**ĐẠI HỌC BÁCH KHOA HÀ NỘI**

**TRƯỜNG CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG**


# ĐỒ ÁN II

---

# ĐỀ TÀI :

# PHÂN LOẠI DÒNG NHẠC CỦA BÀI HÁT

---

Giảng viên hướng dẫn: TS. Đỗ Tiến Dũng

Sinh viên: Nguyễn Trần Hiếu Giang 20191804

HÀ NỘI, 06/2023

# MỤC LỤC

**1. Tổng quan** ..... 2
**2. Các nghiên cứu liên quan** ..... 3
2.1. K-nearest neighbor (KNN) ..... 3
2.2. Mel-frequency cepstral coefficients (MFCC) ..... 4
**3. Dữ liệu âm thanh** ..... 5
**4. Thiết kế thực hiện** ..... 6
4.1. Trích xuất thuộc tính file âm thanh (Feature Extraction) ..... 6
4.2. Model Training ..... 6
4.3. Phân loại file âm thanh ..... 6
**5. Thực nghiệm** ..... 7
5.1. Thuật toán K-nearest neighbor (KNN) ..... 7
5.1.1. Hàm tính khoảng cách giữa các vectơ thuộc tính ..... 7
5.1.2. Hàm xác định các hàng xóm (neighbor) ..... 8
5.1.3. Hàm xác định các hàng xóm gần nhất (nearest neighbor) ..... 8
5.1.4. Hàm tính độ chính xác của KNN ..... 8
5.2. Thuật toán thu thập dữ liệu và train dataset ..... 9
5.2.1. Hàm trích xuất thuộc tính của bài hát ..... 9
5.2.2. Hàm khởi tạo train data và test data ..... 10
5.2.3. Hàm đánh giá độ chính xác ..... 10
5.3. Thử nghiệm thực tế ..... 11
5.3.1. Hàm phân loại dựa trên data đã train ..... 11
5.3.2. Đưa ra kết quả dự đoán ..... 11
**6. Kết luận** ..... 13

# 1. Tổng quan

Phân loại thể loại âm nhạc là nhiệm vụ gán thể loại cho một bản nhạc. Đây có thể là một nhiệm vụ đầy thách thức, vì không có thuộc tính riêng lẻ nào có thể xác định duy nhất một thể loại cụ thể. Thay vào đó, các mô hình học máy thường được sử dụng để phân loại các thể loại âm nhạc dựa trên nhiều tính năng, chẳng hạn như nhịp độ, nhịp điệu và cao độ của âm nhạc.

Một trong những mô hình học máy phổ biến nhất để phân loại thể loại âm nhạc là K-nearest neighbors (KNN). Hiệu suất của KNN có thể được cải thiện bằng cách sử dụng nhiều kỹ thuật, chẳng hạn như lựa chọn tính năng, giảm kích thước và điều chỉnh siêu tham số. Dự án này sẽ sử dụng KNN để phân loại các thể loại âm nhạc bằng cách sử dụng bộ dữ liệu gồm 1.000 mẫu âm thanh.

Có một số thách thức liên quan đến phân loại thể loại âm nhạc. Một thách thức là không có một thuộc tính nào có thể xác định duy nhất một thể loại cụ thể. Ví dụ: một bài hát có thể có tiết tấu nhanh và cao độ, điều này có thể biểu thị cho nhạc pop hoặc rock. Một thách thức khác là các thể loại âm nhạc không ngừng phát triển. Ví dụ, thể loại nhạc điện tử đã phát triển đáng kể trong vài thập kỷ qua. Điều này gây khó khăn cho việc phát triển một mô hình học máy có thể phân loại chính xác các thể loại âm nhạc theo thời gian.

Có một số hướng phân loại thể loại âm nhạc trong tương lai. Một hướng là phát triển các mô hình học máy mạnh mẽ hơn để có thể xử lý tốt hơn những thách thức trong việc phân loại thể loại âm nhạc. Một hướng khác là phát triển các tính năng mới có thể được sử dụng để cải thiện độ chính xác của các mô hình phân loại thể loại âm nhạc.

Cuối cùng, điều quan trọng là phải xem xét ý nghĩa đạo đức của việc phân loại thể loại âm nhạc. Ví dụ: phân loại thể loại âm nhạc có thể được sử dụng để theo dõi thói quen nghe của mọi người hoặc để phân biệt đối xử với một số nhóm người nhất định. Điều quan trọng là phải nhận thức được những ý nghĩa đạo đức này và phát triển các hệ thống phân loại thể loại âm nhạc một cách có trách nhiệm.

# 2. Các nghiên cứu liên quan

## 2.1. K-nearest neighbor (KNN)

KNN là một thuật toán lazy learning, phi tham số, có thể được sử dụng cho cả nhiệm vụ phân loại và hồi quy. Nó hoạt động bằng cách tìm k trường hợp tương tự nhất cho một điểm dữ liệu mới và sau đó gán điểm dữ liệu mới cho lớp của phần lớn các lớp lân cận của nó.

Giá trị k là một siêu tham số phải được chọn bởi người dùng. Một giá trị nhỏ của k sẽ dẫn đến một mô hình phức tạp hơn, có nhiều khả năng vượt quá train data, trong khi giá trị lớn của k sẽ dẫn đến một mô hình đơn giản hơn, nhưng nhiều khả năng không phù hợp với train data.

KNN là một thuật toán đơn giản để triển khai, nhưng nó có thể tốn kém về mặt tính toán đối với các bộ dữ liệu lớn. Nó cũng nhạy cảm với quy mô của các tính năng, vì vậy thường cần phải chuẩn hóa các tính năng trước khi đào tạo mô hình.

Bất chấp những hạn chế của nó, KNN là một thuật toán mạnh mẽ có thể được sử dụng cho nhiều tác vụ khác nhau. Nó đặc biệt phù hợp với các tác vụ mà dữ liệu không được mô hình tuyến tính thể hiện tốt.

Dưới đây là một số ưu điểm của việc sử dụng KNN:

- Đơn giản để hiểu và thực hiện
- Có thể được sử dụng cho cả nhiệm vụ phân loại và hồi quy
- Có thể được sử dụng cho nhiều loại dữ liệu
- Không nhạy cảm với số lượng tính năng

Dưới đây là một số nhược điểm của việc sử dụng KNN:

- Có thể tốn kém về mặt tính toán đối với các bộ dữ liệu lớn
- Nhạy cảm với quy mô của các tính năng
- Có thể dễ bị trang bị quá mức
- Không chính xác như các thuật toán khác cho một số tác vụ

Nhìn chung, KNN là một thuật toán linh hoạt và mạnh mẽ, có thể được sử dụng cho nhiều tác vụ khác nhau. Đó là một lựa chọn tốt cho những người mới bắt đầu tìm kiếm một thuật toán đơn giản và dễ hiểu. Tuy nhiên, điều quan trọng là phải nhận thức được những hạn chế của thuật toán và thực hiện các bước để giảm thiểu chúng.

## 2.2. Mel-frequency cepstral coefficients (MFCC)

MFCC, hoặc Mel-frequency cepstral coefficients, là một tập hợp các tính năng thường được sử dụng trong nhận dạng giọng nói và các ứng dụng xử lý âm thanh khác. MFCC được trích xuất từ phổ công suất ngắn của tín hiệu âm thanh và nó đại diện cho đường bao phổ của tín hiệu. Đường bao quang phổ là thước đo sự phân bố năng lượng trong miền tần số và nó có thể được sử dụng để mô tả âm thanh của một âm vị hoặc từ cụ thể.

MFCC được trích xuất bằng quy trình bao gồm các bước sau:

1. Tín hiệu âm thanh được chia thành các khung thời gian ngắn.
2. Phổ công suất của từng khung thời gian được tính toán.
3. Phổ công suất được chuyển đổi sang thang tần số Mel.
4. Các hệ số tần số Mel được tính toán bằng biến đổi cepstral.

Thang đo tần số Mel là một thang đo phi tuyến tính được thiết kế để mô phỏng gần đúng cách mà thính giác của con người cảm nhận tần số. Biến đổi cepstral là một phép toán chuyển đổi phổ công suất thành một biểu diễn có khả năng chống nhiễu và méo tốt hơn.

MFCC có lợi thế trong việc trích xuất từ tín hiệu âm thanh. Nó tương đối mạnh đối với tiếng ồn và biến dạng, và nó có thể được sử dụng để thể hiện nhiều loại âm thanh. MFCC cũng tương đối dễ tính toán, điều này khiến nó rất phù hợp với các ứng dụng thời gian thực.

MFCC là một công cụ mạnh mẽ có thể được sử dụng để trích xuất thông tin có ý nghĩa từ tín hiệu âm thanh. Nó là tài sản quý giá cho bất kỳ ai làm việc trong lĩnh vực nhận dạng giọng nói hoặc xử lý âm thanh.

Dưới đây là một số chi tiết bổ sung về MFCC:

- Số lượng MFCC được trích xuất thường nằm trong khoảng từ 10 đến 20.
- MFCC có thể được sử dụng để đào tạo các mô hình học máy có thể được sử dụng cho các tác vụ như nhận dạng giọng nói và nhận dạng người nói.
- MFCC cũng có thể được sử dụng để trực quan hóa đường bao quang phổ của tín hiệu âm thanh.

# 3. Dữ liệu âm thanh

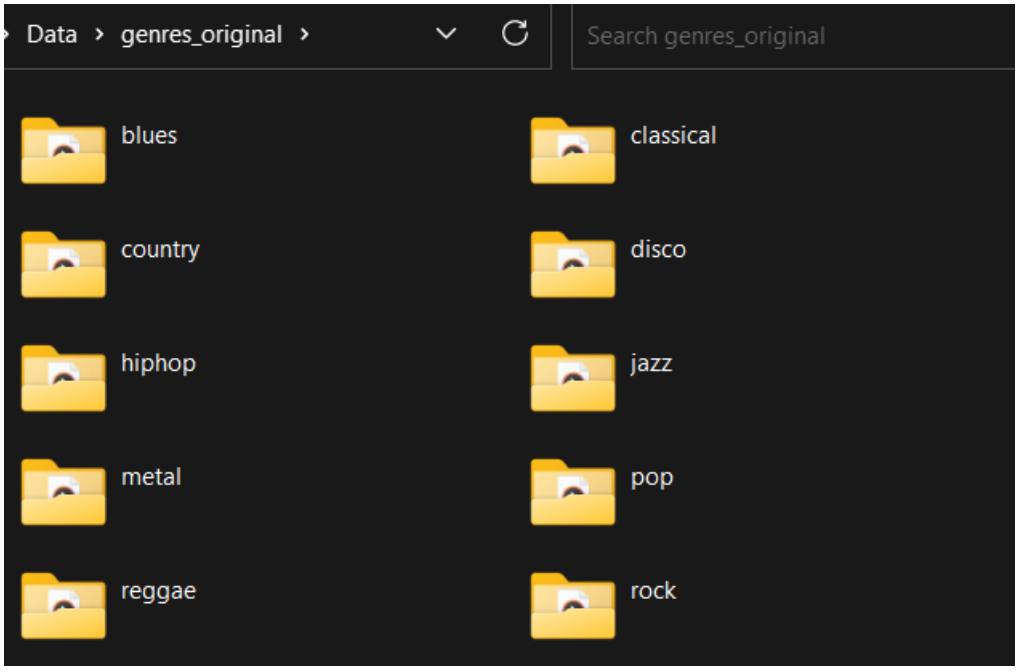
Bộ dữ liệu GTZAN là bộ dữ liệu công khai được sử dụng nhiều nhất để đánh giá trong nghiên cứu nghe máy để nhận dạng thể loại âm nhạc (MGR). Các tệp được thu thập vào năm 2000 - 2001 từ nhiều nguồn khác nhau bao gồm đĩa CD cá nhân, đài phát thanh và bản ghi micrô để thể hiện nhiều điều kiện ghi âm khác nhau. Bộ dữ liệu này được coi như là MNIST của âm thanh (MNIST là một bộ dữ liệu nổi tiếng về các chữ số viết tay thường được sử dụng để đào tạo các hệ thống xử lý ảnh cho học máy).

Tập dữ liệu được sử dụng trong dự án này chứa các tệp nhạc có độ dài 30 giây, ở định dạng .WAV và ở định dạng 16 Bit PCM WAVE trong Mã hóa 0000x01 khiến nó phù hợp với thư viện scipy.io.wave của python.

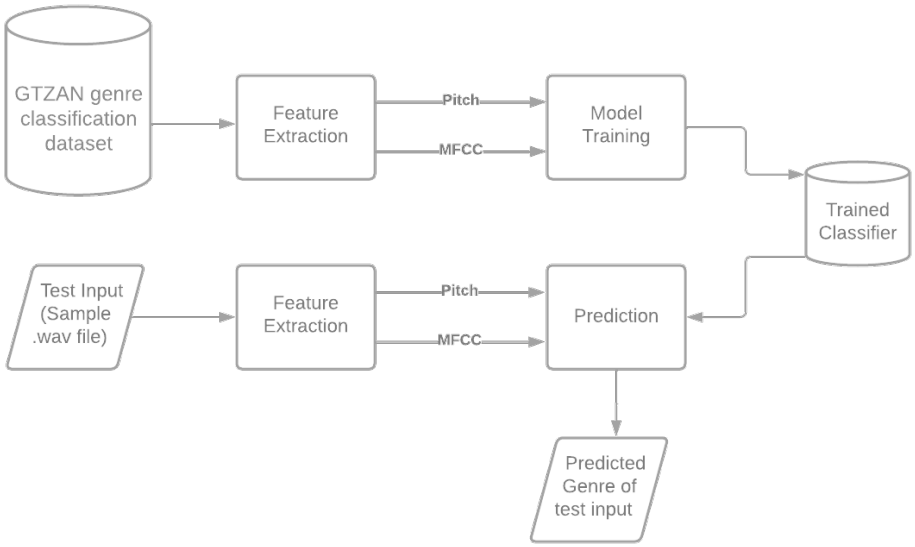
Trong bộ dữ liệu này gồm có:

- **Dòng nhạc** - Một bộ sưu tập gồm 10 dòng nhạc với 100 file âm thanh mỗi tệp, tất cả đều có độ dài 30 giây.
- **Ảnh** - Một đại diện trực quan cho mỗi file âm thanh. Một cách để phân loại là thông qua mạng thần kinh Neural Networks. Bởi vì NN thường sử dụng hình ảnh, nên các tệp âm thanh đã được chuyển đổi thành Mel Spectrogram để thực hiện điều này.
- **2 tệp CSV** - Chứa các thuộc tính của file âm thanh. Một tệp có giá trị trung bình và phương sai cho mỗi bài hát (dài 30 giây) được tính toán trên nhiều tính năng có thể được trích xuất từ file âm thanh.

Trong dự án, ta sẽ chỉ sử dụng đến các file âm thanh và sử dụng MFCC để chuyển đổi chúng thành các ma trận hình ảnh.



# 4. Thiết kế thực hiện




4.1. Trích xuất thuộc tính file âm thanh (Feature Extraction)

Bước đầu tiên cho dự án này là trích xuất các tính năng và thành phần từ tệp âm thanh. Nó bao gồm việc xác định nội dung ngôn ngữ và loại bỏ tiếng ồn.

Mel-frequency cepstral coefficients :

● Vì tín hiệu âm thanh luôn thay đổi nên trước tiên ta chia các tín hiệu này thành các khung nhỏ hơn. Mỗi khung dài khoảng 20 – 40 ms.

● Sau đó, ta xác định các tần số khác nhau có trong từng khung.

● Tiếp theo, ta tách các tần số ngôn ngữ khỏi tiếng ồn.

● Để loại bỏ nhiễu, ta thực hiện phép biến đổi cosine rời rạc (DCT) của các tần số này và chỉ giữ một chuỗi tần số cụ thể có xác suất thông tin cao

4.2. Model Training

Trong dự án này, ta sử dụng K-nearest neighbor vì nhiều nghiên cứu trước đó đã chứng minh rằng thuật toán này mang lại kết quả tốt nhất cho bài toán phân loại âm nhạc.

K-Nearest Neighbor là một trong những thuật toán Machine Learning đơn giản nhất dựa trên kỹ thuật Supervised Learning. Nó giả định sự giống nhau giữa trường hợp / dữ liệu mới và các trường hợp có sẵn và đặt trường hợp mới vào danh mục tương tự nhất với các danh mục có sẵn. Thuật toán KNN lưu trữ tất cả các dữ liệu có sẵn và phân loại một điểm dữ liệu mới dựa trên sự tương đồng. Điều này có nghĩa là khi dữ liệu mới xuất hiện thì nó có thể dễ dàng được phân loại thành một danh mục tốt bằng cách sử dụng thuật toán KNN.

4.3. Phân loại file âm thanh

Bước này liên quan đến việc sử dụng dữ liệu thử nghiệm để thử và dự đoán thể loại của tệp nhạc mẫu so với mô hình được đào tạo và đạt được các tham số chính xác.

# 5. Thực nghiệm

## 5.1. Thuật toán K-nearest neighbor (KNN)

### 5.1.1. Hàm tính khoảng cách giữa các vectơ thuộc tính

```
5  def distance_cal(instance1, instance2, k):
6      mm1 = instance1[0]
7      cm1 = instance1[1]
8      mm2 = instance2[0]
9      cm2 = instance2[1]
10     distance = np.trace(np.dot(np.linalg.inv(cm2), cm1))
11     distance += (np.dot(np.dot((mm2 - mm1).transpose(), np.linalg.inv(cm2)), mm2 - mm1))
12     distance += np.log(np.linalg.det(cm2)) - np.log(np.linalg.det(cm1))
13     distance -= k
14     return distance
```

Hàm ***distance_cal()*** tính toán khoảng cách giữa hai ma trận đầu vào trong mô hình hỗn hợp Gaussian. Hàm tính ma trận trung bình (mean matrix, hay mm) và ma trận hiệp phương sai (covariance matrix, hay cm) giữa từng vectơ thuộc tính của dataset (instance1) và vectơ của testset (instance2), với test set là ma trận gồm các vectơ thuộc tính khi train, hoặc là vectơ thuộc tính của bài hát trong thực nghiệm.

Những hàm numpy được sử dụng:

- np.linalg.inv: tính ma trận nghịch đảo
- np.dot: tính tích vô hướng giữa 2 ma trận
- np.trace: tính tổng giá trị trên đường chéo của ma trận
- np.log: tính logarit tự nhiên (log(exp(x) hay ln(x))

Hàm tính khoảng cách giữa hai ma trận như sau:

***distance = np.trace(np.dot(np.linalg.inv(cm2), cm1))***
***+ (np.dot(np.dot((mm2 - mm1).transpose(), np.linalg.inv(cm2)), mm2 - mm1))***
***+ np.log(np.linalg.det(cm2)) - np.log(np.linalg.det(cm1)) - k***

- ***np.trace(np.dot(np.linalg.inv(cm2), cm1)):*** tính khoảng cách Mahalanobis giữa hai vectơ. Khoảng cách Mahalanobis là khoảng cách từ một vectơ y đến một phân bố với giá trị trung bình μ và hiệp phương sai Σ, được tính theo công thức:

$$d = \sqrt{(y - \mu)\Sigma^{-1}(y - \mu)'}$$

- ***(np.dot(np.dot((mm2 - mm1).transpose(), np.linalg.inv(cm2)), mm2 - mm1)):*** tính khoảng cách Euclide bình phương giữa hai vectơ.
- ***np.log(np.linalg.det(cm2)) - np.log(np.linalg.det(cm1)):*** tính sự khác biệt giữa định thức của ma trận hiệp phương sai nghịch đảo của hai ma trận đầu vào.
- ***k:*** là một hằng số kiểm soát khoảng cách

### 5.1.2. Hàm xác định các hàng xóm (neighbor)

```
def get_neighbors(trainingSet, instance, k):
    distances = []
    for x in range(len(trainingSet)):
        dist = distance_cal(trainingSet[x], instance, k) + distance_cal(instance, trainingSet[x], k)
        distances.append((trainingSet[x][2], dist))
    distances.sort(key=operator.itemgetter(1))
    neighbors = []
    for x in range(k):
        neighbors.append(distances[x][0])
    return neighbors
```

Hàm ***get_neighbors()*** tìm k hàng xóm gần nhất của vectơ thuộc tính trong train dataset. Đầu tiên, ta khởi tạo ma trận chứa khoảng cách đo được tới các thể loại nhạc có trong train dataset. Khoảng cách được tính như trên vì không phải lúc nào khoảng cách giữa vectơ thuộc tính và vectơ trong train dataset cũng đối xứng, nên việc cộng khoảng cách theo cả hai chiều sẽ đảm bảo KNN sẽ không thiên về bất kì vectơ thuộc tính nào trong train dataset.

Sau đó, ta sắp xếp ma trận tăng dần theo khoảng cách đo được và lấy k phần tử đầu.

### 5.1.3. Hàm xác định các hàng xóm gần nhất (nearest neighbor)

```
def nearest_class(neighbors):
    class_vote = {}
    for x in range(len(neighbors)):
        response = neighbors[x]
        if response in class_vote:
            class_vote[response] += 1
        else:
            class_vote[response] = 1
    sorter = sorted(class_vote.items(), key=operator.itemgetter(1), reverse=True)
    genres = []
    votes = []
    for item in sorter:
        genres.append(item[0])
        votes.append(item[1])
    return genres, votes
```

Hàm ***nearest_class()*** xác định những hàng xóm gần nhất của vectơ thuộc tính từ những hàng xóm tìm được ở trên. Với mỗi hàng xóm tìm được, ta thêm một phiếu vote cho hàng xóm đó. Sau khi duyệt xong tất cả hàng xóm, ta sắp xếp giảm dần theo số vote và trả về thể loại nhạc mà âm thanh đầu vào có khả năng thuộc về cao nhất.

### 5.1.4. Hàm tính độ chính xác của KNN

```
def get_accuracy(testSet, predictions):
    correct = 0
    for x in range(len(testSet)):
        if testSet[x][-1] == predictions[x]:
            correct += 1
    return 1.0*correct/len(testSet)
```

Hàm ***get_accuracy()*** được sử dụng để tính độ chuẩn xác của KNN trên test dataset. Ta tỉnh tổng số dự đoán giống với nhãn của test dataset, và lấy tổng đó chia cho số bài hát trong dataset để tính phần trăm dự đoán đúng.

## 5.2. Thuật toán thu thập dữ liệu và train dataset

### 5.2.1. Hàm trích xuất thuộc tính của bài hát

```
directory = "./Data/genres_original"
f = open("my.dat", 'wb')
i = 0

for folder in os.listdir(directory):
    i += 1
    if i == 11:
        break
    for file in os.listdir(directory+"/"+folder):
        # Get Sample rate and data of audio
        (rate, sig) = wav.read(directory+"/"+folder+"/"+file)
        # Extract MFCC from audio data
        mfcc_feat = mfcc(sig, rate, winlen=0.020, appendEnergy=False)
        # Calculate the covariance (how related are the two mfcc)
        covariance = np.cov(np.matrix.transpose(mfcc_feat))
        # Calculate the mean of MFCC (the central tendency of the MFCC)
        mean_matrix = mfcc_feat.mean(0)
        # Create a tuple containing the mean matrix, covariance and a class label
        feature = (mean_matrix, covariance, i)
        # Record the tuple in file
        pickle.dump(feature, f)

f.close()
```

Ta xác định vị trí của file data trong hệ thống và truy cập vào nó. Với mỗi file trong hệ thống, ta trích xuất tần số lấy mẫu và tín hiệu của file âm thanh đang được xử lý. Tín hiệu được trích xuất chính là dữ liệu âm thanh, được lưu dưới dạng ma trận Numpy.

Tiếp theo, ta trích xuất Mel-Frequency Cepstral Coefficients (MFCC) từ dữ liệu âm thanh thu được ở trên. MFCC được ta sử dụng để mô tả âm thanh của file, và có thể được biểu diễn bằng hình ảnh.

Sau đó, ta tính hiệp phương sai của các thuộc tính MFCC trên để đo mức độ liên quan giữa hai thuộc tính MFCC liên tiếp. Cùng lúc đó, ta tính giá trị trung bình của các thuộc tính để đo mật độ phân bố của các thuộc tính MFCC.

Cuối cùng, ta tạo một tuple các thuộc tính để chứa ma trận giá trị trung bình, ma trận hiệp phương sai và nhãn i (i được đánh số từ 1 đến 10, tương ứng với 10 thể loại nhạc mà ta sử dụng). Các tuple này được lưu lại vào file data my.dat.

Sau khi trích xuất những thuộc tính cần thiết của toàn bộ file âm thanh ta có, ta đóng file và chuyển sang bước xử lý tiếp theo.

### 5.2.2. Hàm khởi tạo train data và test data

Ta khởi tạo ma trận dataset để chứa toàn bộ thuộc tính mà ta đã trích xuất từ trên. Trong quá trình đó, ta chia ngẫu nhiên từng file và thuộc tính của chúng thành train và test dataset theo tỉ lệ 2:1.

```
dataset = []


1 usage
def load_dataset(split, trSet, teSet):
    with open("my.dat", 'rb') as f1:
        while True:
            try:
                dataset.append(pickle.load(f1))
            except EOFError:
                f1.close()
                break

    for x in range(len(dataset)):
        if random.random() < split:
            trSet.append(dataset[x])
        else:
            teSet.append(dataset[x])


training_set = []
test_set = []
load_dataset(0.66, training_set, test_set)
```

### 5.2.3. Hàm đánh giá độ chính xác

```
predictions = []
for y in range(len(test_set)):
    predicts, votes = nearest_class(get_neighbors(dataset, test_set[y], 7))
    predictions.append(predicts[0])

accuracy = get_accuracy(test_set, predictions) * 100
print(f"The accuracy is about{accuracy: .2f}%")
```

Với mỗi vectơ thuộc tính trong test dataset, ta xác định những hàng xóm gần nhất của nó và đưa ra dự đoán. Ở đây ta chọn k = 7 là tối ưu nhất vì với k bé hơn, dự đoán sẽ chuẩn hơn nhưng độ chính xác không đáng tin cậy; còn với k lớn hơn thì kết quả sẽ bị nhiễu, làm giảm độ chính xác. Với những dự đoán được đưa ra, ta tính độ chính xác dựa trên nhãn có sẵn của test data set và đưa ra kết quả. Kết quả của quá trình này như sau:

```
C:\Users\Pengu\AppData\Local\Programs\Python\Python310\python.exe "D:\Project 2\test 2\music-genre-classification.py"
WARNING:root:frame length (882) is greater than FFT size (512), frame will be truncated. Increase NFFT to avoid.
The accuracy is about 84.24%

Process finished with exit code 0
```

Độ chính xác rơi vào khoảng 84,24%, tương đối tốt so với thuật toán KNN. Dòng lỗi đỏ xuất hiện do có một số file âm thanh có thời lượng dài hơn khung cho phép của FFT và được tự động cắt đi, không ảnh hưởng đến quá trình train và test. Ở phần sau ta sẽ thử nghiệm với file âm thanh ngẫu nhiên từ data đã train và file âm thanh từ thư viện nhạc của máy.

## 5.3. Thử nghiệm thực tế

### 5.3.1. Hàm phân loại dựa trên data đã train

Tương tự như phần 5.2.3, ta load dataset để sử dụng trong quá trình phân loại.

```
# Create Result matrix
directory = "./Data/genres_original"
results = dict()
i = 1
for folder in os.listdir(directory):
    results[i] = folder
    i += 1
```

Ta khởi tạo từ điển results để lưu kết quả dự đoán dưới dạng chữ thay vì từ 1 đến 10.

```
def classifier(path, k):
    (rate, sig) = wav.read(path)
    mfcc_feat = mfcc(sig, rate, winlen=0.020, appendEnergy=False)
    covariance = np.cov(np.matrix.transpose(mfcc_feat))
    mean_matrix = mfcc_feat.mean(0)
    feature = (mean_matrix, covariance, 0)
    predicts, votes = nearest_class(get_neighbors(dataset, feature, k))

    plt.subplot(1, 2, 1)
    plt.plot(mfcc_feat)
    plt.title("MFCC Feat")

    plt.subplot(1, 2, 2)
    plt.plot(mean_matrix)
    plt.title("Mean Matrix")

    plt.show()
    return [results[predict] for predict in predicts], votes

```

Ta trích xuất thuộc tính của file âm thanh được trỏ đến và đưa ra dự đoán, cũng như thể hiện thuộc tính được trích xuất và giá trị trung bình của chúng bằng hình ảnh. Ở đây, thay vì chỉ đưa ra dự đoán có số vote cao nhất như trong quá trình train và test, ta lưu lại toàn bộ dự đoán và phần trăm vote của từng dự đoán.

### 5.3.2. Đưa ra kết quả dự đoán

```
from classifier import classifier

path = "Test file/pop.00018.wav"
k = 7
predictions, votes = classifier(path, k)
print("According to my predictions: ")
for i in range(len(predictions)):
    possibility = votes[i] / k * 100
    print(f"There's{possibility: .2f}% chance that it's " + predictions[i].upper())

```

Ta xác định vị trí của file âm thanh và phân loại nó. Kết quả lần lượt như sau:

Với file ***pop.00018.wav*** từ dataset, máy rất tự tin với dự đoán POP và nó hoàn toàn chính xác

```
C:\Users\Pengu\AppData\Local\Programs\Py
According to my predictions:
There's 100.00% chance that it's POP
```

Với file ***rock.00010.wav*** từ dataset, máy chỉ dám chắc 57,14% đây là dòng nhạc ROCK, với khả năng còn lại chia đều cho HIPHOP, DISCO và COUNTRY.

```
C:\Users\Pengu\AppData\Local\Programs\Pyt
According to my predictions:
There's 57.14% chance that it's ROCK
There's 14.29% chance that it's HIPHOP
There's 14.29% chance that it's DISCO
There's 14.29% chance that it's COUNTRY
```

Với file ***Someone In The Crowd.wav*** từ hệ thống, dự đoán khá chính xác khi đây là một bài hát được kết hợp từ POP và JAZZ. (Đây là một trong những bài hát chủ đạo của bộ phim La La Land – một bộ phim về âm nhạc JAZZ)

```
C:\Users\Pengu\AppData\Local\Programs\Py
WARNING:root:frame length (960) is great
According to my predictions:
There's 57.14% chance that it's POP
There's 14.29% chance that it's JAZZ
There's 14.29% chance that it's HIPHOP
There's 14.29% chance that it's REGGAE
```

Với file ***Blake Shelton – Drink On It.wav*** từ hệ thống, dự đoán đã không chính xác vì đây là bài hát thuộc thể loại COUNTRY, được trình bày bởi Blake Shelton – một nhạc sĩ đồng quê nổi tiếng của Mỹ.

```
C:\Users\Pengu\AppData\Local\Programs\Pyt
WARNING:root:frame length (960) is greate
According to my predictions:
There's 71.43% chance that it's POP
There's 14.29% chance that it's REGGAE
There's 14.29% chance that it's HIPHOP
```

# 6. Kết luận

Trong dự án này, ta đã sử dụng thuật toán K-nearest neighbor để phân loại dòng nhạc của bài hát. Ta đã dùng dataset với 1000 bài hát, mỗi file đều được gán nhãn với thể loại của nó. Ta đã đào tạo mô hình KNN, thử nghiệm nó và đạt được độ chính xác lên đến 84%, là một kết quả tương đối tốt đối với mô hình này.

Tuy vậy, do dataset còn rất nhỏ so với thư viện âm nhạc của thế giới ở thời điểm hiện tại và sự phát triển của ngành âm nhạc, độ chính xác này hoàn toàn có thể cải thiện được. Ta có thể sử dụng dataset lớn hơn, sử dụng phương pháp trích xuất thuộc tính khác, hoặc sử dụng thuật toán học máy phức tạp hơn, tiên tiến hơn.

Trong tương lai, ta có thể phát triển thuật toán để tạo một playlist ngẫu nhiên gồm những bài hát thuộc thể loại mà người dùng hay nghe trên các trang nghe nhạc trực tuyến, hay phát hiện vi phạm bản quyền trong quá trình sản xuất và phát hành nhạc.

Nhìn chung, dự án này đã đạt mục tiêu được đề ra. Với bài toán phân loại dòng nhạc của bài hát, KNN thực sự là một thuật toán nhiều triển vọng và có thể được phát triển xa hơn nữa trong tương lai.