training set and the remaining 30% to test set
4 X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.3,
5                                              random_state=1234)

```

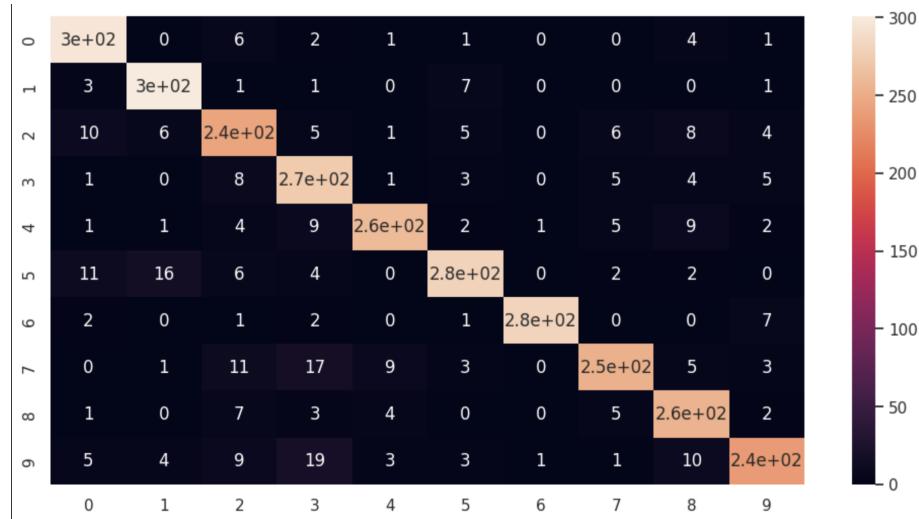
(a) KNN Classifier

```

1 from sklearn.neighbors import KNeighborsClassifier
2 from sklearn.metrics import classification_report, confusion_matrix
3 import matplotlib.pyplot as plt
4 import seaborn as sns
5
6 knn_cls=KNeighborsClassifier(n_neighbors=3)
7 knn_cls.fit(X_train,y_train)
8 y_pred=knn_cls.predict(X_test)
9
10 print("Training set score: {:.3f}".format(knn_cls.score(X_train, y_train)))
11 print("Test set score: {:.3f}".format(knn_cls.score(X_test, y_test)))
12
13 cf_matrix = confusion_matrix(y_test, y_pred)
14 sns.set(rc = {'figure.figsize':(12,6)})
15 sns.heatmap(cf_matrix, annot=True)
16 print(classification_report(y_test,y_pred))

```

	precision	recall	f1-score	support
0	0.90	0.95	0.92	312
1	0.91	0.96	0.94	314
2	0.82	0.84	0.83	288
3	0.81	0.91	0.86	300
4	0.93	0.88	0.91	295
5	0.92	0.87	0.89	322
6	0.99	0.96	0.97	295
7	0.91	0.84	0.87	302
8	0.86	0.92	0.89	277
9	0.90	0.81	0.86	292
accuracy			0.90	2997
macro avg	0.90	0.89	0.89	2997
weighted avg	0.90	0.90	0.90	2997



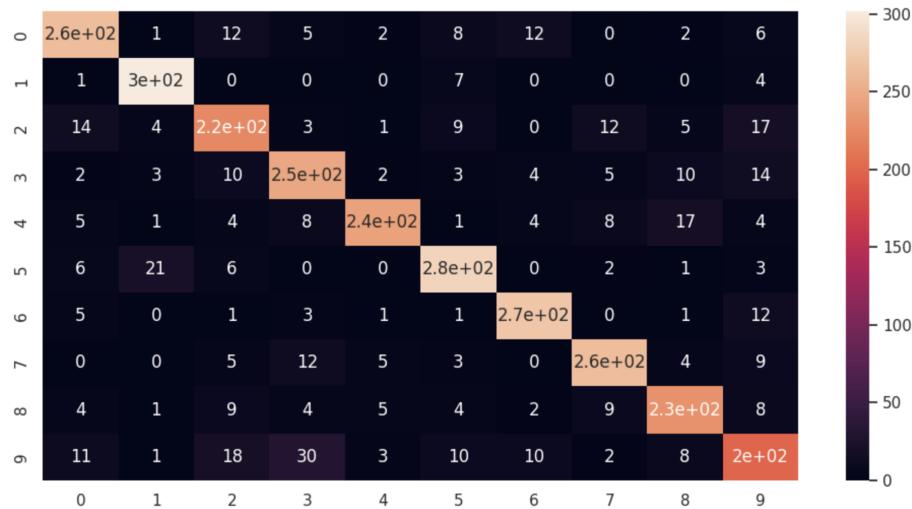
(b) SVM Classifier

```

1 from sklearn.svm import SVC
2
3 svm_cls = SVC(kernel='rbf', degree=8)
4 svm_cls.fit(X_train, y_train)
5
6 print("Training set score: {:.3f}".format(svm_cls.score(X_train, y_train)))
7 print("Test set score: {:.3f}".format(svm_cls.score(X_test, y_test)))
8
9 y_pred = svm_cls.predict(X_test)
10
11 cf_matrix3 = confusion_matrix(y_test, y_pred)
12 sns.set(rc = {'figure.figsize':(12,6)})
13 sns.heatmap(cf_matrix3, annot=True)
14 print(classification_report(y_test, y_pred))

```

Training set score: 0.920					
Test set score: 0.843					
	precision	recall	f1-score	support	
0	0.85	0.85	0.85	312	
1	0.90	0.96	0.93	314	
2	0.77	0.77	0.77	288	
3	0.79	0.82	0.81	300	
4	0.93	0.82	0.87	295	
5	0.86	0.88	0.87	322	
6	0.89	0.92	0.91	295	
7	0.87	0.87	0.87	302	
8	0.83	0.83	0.83	277	
9	0.72	0.68	0.70	292	
accuracy			0.84	2997	
macro avg	0.84	0.84	0.84	2997	
weighted avg	0.84	0.84	0.84	2997	



3. Music Genre Classification - Simple Feed Forward Neural Network

Chúng ta sẽ dùng Pytorch để dựng mô hình, tiến hành import những thư viện sau

```

1 import os
2 import numpy as np
3
4 import torch
5 from torch import nn, optim
6 from torch.functional import F
7 from torch.utils.data import DataLoader, TensorDataset

```

Khởi tạo một mô hình Multilayer Perceptron đơn giản

```

1 class MLP(nn.Module):
2     def __init__(self, input_size):
3         super(MLP, self).__init__()
4         self.flatten = nn.Flatten()
5         self.fc1 = nn.Linear(input_size, 512)
6         self.fc2 = nn.Linear(512, 256)
7         self.fc3 = nn.Linear(256, 128)
8         self.fc4 = nn.Linear(128, 64)
9         self.fc5 = nn.Linear(64, 32)
10        self.fc6 = nn.Linear(32, 10)
11        self.dropout = nn.Dropout(0.2)
12
13    def forward(self, x):
14        x = self.flatten(x)
15        x = F.relu(self.fc1(x))
16        x = self.dropout(x)
17        x = F.relu(self.fc2(x))
18        x = self.dropout(x)
19        x = F.relu(self.fc3(x))
20        x = self.dropout(x)
21        x = F.relu(self.fc4(x))
22        x = self.dropout(x)
23        x = F.relu(self.fc5(x))
24        x = self.dropout(x)
25        x = F.softmax(self.fc6(x), dim=1)
26        return x

```

```

1 input_size = X_train.shape[1]
2 model = MLP(input_size)

```

Chúng ta sẽ dùng Cross Entropy làm hàm loss với Optimizer Adam

```

1 criterion = nn.CrossEntropyLoss()
2 optimizer = optim.Adam(model.parameters(), lr=0.000146)

```

Training loop

```

1 num_epochs = 300
2 batch_size = 256
3
4 train_dataset = TensorDataset(torch.tensor(X_train), torch.tensor(y_train))
5 train_loader = DataLoader(train_dataset, batch_size=batch_size, shuffle=True)
6
7 val_dataset = TensorDataset(torch.tensor(X_test), torch.tensor(y_test))
8 val_loader = DataLoader(val_dataset, batch_size=batch_size, shuffle=False)
9
10 step = 0
11
12 for epoch in range(num_epochs):
13     model.train()
14     for inputs, labels in train_loader:
15         optimizer.zero_grad()
16         outputs = model(inputs.float())
17         loss = criterion(outputs, labels)
18         loss.backward()
19         optimizer.step()
20
21     if step % 100 ==0:
22         print(f"Step {step}, Train Loss: {loss.item():.4f}")
23     step += 1
24
25     # Validation
26     model.eval()
27     val_loss = 0.0
28     correct = 0
29     total = 0
30     with torch.no_grad():
31         for inputs, labels in val_loader:
32             outputs = model(inputs.float())
33             loss = criterion(outputs, labels)
34             val_loss += loss.item()
35             _, predicted = outputs.max(1)
36             total += labels.size(0)
37             correct += predicted.eq(labels).sum().item()
38
39     val_loss /= len(val_loader)
40     val_accuracy = 100 * correct / total
41     print(f"Epoch {epoch+1}/{num_epochs}, Validation Loss: {val_loss:.4f},
42           Validation Accuracy: {val_accuracy:.2f}%")

```

Kiểm tra kết quả mô hình

```

1 # Sample testing
2 model.eval()
3 with torch.no_grad():
4     predictions = model(torch.tensor(X_test).float())
5     _, predicted_indices = predictions.max(1)
6     print("Expected Index: {}, Predicted Index: {}".format(y_test,
predicted_indices.numpy()))

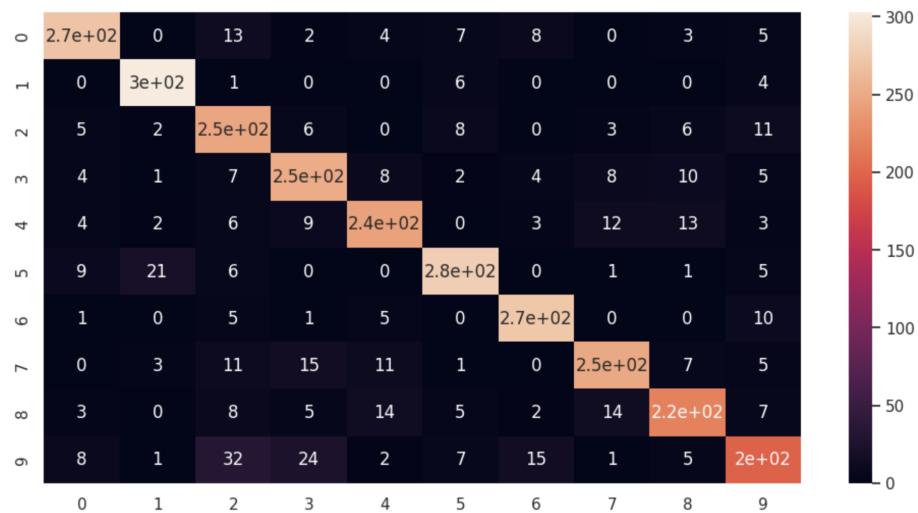
```

```

7
8 # Confusion Matrix and Classification Report
9 from sklearn.metrics import classification_report
10 import seaborn as sns
11
12 y_pred = predicted_indices.numpy()
13 cf_matrix = confusion_matrix(y_test, y_pred)
14 sns.set(rc={'figure.figsize':(12,6)})
15 sns.heatmap(cf_matrix, annot=True)
16 print(classification_report(y_test, y_pred))

```

	precision	recall	f1-score	support
0	0.89	0.87	0.88	312
1	0.91	0.96	0.94	314
2	0.74	0.86	0.79	288
3	0.80	0.84	0.82	300
4	0.85	0.82	0.84	295
5	0.89	0.87	0.88	322
6	0.90	0.93	0.91	295
7	0.86	0.82	0.84	302
8	0.83	0.79	0.81	277
9	0.78	0.67	0.72	292
accuracy			0.84	2997
macro avg		0.84	0.84	2997
weighted avg		0.85	0.84	2997



Phần VI: Câu hỏi trắc nghiệm

- 1) Biểu đồ sóng (Waveform) biểu diễn điều gì trong âm thanh?
 - A. Biểu diễn tần số của âm thanh qua thời gian
 - B. Biểu diễn độ dài của tín hiệu âm thanh
 - C. Biểu diễn sự biến đổi biên độ của sóng âm so với thời gian
 - D. Biểu diễn sự va chạm của các hạt không khí khi tạo ra sóng âm
- 2) Vì sao tốc độ lấy mẫu là một tham số quan trọng?
 - A. Bởi vì nó biểu thị độ dài của tệp âm thanh
 - B. Bởi vì nó quyết định số lượng mẫu được lấy mỗi giây từ tín hiệu liên tục, ảnh hưởng đến chất lượng tín hiệu số và kích thước dữ liệu
 - C. Bởi vì nó giúp định dạng tệp âm thanh
 - D. Bởi vì nó ảnh hưởng đến tần số của âm thanh
- 3) Lợi ích chính của biểu đồ phổ so với biểu đồ sóng là gì?
 - A. Nó cung cấp thông tin chi tiết về cách các tần số khác nhau phân phối theo thời gian.
 - B. Nó cho phép hiển thị cấu trúc thời gian của âm thanh.
 - C. Nó giúp biểu diễn động lực thời gian và nhịp điệu của âm nhạc.
 - D. Nó đại diện cho sự biến đổi biên độ của tín hiệu âm thanh theo thời gian.
- 4) Điểm khác biệt chính giữa biểu đồ phổ và phổ Mel là gì?
 - A. Biểu đồ phổ cung cấp một biểu diễn trực quan về nội dung tần số của tín hiệu âm thanh theo thời gian, trong khi phổ Mel biểu diễn âm lượng của tín hiệu âm thanh theo thời gian.
 - B. Phổ Mel chỉ được sử dụng cho phân tích giọng nói, trong khi biểu đồ phổ có thể được sử dụng cho bất kỳ tín hiệu âm thanh nào.
 - C. Biểu đồ phổ cung cấp một biểu diễn bị méo của tín hiệu âm thanh, trong khi phổ Mel sửa chữa những méo này.
 - D. Phổ Mel nhấn mạnh các dải tần số quan trọng nhất cho việc nghe và hiểu của con người dựa trên thang Mel, trong khi biểu đồ phổ cung cấp biểu diễn tần số tuyến tính.
- 5) Trong ngữ cảnh xử lý âm thanh, tại sao RMS-E (Root Mean Square Energy) lại quan trọng?
 - A. RMS-E cho phép đo độ cao của âm thanh.
 - B. RMS-E được sử dụng để phát hiện các đỉnh trong tín hiệu âm thanh.
 - C. RMS-E đo "âm lượng cảm nhận" và thường được sử dụng trong các thuật toán chuẩn hóa và nén âm thanh.
 - D. RMS-E được sử dụng để tối ưu hóa tốc độ lấy mẫu của tín hiệu âm thanh.
- 6) Trong ngữ cảnh xử lý giọng nói, tần số cắt không (Zero-Crossing Rate) được sử dụng như thế nào?
 - A. Để đo độ to lớn của âm thanh.
 - B. Để phân biệt giữa các đoạn có âm hữu thanh và âm vô thanh.
 - C. Để nhận dạng các từ riêng biệt trong một câu.
 - D. Để phân tích giai điệu của một bài hát.

- 7) Trong ngữ cảnh xử lý tín hiệu âm nhạc, Spectral roll-off được sử dụng để làm gì?
- Để xác định tần số cơ bản của tín hiệu.
 - Để đo độ lớn của âm thanh.
 - Để cung cấp thông tin về hình dạng phổ và phân biệt giữa các nhạc cụ và âm thanh có khác biệt về harmonics.
 - Để xác định tốc độ của tín hiệu âm thanh.
- 8) Spectral centroid trong xử lý tín hiệu âm thanh thường liên quan đến yếu tố nào?
- "Sắc thái" hoặc timbre của âm thanh, chỉ ra mức độ "sáng" hay "sắc nét" của âm thanh.
 - Độ to lớn hoặc độ ồn của âm thanh.
 - Tần số cơ bản của tín hiệu.
 - Độ cao tương đối của âm thanh so với âm thanh khác.
- 9) Đặc trưng Spectral bandwidth trong xử lý tín hiệu âm thanh thường mô tả yếu tố gì?
- Tần số cơ bản của âm thanh.
 - "Sắc thái" hoặc timbre của âm thanh.
 - Tổng năng lượng phổ của âm thanh.
 - Độ rộng của phân phối tần số trong âm thanh.
- 10) Đặc trưng Chroma trong xử lý tín hiệu âm thanh thường được sử dụng để mô tả yếu tố gì trong âm nhạc?
- Nhịp điệu và tốc độ của âm nhạc.
 - Nội dung hòa âm và hợp âm của âm nhạc.
 - Độ sáng hoặc tối của timbre trong âm nhạc.
 - Tổng cường độ âm thanh.
- 11) Tại sao việc tách thành phần Hài hòa và Gõ (HPSS) quan trọng trong MIR?
- Để chỉnh sửa tín hiệu và làm cho nó nghe to hơn.
 - Để thay đổi tần số của tín hiệu.
 - Để điều chỉnh nhịp điệu so với yếu tố giai điệu một cách độc lập và nghiên cứu chúng một cách riêng biệt.
 - Để làm giảm tiếng ồn từ tín hiệu âm nhạc.
- 12) Trong bối cảnh nào MFCC thường được sử dụng nhiều nhất?
- Nhận dạng giọng nói và xác thực người dùng dựa trên đặc điểm giọng nói của họ.
 - Điều chỉnh độ lớn của âm thanh.
 - Để điều chỉnh nhịp điệu so với yếu tố giai điệu một cách độc lập và nghiên cứu chúng một cách riêng biệt.
 - Phát hiện nhiễu trong bản ghi âm.