Audio signal analysis for Songs' Mood Recognition by Clustering and Neural Network

By
My H Pham

Presented in Partial Fulfillment of the
Requirements of Senior Independent Study

Advised by
Dr. Colby Long
Statistical and Data Science

Spring 2025

Abstract

In the evolving landscape of digital music consumption, accurate mood classification plays a crucial role in enhancing user experience on streaming platforms. While services like Spotify and SoundCloud have introduced mood-based features, inconsistencies in labeling often result in disjointed listening experiences. This project proposes a music mood recognition algorithm based on audio processing analysis and machine learning, particularly focusing on clustering techniques and neural networks.

This study utilizes a dataset of 389 songs across eight genres from the Free Music Archive (FMA), with music features such as tempo, key, scale (surface-level), and audio deep learning features such as spectral centroid and Mel-spectrograms, which are extracted through extensive preprocessing, including denoising, trimming, normalization, and Short-Time Fourier Transform (STFT). Both audio features and deep spectral representations are analyzed together and individually to find the best models.

Two machine learning approaches are employed: unsupervised clustering (K-Means and Fuzzy C-Means) to group songs into mood categories without labels, and a supervised neural network trained with sparse categorical cross-entropy and optimized via Stochastic Gradient Descent. Principal Component Analysis (PCA) is used to reduce feature dimensionality. Mood categories are defined by energy and emotion levels (Energetic/Relaxing × Happy/Sad), forming four mood groups.

The results show K-Means provides clearer, interpretable clusters, while the neural network—though flexible—exhibits overfitting, with a test accuracy of 43.59%. Fuzzy clustering reveals the overlapping emotional content in music more effectively than hard clustering.

Limitations include dataset size, subjectivity in mood labeling, and lack of lyrical or contextual data.

This project contributes a potential framework for mood-based recommendation, offering feasible applications in playlist generation and music discovery for users and artists alike. Future work may explore deep learning architectures, context-aware personalization, and integration of lyrics for a more comprehensive emotional modeling of music.

Acknowledgements

This work is dedicated to my parents, Hai Pham and Anh Nguyen. Thank you for your trust, investment, and support for me throughout my college. Thank to my older sister Elaine Pham for going through the hard time with me. Thank Dr. Long for your guidance in this project, in my study and career. Thank you for your patience with me, I would not be able to get far without your support. I want to thank Hiep Nguyen to assure me with my dedication in this career path. Thank nurse Lori Hartzler for consulting me and helping me through my hardest time. Lastly, thank to my Bangladeshi and Nepali friends for making Wooster home.

Table of Contents

List of Figures

List of tables

I. Introduction

Music has been an integral part of human life for over 40,000BP, evolving alongside advancements in technology and accessibility [34]. The growth of music production technology in the late 19th century revolutionized how people engage with music, leading to the emergence of various listening platforms [33]. Today, digital streaming services such as Spotify and SoundCloud leverage algorithms to enhance user experiences by offering personalized recommendations and categorized content.

Spotify is widely recognized for its sophisticated recommendation system, which analyzes user preferences to suggest songs with similar characteristics, enabling listeners to discover new music aligned with their tastes. Spotify combines both user-based and item-based collaborative filtering methods for their recommendation system [35]. User-based method analyzes other users' searching history and playlists to find users with similar interest and recommend their songs to others, while item-based method analyzes songs to find similarities. Spotify's item-based method analyzes individual's listening and search history, and playlists to find similar songs. SoundCloud recently introduced a feature that categorizes tracks based on mood and genre in October 2024, with 64 different categories [31]. Prior to this update, SoundCloud's shuffle function would randomly play any song within a playlist, often resulting in abrupt shifts in mood or energy levels, which is the inspiration for this project to develop a songs mood classification algorithm. The lack of consistency in mood flow could disrupt the listening experience, particularly for users who rely on music to maintain a specific emotional state—whether for relaxation or energetic. While the introduction of mood-based categorization is a step forward, the accuracy of these labels remains inconsistent. Some tracks are incorrectly classified, leading to mismatches between the intended mood and the actual audio characteristics.

To address this issue, I aim to develop an item-based music mood classification algorithm that more precisely categorizes songs based on their audio features without an access to users' database. Both supervised and unsupervised methods are used in this project. Audio and spectral features are selected and will be used individually and combined to find the best models. Previous study proved that combining spectral and audio features outperforms any individual features [30].

Given the limitations in accessing user preference data and proprietary platform information, my approach will focus on audio analysis rather than behavioral tracking. The proposed model will utilize a combination of deep and surface-level features to classify songs. Deep features, such as Mel-spectrograms and spectral centroids, will capture the underlying tonal and spectral properties of audio, while surface features, including tempo, key, scale, and dynamic strength, will provide additional context for classification. The project will integrate both supervised and unsupervised learning techniques, evaluating models with and without the inclusion of Mel-spectrograms to determine the most effective classification approach.

Existing music recommendation systems primarily emphasize genre classification or popularity-based suggestions, often neglecting the emotional resonance and continuity of a listener's experience. This project aims to fill this gap by prioritizing mood-based classification to create a more cohesive and immersive listening experience. By enabling raw audio files as input and incorporating extensive preprocessing techniques with item-based approach using audio and spectral features, the existing system will be accessible to a wide range of users, including both listeners and music creators.

Inspired by SoundCloud’s open platform, which allows independent artists to share their work without traditional industry barriers, this project could serve as a valuable tool for amateur musicians and indie music enthusiasts. If deployed as an application or integrated as an extension feature, the proposed algorithm could automatically tag music with accurate mood labels, improving discoverability for emerging artists while enhancing the listening experience for audiences. Ultimately, this initiative seeks to refine how music is categorized, ensuring that listeners can enjoy seamless transitions between tracks that align with their desired emotional ambiance.

II. Literature Review

The paper "Detecting Music Genre Using Extreme Gradient Boosting" presents a methodology for classifying music genres in the CrowdAI music genre challenge [19]. Researchers Benjamin Murauer and Günther Specht used various models, including convolutional neural networks (CNNs) for spectrogram classification, deep neural networks (DNNs), and ensemble methods with numerical audio features. Additionally, the variables mentioned are Numerical Audio Features, such as Average loudness, Mel bands skewness, mean Spectral flux, median Rhythm beats per minute (bpm), Danceability, Tonal key (categorical, one-hot encoded), and Tonal chord (categorical, one-hot encoded). They also use Spectrograms, which different music genres feature different patterns in the distribution and occurrences of specific frequency ranges, which are displayed in the image. I will also apply these features in my model to find patterns of spectrograms for mood. Their best-performing model was an extreme gradient boosting (XGBoost) classifier – numerical features, which outperformed other approaches with a mean log loss score of 0.82. Despite experimenting with DNN and CNN models, their effectiveness was limited by GPU memory, and the ensemble methods ultimately provided the highest accuracy in predicting genres based on extracted audio features.

The paper “Unsupervised Approach to Hindi Music Mood Classification” presents an unsupervised approach to Hindi music mood classification using audio features. Key features selected for classification include intensity, timbre (such as MFCCs, spectral shape, and spectral contrast), and rhythm (rhythm strength, regularity, and tempo) [20]. These features were extracted using the jAudio toolkit, and different patterns were associated with various emotional expressions in music. The classification was conducted using the fuzzy c-means clustering algorithm, where songs were grouped into five mood clusters based on a custom mood taxonomy inspired by MIREX and Russell’s Circumplex model. The approach achieved an accuracy of 48%, highlighting potential improvements with additional features and enhanced clustering techniques.

The paper "Music Emotion Recognition by Using Chroma Spectrogram and Deep Visual Features" explores a deep learning approach for recognizing emotions in music. Authors Mehmet Bilal Er and Ibrahim Berkan Aydilek propose a method that leverages pre-trained deep learning models—AlexNet and VGG-16—on chroma spectrograms to classify emotions in music [21]. Chroma spectrograms, which depict the energy distribution around musical notes, serve as input features. Deep visual features are extracted from specific layers in these networks (such as Fc6 and Fc7), which are then used to train Support Vector Machines (SVM) and Softmax classifiers.

The study also includes data augmentation through time-stretching and shifting, which improved classification accuracy. Testing on two datasets, the best result was achieved with the VGG-16 model’s Fc7 layer, reaching 89.2% accuracy. This research demonstrates that pre-trained deep networks can effectively recognize music emotions, particularly when using augmented data.

The study "When Lyrics Outperform Audio for Music Mood Classification: A Feature Analysis" by Xiao Hu and J. Stephen Downie investigates the comparative effectiveness of audio and lyric features in classifying music moods [6]. Using a dataset of 5,296 songs with audio and lyrics, divided into 18 mood categories based on user-generated tags, the research reveals that certain moods are better classified by lyrics, while others by audio. Notably, lyric features outperformed audio in seven mood categories (e.g., "romantic," "angry," "cheerful"), showing strong semantic connections between lyrics and mood. Only one category, "calm," was better classified by audio.

The study applies multiple feature types from both lyrics (including content words, psycholinguistic lexicons, and stylistic features) and audio (spectral features). Findings suggest that content words and psycholinguistic features, such as General Inquirer and ANEW (Affective Norms for English Words), often align well with mood categories. These insights highlight the significance of semantic elements in lyrics for mood classification, especially in categories that rely on emotional and psychological word associations. The study concludes that combining lyric and audio features can enhance mood classification and suggests further exploration of non-spectral audio features, like rhythmic and harmonic components.

The paper "An Artificial Intelligence-based Classifier for Musical Emotion Expression in Media Education" by Jue Lian explores the development of a music emotion classifier using artificial intelligence, specifically designed for online music education environments [23]. The study addresses the limitations of existing music emotion analysis, which often rely on pre-labeled content like lyrics and fail to capture real-time audio features effectively. To overcome this, Lian integrates the Internet of Things (IoT) technology with an AI-based classifier, employing a Radial Basis Function (RBF) neural network to classify emotions in music.

The system uses sound-level segmentation, feature extraction, and intelligent note segmentation to identify and classify audio features. It then categorizes emotions into four categories (Quiet, Happy, Sad, and Excited) aligned with the Hevner emotion model. This classifier achieved a high accuracy rate of over 95%, with a fast audio recognition time of just 0.004 minutes. By leveraging IoT, the model also enables real-time feedback and interaction between teachers and students, potentially enhancing the teaching loop in online music education by aligning emotional recognition with human cognition.

In previous research, two common techniques used for music mood classification are clustering and neural network, which I used for this project. With neural network, the model can analyze spectrogram images together with important music features. In this project, I will use Mel-spectrogram, whose scale is more suitable for human hearings and less technical than spectrograms. I will also compare clustering models using music features, with and without Mel-spectrogram analysis; and a neural network model combined all variables. From the work of Jue Lian, I decide not to use lyrics as pre-labeled content was not effective [23]. Previous study has labeled mood by Hevner emotion model (Quiet, Happy, Sad, and Excited) [30]. We also have 4 groups; however, the mood is classified objectively combining energy (Energetic-E/Relaxing-R) and emotion (Happy-H/Sad-S) into EH, ES, RH, or RS. Furthermore, in this project I will use K-means clustering on only surface-level music features, which has been previously used by researchers for music semantic classification [30].

III. Music Background

i. Spectrogram and Mel-spectrogram

A spectrogram is a visual representation of the spectrum of frequencies in a sound over time. Spectrum in audio is the frequency range that is interpretable for human hearing. The span is often in between 20Hz to 20000Hz. A frequency is the number of times a sound wave repeats per second, which is the main determinator for pitch (high or low notes). Low frequency ranges from 5hz to 70hz; mid frequency is around 500hz; high frequency is above 2000hz [1]. Spectrogram shows which frequencies are present and their intensities, helping to analyze the texture and timbre (brightness, depth, richness of sounds) of a song.

The x-axis represents time, while the y-axis shows frequency, measured in Hz. The third dimension shows the amplitude-what we perceived as volume or loudness, of a frequency at a time. The color of the spectrogram specifies the intensity of amplitude. Higher amplitudes are shown by darker color – purple in the spectrogram.

Spectrograms are not only useful for identifying frequency ranges but also for visualizing detailed speech or sound structures. When analyzing speech signals, we can observe two key features: formants and harmonics. Formants appear as broad, smooth bands and help define vowel sounds, while harmonics appear as regularly spaced horizontal lines indicating pitch and voice quality [50].

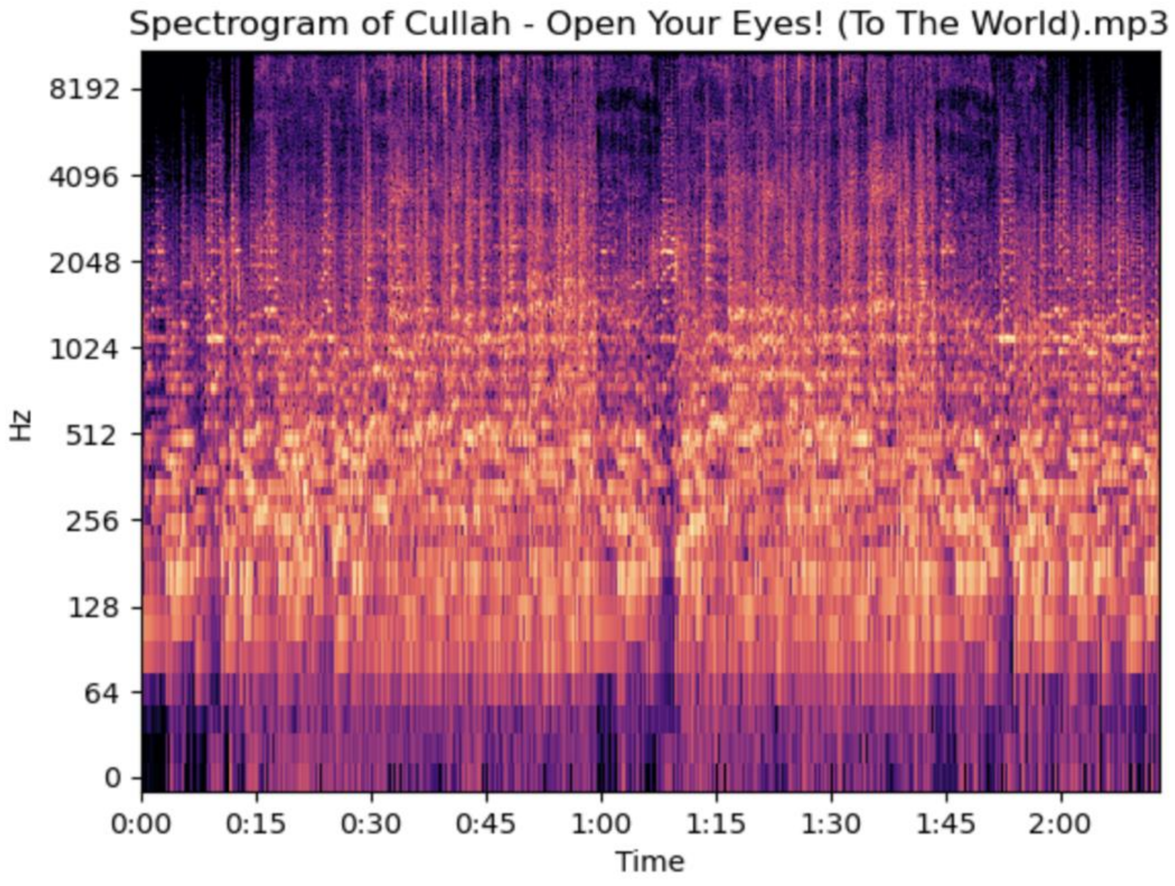


Figure 3.1.1: Spectrogram of a song in dataset

Low Frequencies (bass and drums) typically create a sense of depth and power, often evoking darker, more grounded emotions. Emphasizing lower frequencies can give a song a heavier, more intense feel [37].

High Frequencies (vocals, cymbals) often associated with lightness, clarity, and brightness, influencing a song’s perceived brightness. High frequencies can add a sense of airiness or excitement [37].

Regarding frequency dynamics, changes in intensity across frequencies, for instance, when a song builds up in energy or suddenly quiets down, affect emotional tension [36]. A song that gradually increases in high frequencies can create excitement, while a sudden drop might introduce a feeling of suspense or release.

Spectrograms also show harmonic content, which contributes to the song's "color." Rich harmonics can make a song sound fuller and warmer, while fewer harmonics might make it sound stark or cold. In the spectrogram above, lower frequencies are densely packed, indicating strong bass components like kick drums and basslines. There are multiple horizontal bands at regular intervals in the mid frequencies, suggesting harmonic overtones which can come from melodic instruments or vocals. We can see clear repetitive structures at every 1:00 minute, which may indicate chorus or verse repetition in the song.

In music, spectrograms provide insight into the distribution of frequencies within a song, showing how different pitches and overtones are present at any given moment. The use of spectrograms allows for detailed analysis of a song’s timbre, texture, and dynamics, which can impact the listener's emotional response. For instance, a dense, complex spectrogram with overlapping frequencies might convey intensity or tension, while a sparse spectrogram could suggest calmness or simplicity. In this project, analyzing spectrograms helps to identify distinct characteristics within songs that correlate with specific moods, enabling more accurate mood classification.

A Mel-spectrogram (Mel Spec) is a time-frequency representation of an audio signal that maps frequencies to the Mel scale, which is designed to match human auditory perception. This transformation makes it more effective for analyzing sounds as humans hear them, rather than how they are physically produced. Unlike a regular spectrogram, a Mel-spectrogram makes two key modifications. First, it replaces the linear frequency scale with the Mel scale on the y-axis. This adjustment reflects the way humans perceive pitch, where lower frequencies are perceived with finer resolution than higher frequencies. Second, it uses the decibel (dB) scale instead of raw amplitude to represent intensity, with color variations indicating different levels of loudness.

In a Mel-spectrogram, Hertz (Hz) is used to measure frequency, while decibels (dB) indicate the intensity of sound. This format is widely used in music analysis, speech recognition, and machine learning applications for audio processing, making it an essential tool in modern sound analysis [5].

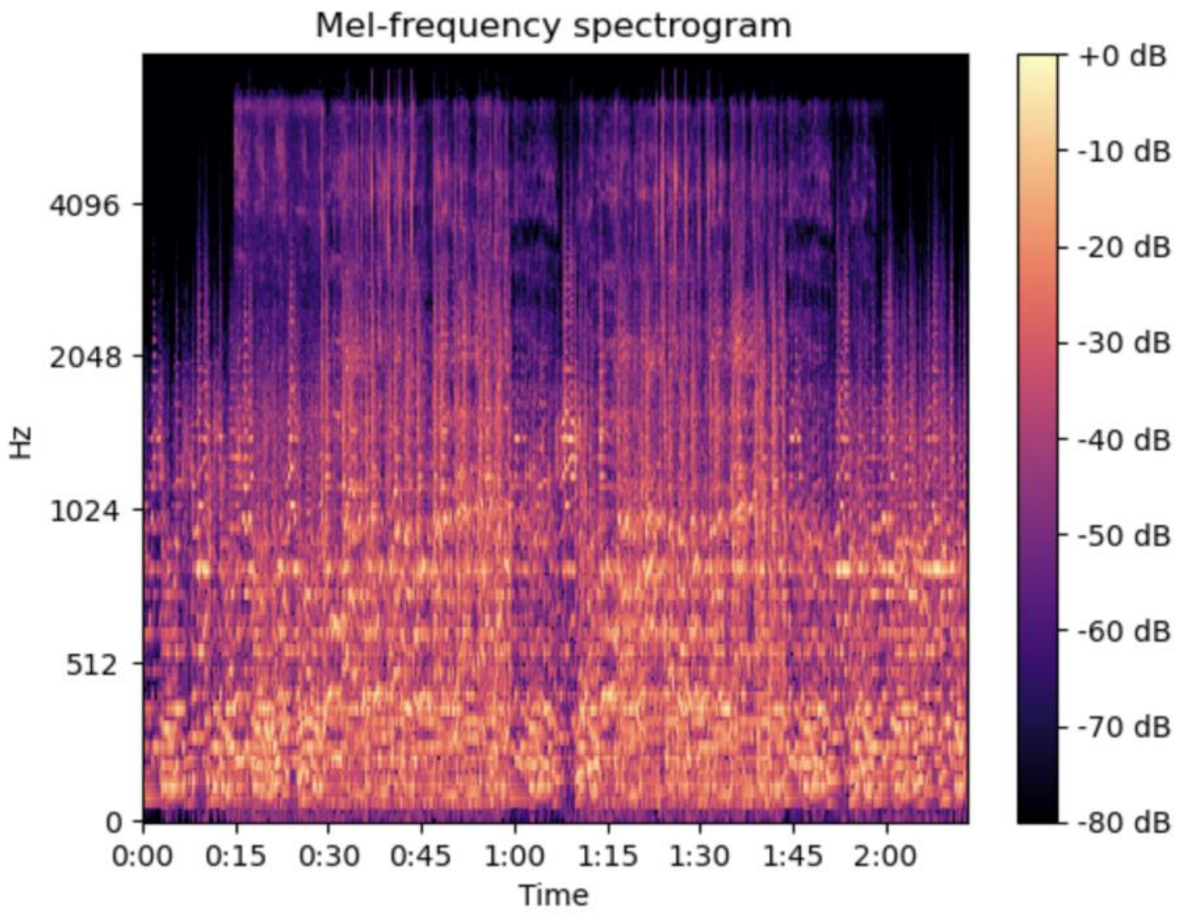


Figure 3.1.2: Mel-Spectrogram of a song in the dataset

ii. Spectral Centroid:

Spectral centroid shows where the central of spectrum is located. Spectral centroid is associated with the brightness of sound, which can potentially correlate with music mood classification [2].

To calculate spectral centroid, peaks in the frequency spectrum are detected. The vertical lines line up with the peaks of the spectrum, some aren't included as below threshold point. This is peak detection threshold method. More details on the calculation method can be found at Spectral Centroid [40].

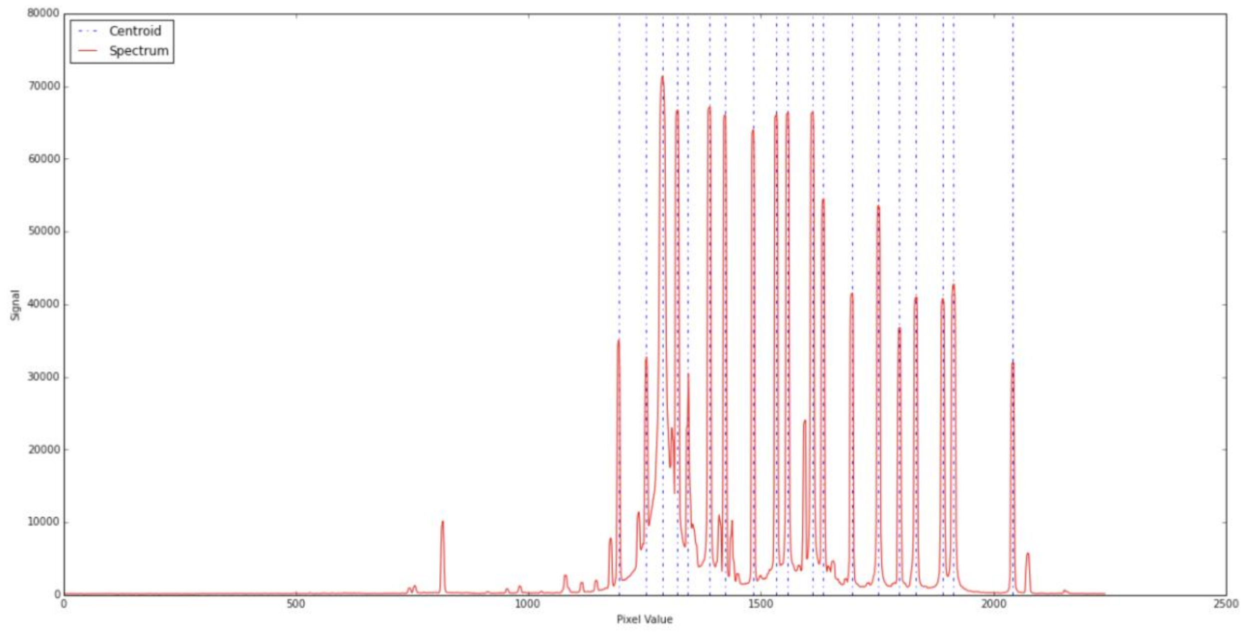


Figure 3.2.1: Centroid threshold peak detection [4]

Spectral centroid is a good predictor of the "brightness" of a sound, it is widely used in digital audio and music processing as an automatic measure of musical timbre. Previous work has shown that higher values correspond to brighter sounds. In the study, others 1 has more background noise and silence, and others 2 and 3 have more abrupt sounds. We can see spectral centroids are higher for others 2 and 3, which can imply higher spectral centroid tends to present more abrupt sounds, while lower centroid shows mainly noise or silence [3].

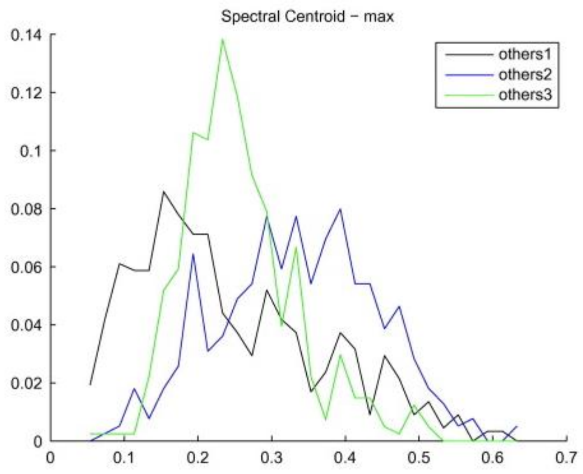


Figure 3.2.2: Special Centroid visualization of 3 different noises [22]

iii. Scale

A scale is a sequence of notes arranged in ascending or descending order of pitch. Scales serve as the foundation for melodies, harmonies, and chord progressions in music. They can vary in the number of notes and the intervals between them, leading to distinct sounds and moods.

Generally, with a minor scale, the song will be more likely to lean on the sad/ less energetic side; while with a major scale, the song would sound brighter, thus fall to the side of happy or positive [8].

iv. Key

The key to a song refers to the scale or group of pitches that form the foundation of the piece. Keys range from A to G, with sharp and flat.

The key of a piece refers to the tonal center around which its pitches are organized, often established by a specific scale. A song's key can heavily influence its emotional effect; for example, major keys are typically associated with happy or uplifting moods, while minor keys often convey sadness, introspection, or intensity [9]. By identifying the key of a song, this project can better categorize songs into mood-based playlists. The key provides context to the pitch structure and establishes harmonic expectations, which contribute to the overall mood of the music. This feature is crucial for the recommendation system as it enables the creation of cohesive playlists that maintain a consistent emotional tone.

In Western classical music tradition, each key has historically been associated with distinct emotional qualities. Major keys are often linked to positive and uplifting emotions—C Major represents innocence and simplicity, D Major evokes triumph and celebration, E Major conveys joy, and A Major suggests youthful love and trust. In contrast, minor keys tend to express more introspective or sorrowful emotions—C Minor reflects unfulfilled love, D Minor conveys melancholy, E Minor captures gentle lament, and F Minor suggests deep despair. Some keys carry more complex or intense meanings: D♭ Major and A♭ Minor reflect internal struggle or emotional ambiguity, while C# Minor and D# Minor evoke spiritual lament and dark psychological distress. B♭ Minor and B Major are tied to extremes, ranging from bitterness and suicidal despair to wild emotional passion and fury. Though these associations are culturally and historically shaped rather than universally fixed, they illustrate how composers and listeners have long attributed symbolic emotional content to musical keys [32].

v. Tempo

The tempo is the speed at which a song is played and is measured in beats per minute (BPM). Tempo plays a significant role in shaping the emotional quality of music. Fast tempo generally conveys energy, excitement, and intensity. Moderate tempo often feels relaxed or reflective. Moderate tempo can create a sense of balance and are common in pop, jazz, and some folk music. Slow tempo is often associated with calmness, sadness, or reflection.

The tempo of a song, measured in beats per minute (BPM), is the speed at which the music is played. BPM plays a critical role in determining the energy level of a song; slower tempos (60-80 BPM) are often associated with relaxing or melancholic moods, while faster tempos (120-140 BPM) convey excitement, joy, or intensity [10]. By analyzing the BPM of songs, this project can further refine its mood classification capabilities, matching songs with similar tempos within the same playlist to maintain a consistent energy level. Understanding BPM also allows for smoother transitions between playlists, ensuring that listeners experience a steady, cohesive flow that aligns with their desired mood.

Through the combined analysis of pitch, key, spectrogram, and BPM, this project aims to create a sophisticated recommendation system that considers both the musical and emotional elements of songs. Each of these factors contributes to the system's ability to classify songs effectively by mood, resulting in playlists that offer a tailored, immersive listening experience.

IV. Mathematical Background

i. Clustering

Clustering is a statistical method used to group a set of data points into clusters or categories based on their similarities. In the context of this project, clustering is a crucial technique for organizing songs into mood-based groups, allowing for effective classification and recommendation of music by emotional tone. Clustering algorithms, such as k-means, hierarchical clustering, and DBSCAN (Density-Based Spatial Clustering of Applications with Noise), are commonly used to identify patterns and group data points with shared characteristics. By leveraging these methods, the system can classify songs with similar musical features into cohesive groups, each corresponding to a specific mood.

In clustering, each song is represented by a set of features such as pitch, key, spectrogram characteristics, and BPM, that define its uniqueness. The clustering algorithm then analyzes these features across multiple songs to identify similarities, grouping songs that share certain attributes. For example, songs with similar tempos, frequency distributions, and harmonic structures might be clustered together as "relaxing" or "energetic”, “happy” or “sad”.

This clustering approach is particularly valuable in creating mood classification, as the method can group songs dynamically rather than relying on predefined genre labels. The system can adjust clusters based on patterns within the data, providing more flexibility and adaptability. Moreover, clustering methods are capable of handling high-dimensional data, which is essential given the complexity and variety of musical features analyzed in this project.

a. K-Means clustering

K-Means clustering is a partitioning method, which groups datapoints into a predetermined number of distinct clusters. Firstly, cluster centers are randomly selected, and all data points will be grouped to the nearest cluster. Then the centroid values - average distance of each object within a cluster from the cluster's center are recalculated given new group of datapoints, and datapoints again are reassigned to the nearest cluster after calculation. Centroids will keep moving and this process repeats until cluster centers stop moving [11].

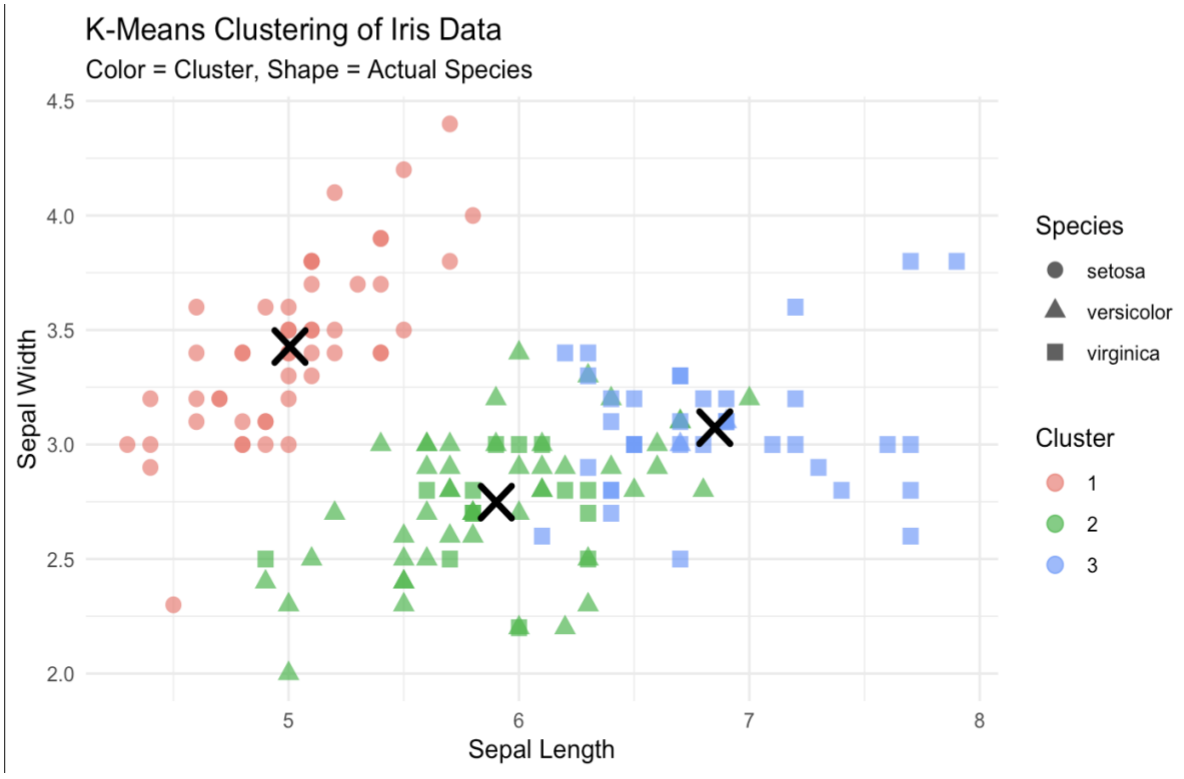


Figure 4.1.1: K-means on Iris Dataset with cluster centers and misclassification by shape

The K-means clustering model applied to the Iris dataset aims to group the data into three clusters based on four features: Sepal Length, Sepal Width, Petal Length, and Petal Width. K-means is an unsupervised learning algorithm that partitions the dataset into K clusters by minimizing the variance within each cluster. It starts by selecting K random centroids, assigns each data point to the nearest centroid, then iteratively updates the centroids based on the mean of the assigned points until convergence. This results in natural groupings of data points based on feature similarity. In this case, the model groups the Iris flowers into three clusters, which are expected to correspond to the three known species (Setosa, Versicolor, and Virginica), though the algorithm does not use species labels during training. The clustering results are visualized to analyze patterns and evaluate performance against the true species labels.

Predicted \ Reference	**setosa**	**versicolor**	**virginica**
setosa	50	0	0
versicolor	0	48	14
virginica	0	2	36

Figure 4.1.2: Confusion Matrix of K-means clustering on Iris dataset

Metric	**Value**
Accuracy	0.8933
95% CI (lower)	0.8326
95% CI (upper)	0.9378
No Information Rate	0.3333
P-Value [Acc > NIR]	< 2.2e-16
Kappa	0.84

Figure 4.1.3: Overall Statistics of K-means clustering on Iris dataset

The K-means clustering model applied to the Iris dataset groups the data into three clusters based on four numerical features, achieving an accuracy of 89.33%. The confusion matrix shows that Setosa is perfectly classified, while some misclassification occurs between Versicolor and Virginica, likely due to overlapping feature distributions. The Kappa statistic of 0.84 indicates a strong agreement between predicted clusters and actual species labels, confirming that K-means effectively captures natural groupings despite being an unsupervised learning algorithm.

b. Fuzzy C-Means Clustering

Fuzzy means clustering, also known as fuzzy c-means clustering, is an extension of the traditional k-means clustering algorithm that allows for greater flexibility by assigning data points to multiple clusters with varying degrees of membership. Unlike conventional clustering methods, where each data point belongs to only one cluster, fuzzy means clustering acknowledges that some data points may exhibit characteristics of multiple clusters simultaneously [13]. This approach is particularly useful in applications, such as music classification, where song attributes can overlap across different moods.

In fuzzy means clustering, each data point is assigned a membership value between 0 and 1 for each cluster, indicating the degree to which it belongs to that cluster [13]. The algorithm iteratively optimizes these membership values based on the distance between each data point and the centroids (central points) of the clusters. This method provides a more nuanced classification by allowing songs to belong to multiple mood clusters with varying degrees of confidence, which is valuable in cases where songs don’t fit neatly into a single mood category.

For example, a song might have a relaxing tempo but an energizing beat, placing it somewhere between a "calm" and an "energetic" mood. Fuzzy means clustering would assign partial memberships to both the calm and energetic clusters, capturing the song’s complex emotional qualities more accurately than hard clustering methods. This approach enables the recommendation system to recognize and handle the subtle emotional variations within songs, providing a more tailored and responsive listening experience.

Fuzzy means clustering also enhances the adaptability of the classification system, as it can dynamically adjust to the multifaceted nature of music. Instead of forcing a song into a single category, the system can classify it across multiple moods, allowing for richer playlist options that reflect the full spectrum of a listener's emotional preferences. This flexibility is key to creating mood-based playlists that feel cohesive while accommodating the natural diversity of musical expression.

ii. Neural Networks

Neural network is inspired from biological brain neurons and how they work. The first artificial neuron network architect comes from computational model of biological neurons work in animal brain to perform complex computation using propositional logic [12]. Neural network is versatile and scalable, which makes it ideal to tackle complex machine learning tasks. Some popular applications include image classification, speech recognition, and recommending algorithms.

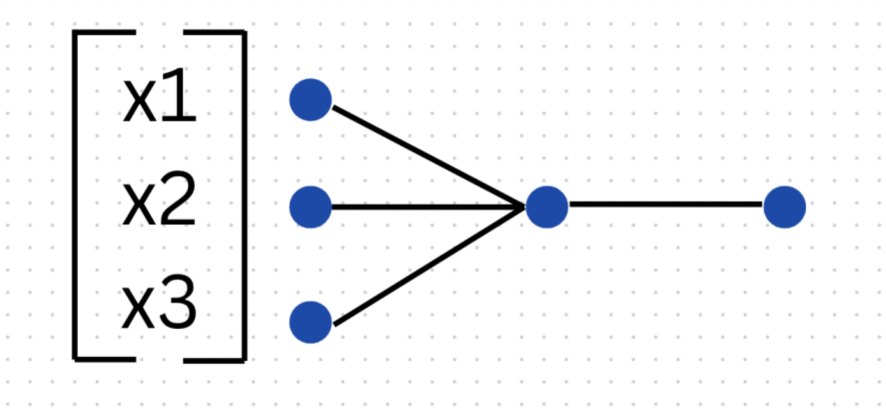


Figure 4.2.1: Visual representation of how a neuron works

A neuron can take multiple inputs but produce only one output. Each input is multiplied with its own weight and added up with a bias $x_1w_1 + x_2w_2 + x_3w_3 + x_4w_4$; where w_1, w_2, w_3 are weights for each input, and w_4 is bias.

$$\begin{matrix} x_1 \\ x_2 \\ x_3 \\ 1 \end{matrix} \cdot \begin{matrix} w_1 \\ w_2 \\ w_3 \\ w_4 \end{matrix}$$

We can write the above equation in the form of vector and calculate dot product function by adding value 1 for the first vector to match with bias w_4. The result of this function will pass to an activation function, in this project we use ReLU, which will be explained in the next section.

Key components in a neural network model are perceptron, parameters, activation function, and optimizer.

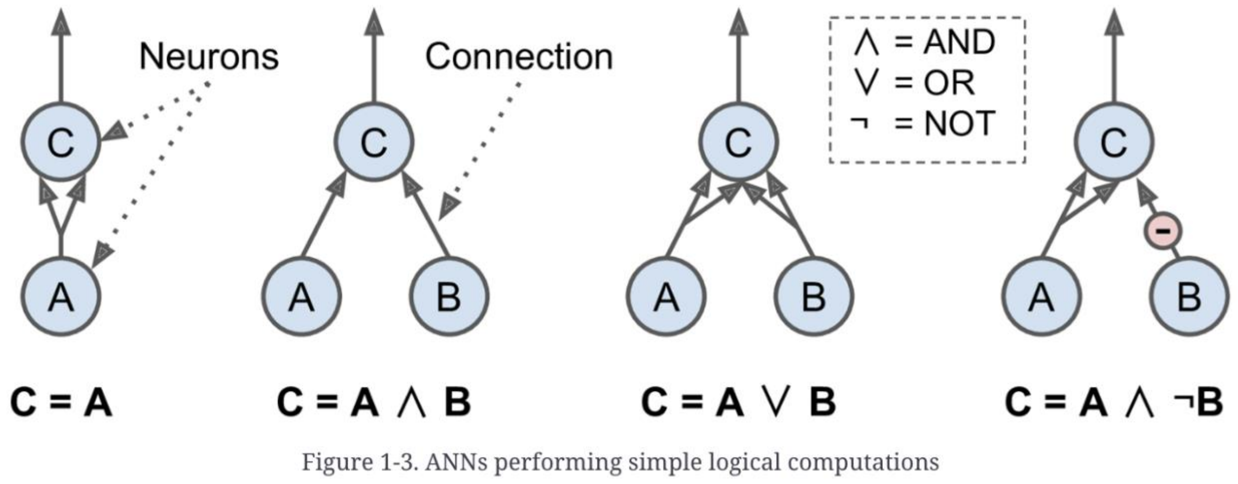


Figure 4.2.2 Logical computation for neurons [12]

The first network is an identity function – a simple form of activation function. When neuron A gets activated, neuron C gets activated because neuron C receives two input signals from neuron A; When neuron A is off, neuron C is off. A gives 2 signals to C, showing C's activation is solely dependent on A.

The second network represents logical (AND). Neuron C is only activated when both neuron A and neuron B are activated. Both signals must be activated at the same time to activate neuron C.

The third network demonstrates logical (OR). Neuron C can be activated when either neuron A and B is activated, or both.

The last neuron model presents a combination of logical (AND) and logical (NOT). Neuron C is activated when neuron A is activated, and neuron B is not activated. If neuron A is always activated, then neuron C is active when neuron B is not activated and vice versa.

Perceptron:

Perceptron is the simplest artificial neuron network architectures, where instead of binary inputs and outputs, the model takes numeric neuron. This is called Linear Threshold Unit (LTU). LTU computes a weighted sum of its inputs, then applies a step function. Step function is a constant function of real numbers when they can be written with finitely many pieces. A perception is formed by single layer of LTUs where each neuron connects to all inputs [12].

Parameters:

Internal parameters include hidden layers: A list of numbers represents the internal structure of neural network, where each number is a hidden layer. Total number of internal parameters is total of weights and biases. Total weights are sum of the product of each pair of adjacent layers. Total biases are total of hidden neurons and output neurons.

For example, if we have 10 input neurons, 3 hidden layers with 4,3,2 neurons, and an output of 2 neurons.

1. Input layer (10) → First hidden layer (4)
   - Number of weights: 10×4=40
2. First hidden layer (4) → Second hidden layer (3)
   - Number of weights: 4×3=12
3. Second hidden layer (3) → Third hidden layer (2)
   - Number of weights: 3×2=6
4. Third hidden layer (2) → Output layer (2)

- Number of weights: 2×2=4

Total weights = 40+12+6+4=62

Each neuron (except input neurons) has a bias.

1. First hidden layer: 4 biases
2. Second hidden layer: 3 biases
3. Third hidden layer: 2 biases
4. Output layer: 2 biases

Total biases = 4+3+2+2=11

Total parameters (weights + biases) = 62+11=73

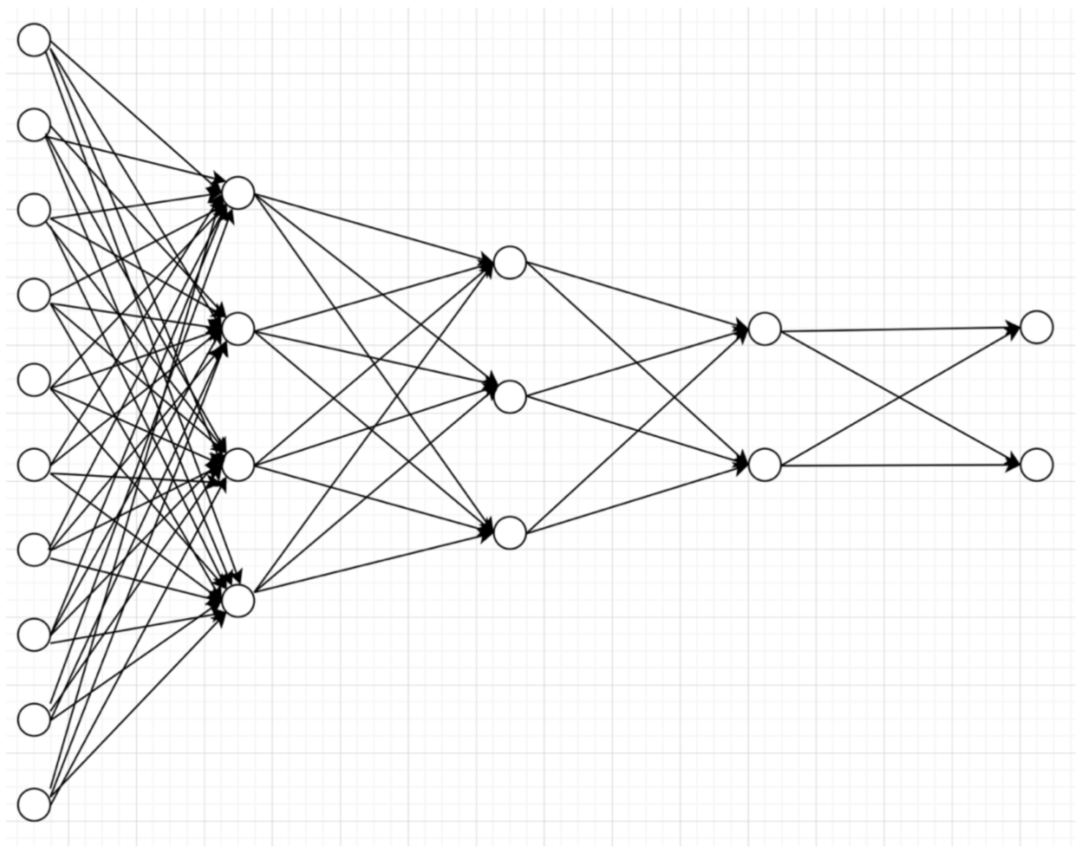


Figure 4.2.3: Visual representation of how neural network hidden layers work

Backpropagation

Backpropagation describes weights and biases getting updated if there is an error in prediction. Based on how brain functions, if a neuron often triggers another neuron, the connection between those 2 neurons will be stronger. Weight between two neurons increases if they have correct output (same label). This is the basic of how to train perceptron. A fascinating feature of neural network when it comes to training is if the model predict result is an error, it can unlearn the connection between neurons leading to the wrong output, instead, it reinforces connections weights of outputs that would have produced correct outputs.

Weight initialization techniques: He Uniform Initialization

It is crucial to have an efficient weight initialization technique to ensure model accuracy. He Uniform Initialization technique pairs well with ReLU Activation function which will come up in the next passage. For this technique, weights are assigned from values of a normal distribution. The scale of the random values in the distribution follows by:

$$\sqrt{\frac{2}{number\ of\ inputs}}$$

Activation function:

Activation function is a mathematical function that introduces non-linearity on the model, allowing the network to handle complex pattern. Neural network behaves like a linear regression model without this activation function. Activation functions is calculated by the weighted sum of inputs and adding a bias term.

ReLU (Rectified Linear Unit) Function

ReLU function is defined by A(x)=max(0,x), this means that if the input x is positive, ReLU returns x, if the input is negative, it returns 0. The advantage is ReLU costs less computation expenses because the mathematical operations are simpler, and few neurons are activated at one time, so the network is sparse and easier to compute. Another advantage of this activation form is non-saturating [15].

In the example below, ReLU returns 1.66. After each layer, the output of previous layer becomes the input for the next layers.

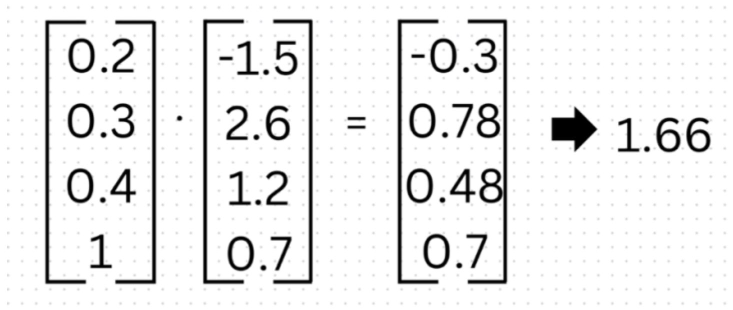


Figure 4.2.4: Representation of ReLU computation

Softmax activation function:

Softmax is ideal for multiclass classification and probabilistic interpretation, which takes a vector of raw outputs and converts them to probability scores [14].

$$Softmax(z)_i = \frac{e^{z_i}}{\sum_{j=1}^{N} e^{z_j}}$$

- Z: vector of raw outputs from the neural network
- e ≈ 2.718
- N: number of classes
- i-th predicted probability of the test input in class i [14].

In this project, I use ReLU activation function for first, second and third hidden layers, and Softmax for the output layer.

Optimizer

SGD: Stochastic Gradient Descent

Gradient Descent is an iterative optimization process that searches for an objective function's optimum value (Minimum/Maximum). It is one of the most used methods for changing a model's parameters to reduce a cost function in machine learning projects.

The Stochastic Gradient Descent (SGD) loop iteratively updates model parameters to minimize the cost function until convergence or reaching a predefined iteration limit. First, the training dataset is shuffled to introduce randomness, preventing the model from learning in a fixed order and improving generalization. Then, the algorithm iterates over each training example (or a small batch) in the shuffled order, computing the gradient of the cost function with respect to the model parameters. Using this gradient, the parameters are updated by taking a step in the negative gradient direction, scaled by a predefined learning rate. After each update, a convergence criterion is evaluated, such as the change in the cost function between iterations, to determine if further optimization is needed. Once the algorithm meets the convergence condition or the maximum number of iterations is reached, it returns the optimized model parameters, finalizing the training process [16].

Principal component analysis (PCA)

PCA aims to reduce the dimension of a large dataset by selecting most important and representative variables. PCA starts with standardizing the data, then finding relationships by covariance matrix. Directions between data points are calculated using Eigenvectors and Eigenvalues. Eigenvectors find the axes of direction where data spreads out; Eigenvalue ranks the importance of the directions [17].

PC1: Direction of maximum variance, captures the most important information

PC2: Second best direction, orthogonal to PC1

Then, we select the top directions which capture around 95% of the variance. This method works well with my high dimensional datasets. PCA improves feature extraction, noise reduction, and data preprocessing before feeding neural network model.

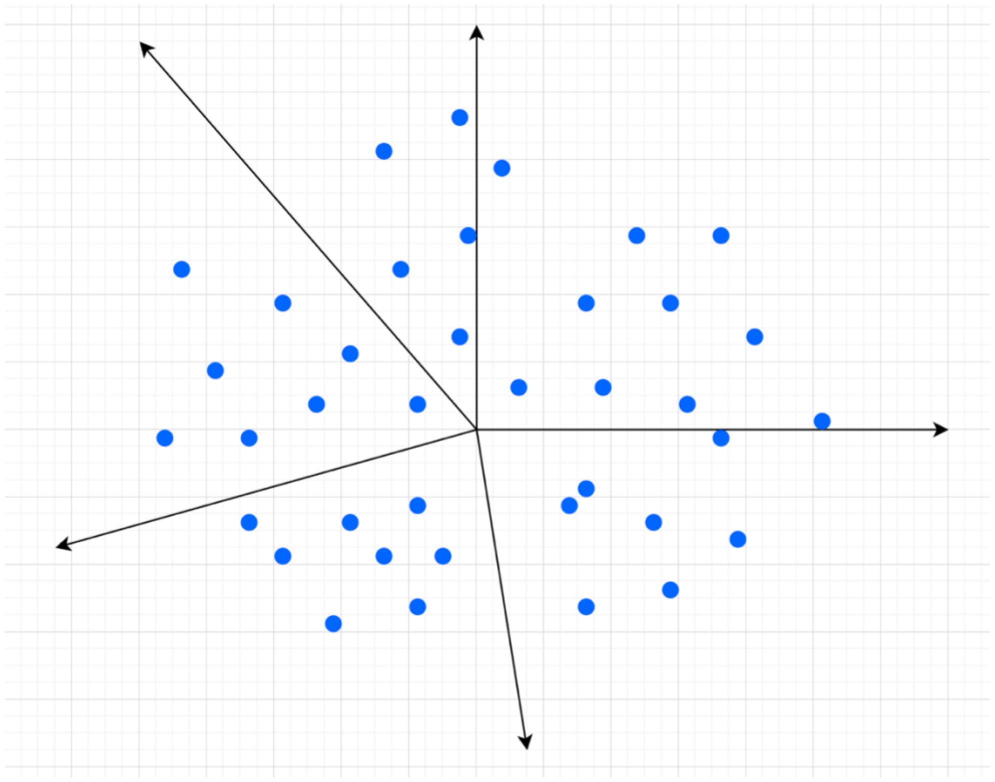


Figure 4.2.5: a random 5D Dataset

Assume we have a 5D dataset like the graphic above. First, we calculate a covariance matrix with n=5. This matrix presents the covariance between each pair of variables. Positive covariances often suggest a positive relationship, and otherwise. If covariances equal to 0, the 2 variables do not vary together.

Covariance Matrix

$$\begin{bmatrix} \mathrm{Var}(x_1) & \cdots & \mathrm{Cov}(x_{n,} x_1) \\ \vdots & \ddots & \vdots \\ \mathrm{Cov}(x_{n,} x_1) & \cdots & \mathrm{Var}(x_n) \end{bmatrix}$$

Figure 4.2.6 Covariance Matrix [41]

Next step is conducting a linear transformation to find eigenvectors(direction) and eigen values(magnitude). PC1 the eigenvector corresponding to the largest eigenvalue. PC2 is the eigenvector corresponding to the second-largest eigenvalue.

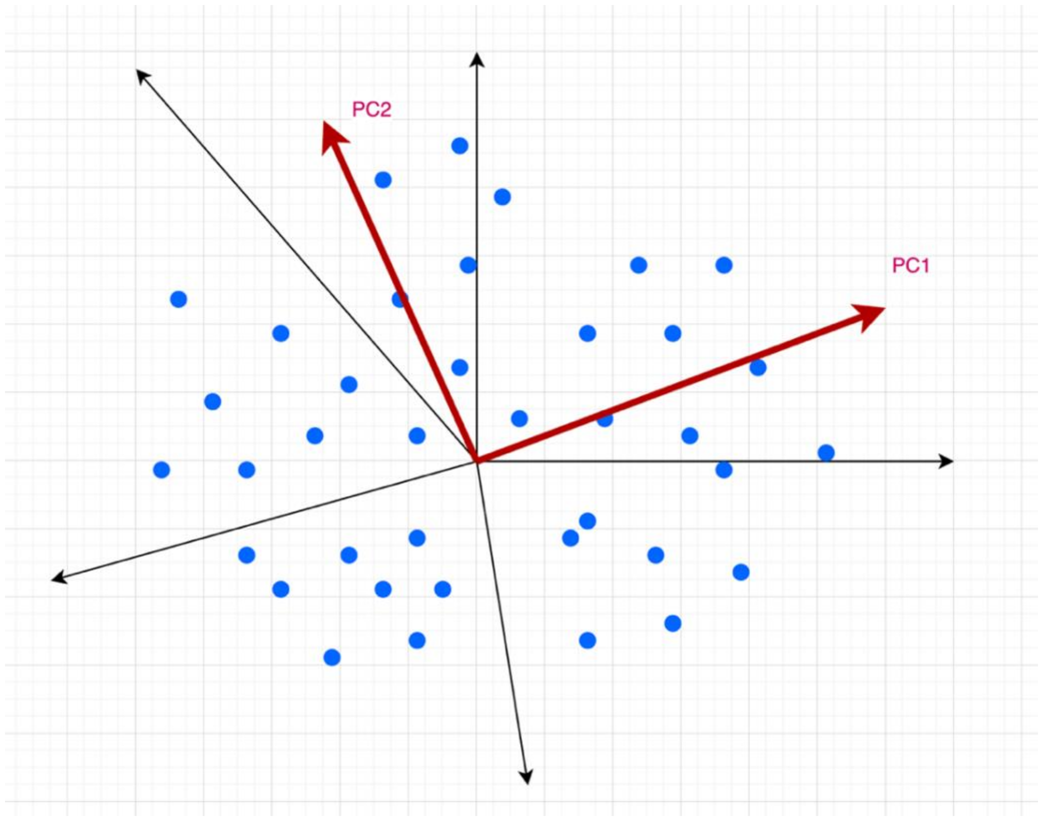


Figure 4.2.7: PC1 and PC2 on a 5D dataset

For more information on calculating covariance matrix, read more at: Covariance Matrix [41].

For more information on calculating eigenvalues and eigenvectors, read more at: Eigen values [42].

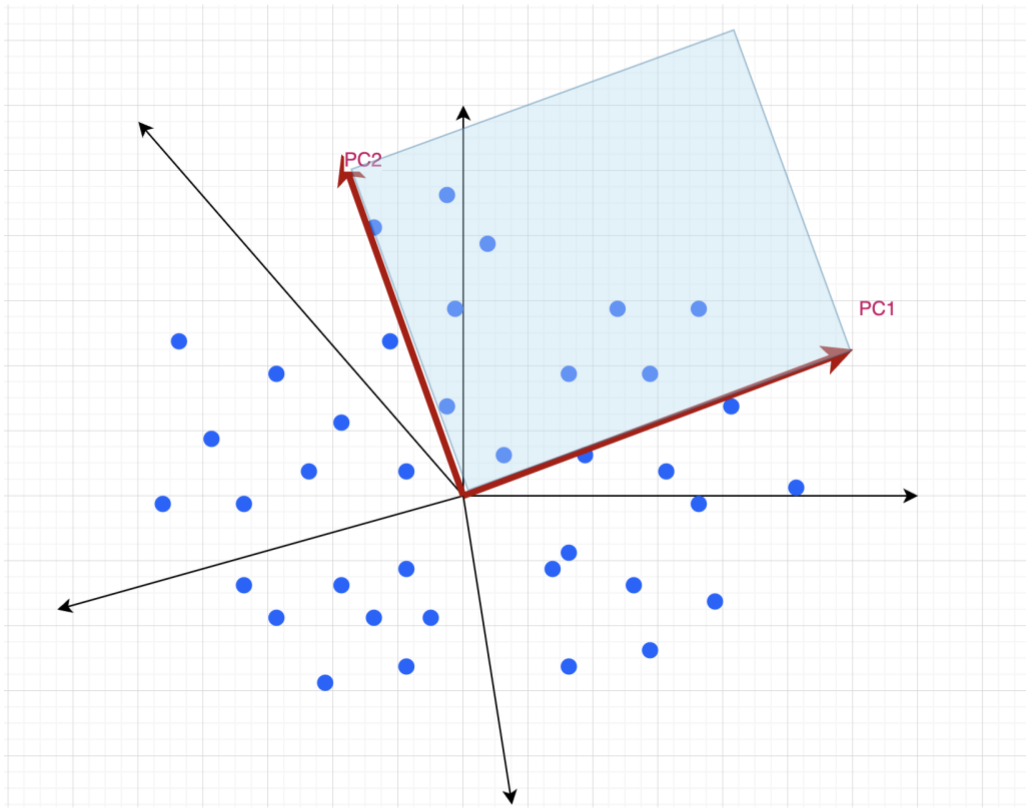


Figure 4.2.8: Plane of data points after PCA

The light blue plane bounded by PC1 and PC2 is the new plane of data points after dimensional reduction.

Loss Computation

Loss computation is a fundamental step in training neural networks, as it quantifies how far the model's predictions are from the actual values. In classification problems, the cross-entropy loss is widely used because it effectively measures the difference between probability distributions.

For multi-class classification tasks, categorical cross-entropy is commonly used when labels are provided in a one-hot encoded format [18]. However, when labels are represented as integer class indices instead of one-hot vectors, sparse categorical cross-entropy is preferred, which I used for training in this project. This loss function calculates the error by comparing the model's predicted probability distribution with the correct class index without requiring one-hot encoding, making it more memory-efficient and convenient for large datasets.

During training, the computed loss is used to update model parameters through backpropagation and an optimization algorithm like Stochastic Gradient Descent (SGD) or Adam. By minimizing the loss over multiple epochs, the model improves its accuracy and generalization ability.

V. Dataset

The dataset for this project was collected from the FMA (Free Music Archive) website, consisting of MP3 audio files. Eight distinct music genres were selected for analysis: Blues, Electronic, Hiphop, International, Jazz, Pop, Rock, and SoulRnb. To ensure balance and unbiases, each genre contains more than 50 different songs in the final dataset. Key music features were extracted to facilitate classification. These include bpm, key and scale, and spectral centroid for insights on overall brightness of songs. Songs will be categorized under 4 different moods. In addition to these features, Mel-spectrograms were generated for each song after thorough audio processing techniques, with a remaining of 389 songs. The Mel-spectrograms, which represent the frequency domain of the audio signals over time, were analyzed in vector form and integrated with the other extracted features into a consolidated array. Lastly, mood labels were given subjectively by me. This combined dataset serves as the basis for the classification and recommendation system developed in this project.

i. Data Preparation

After preprocessing the MP3 audio files, we extract key musical features such as key, beat, tempo, strength, scale, spectral centroid, and Mel-spectrogram for each song. To handle categorical data, we one-hot encode key and scale and encode the mood labels. Since the Mel-spectrogram is an image, we convert it into a numerical array and merge it with the other extracted features. The resulting arrays are extremely large, containing over 500,000 elements, making dimensionality reduction essential. To address this, we apply Principal Component Analysis (PCA) to reduce complexity while retaining key information. For clustering models, we exclude the mood variable, focusing only on the extracted audio features to allow for unsupervised pattern discovery in the data. The final dataset is stored under pickle (pkl) format.

ii. EDA

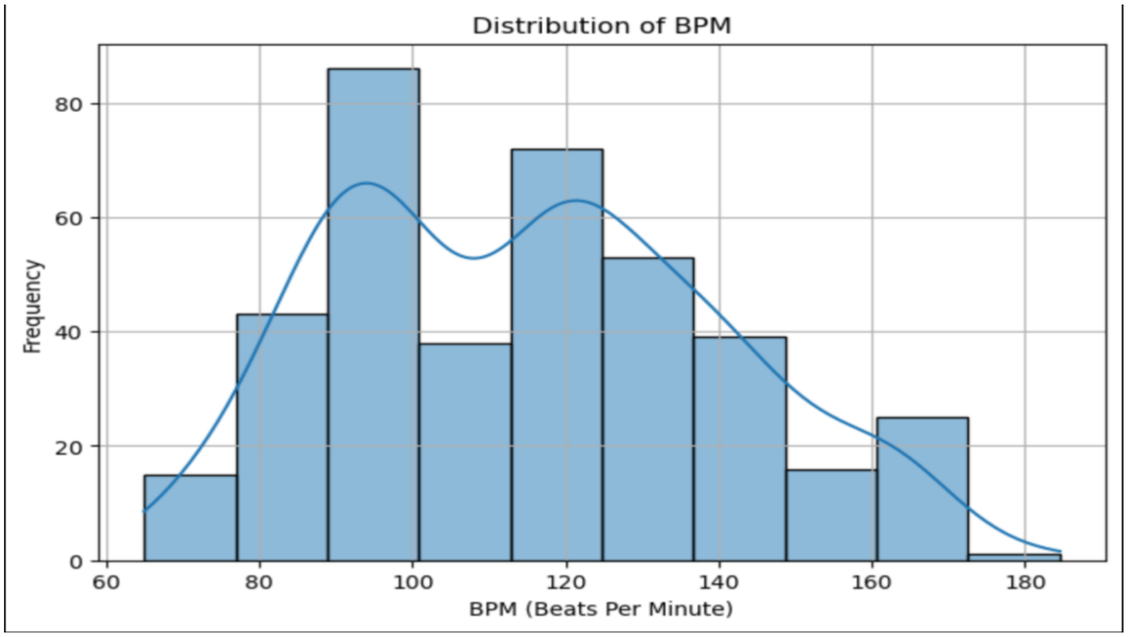


Figure 5.2.1: Distribution of Song Tempos in Beats Per Minute (BPM)

This histogram represents the tempo of songs, showing peaks around 90-100 BPM and 120-130 BPM, indicating common tempo ranges. The bimodal nature of the distribution suggests that two distinct tempo groups are prevalent.

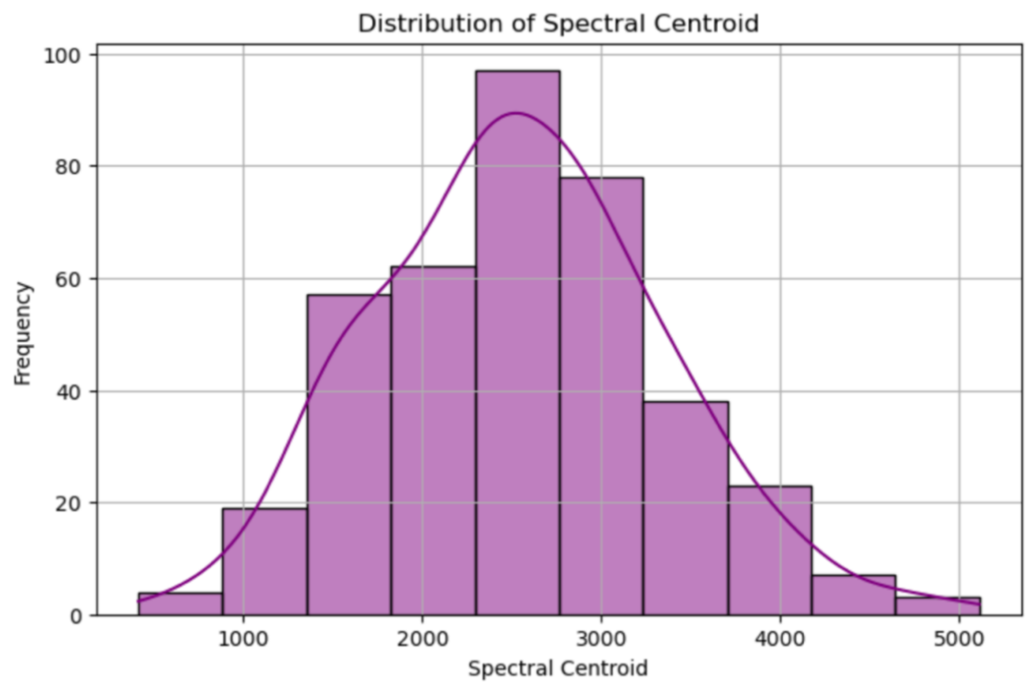


Figure 5.2.2: Spectral Centroid Distribution: Brightness Characteristics of Songs

This histogram visualizes the spectral centroid, which measures the brightness of a sound. The distribution follows a bell-shaped curve, suggesting that most songs cluster around a mid-range frequency.

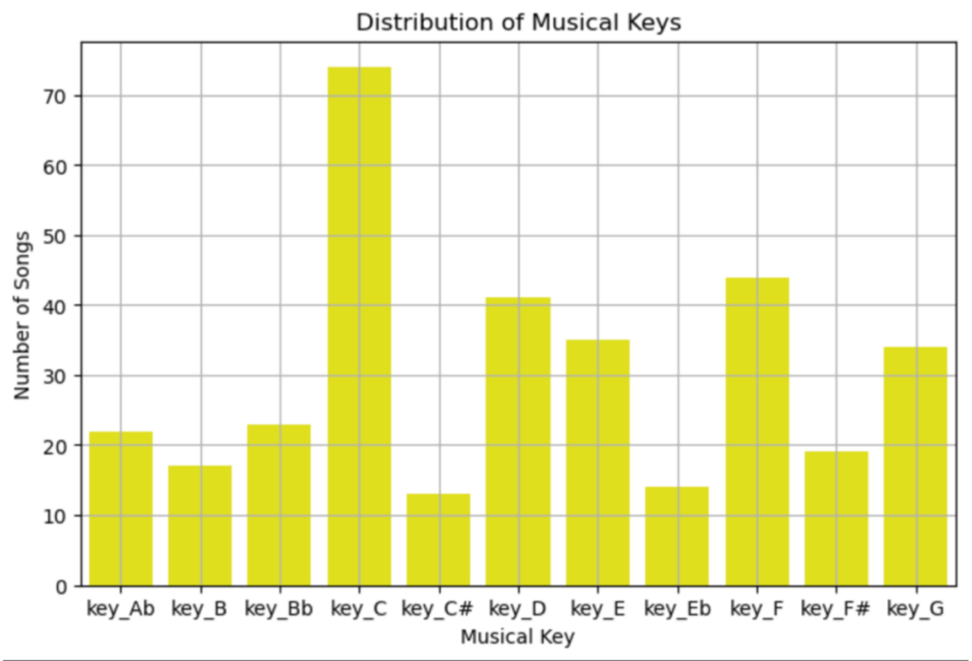


Figure 5.2.3 Distribution of Musical Keys in the dataset

This bar chart illustrates the frequency of different musical keys. Key C appears most frequently, followed by F and D, while keys like C# and Eb are less common, indicating key preference trends in the dataset.

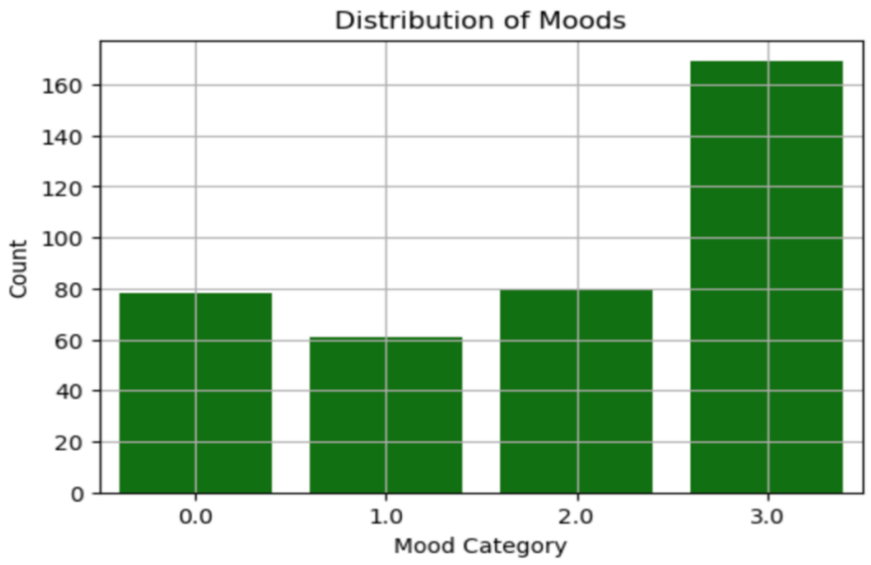


Figure 5.2.4: Distribution of Labeled Music Moods

This bar chart categorizes songs into different mood categories (0 to 3), showing that Mood Category 3 is the most common in our random selected dataset, while others have relatively balanced distributions.

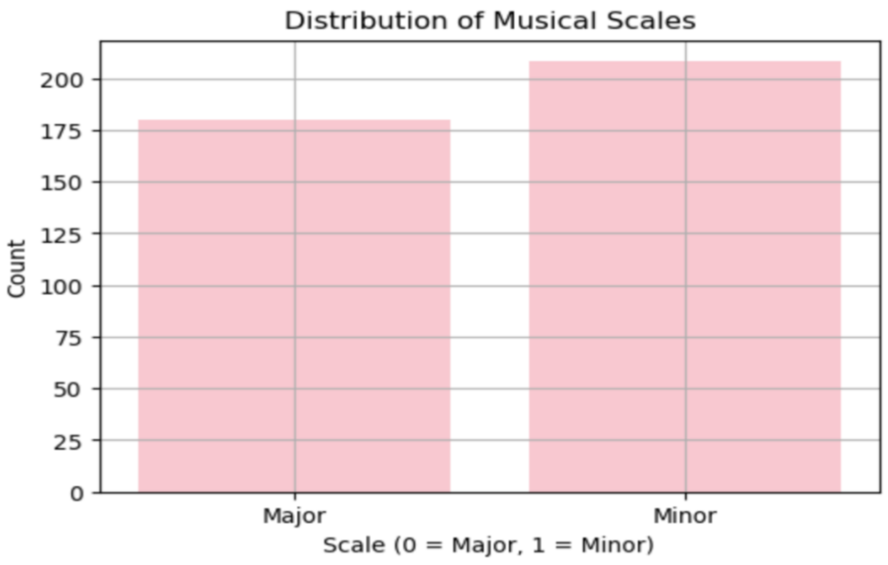


Figure 5.2.5 Distribution of Major and Minor Scales in Songs

The bars indicate that Minor scales appear more frequently than Major scales, suggesting a higher proportion of minor-key pieces in the dataset.

VI. Audio Preprocessing

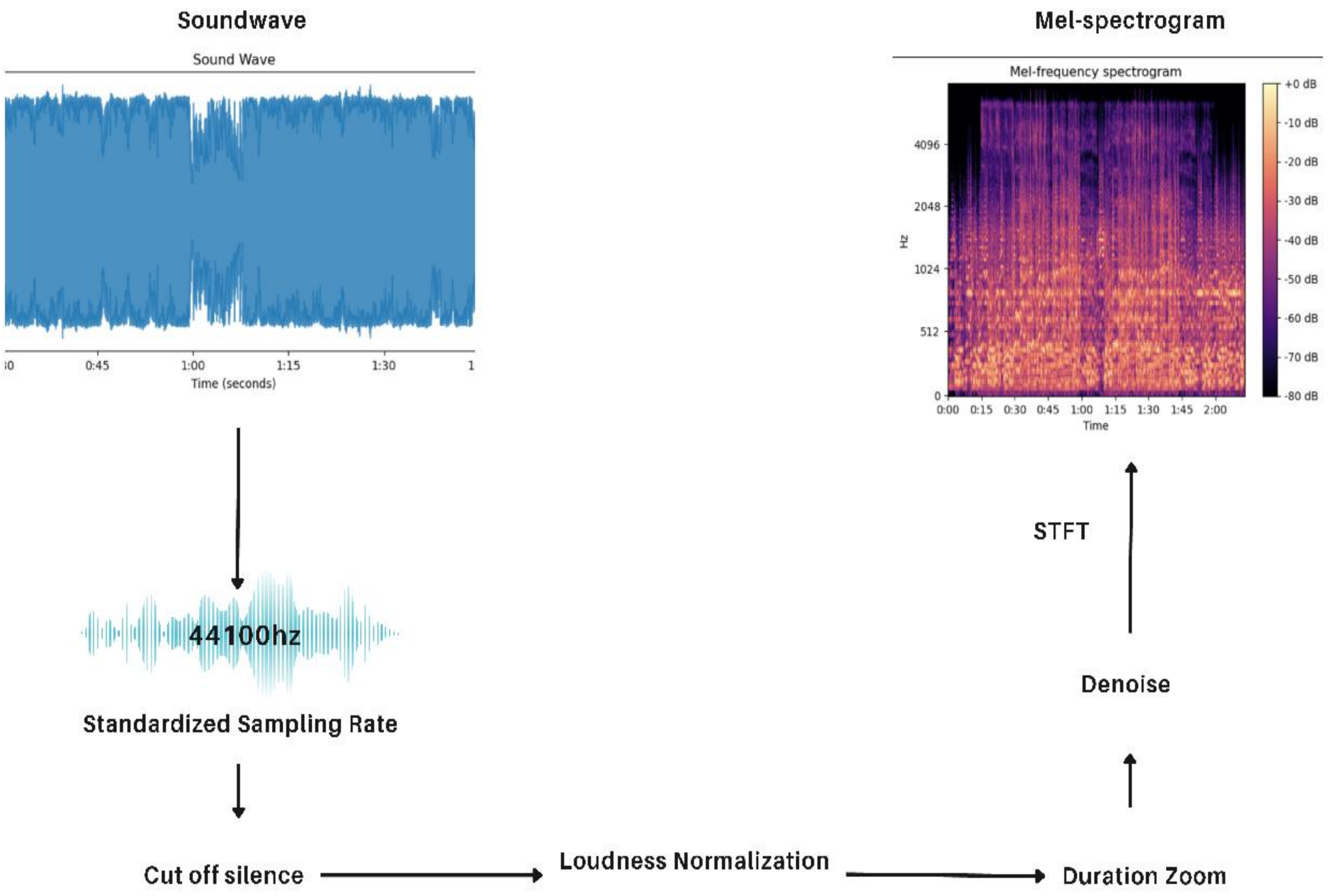


Figure 6.1: Audio signal preprocessing diagram

It is important to conduct audio preprocessing before extracting important features to feed in machine learning techniques. We will conduct noise reduction, silence removal, loudness normalization (a strategy to adjust volume to a standard level by dividing the signal sequence with the highest value of signal), Short-time Fourier transform, and resampling. [24]

i. Resample

The sample rate (or sampling rate) is the number of samples taken per second. The units for sample rate are samples per second (sps) or Hertz (Hz). In signal processing, sampling is the technique to reduce a continuous time signal to discrete time signal. An example is the conversion of a soundwave to a sequence of samples.

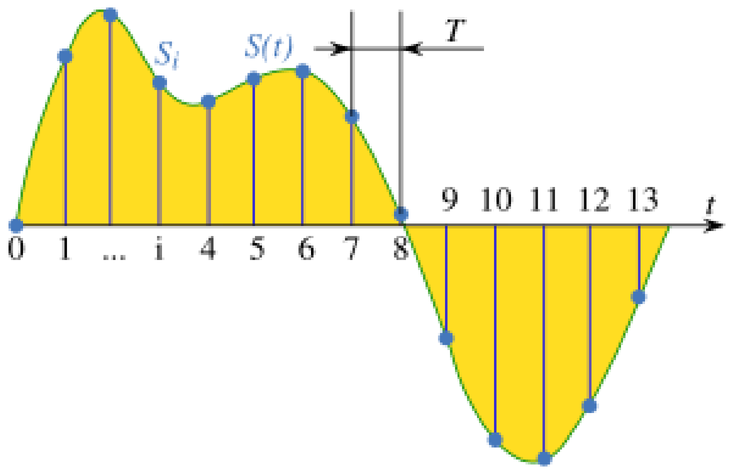


Figure 6.1.1: Conversion of Soundwave [48]

Audio signals are subject to the same criteria as other analog signals. Humans can hear sounds in the 20-20,000 Hz range, so music and other sound waves are often sampled at 44.1kHz or 48kHz. Audio is also sometimes recorded at 88.2kHz or 96kHz in a process known as oversampling, wherein the sample rate is taken to be well over the Nyquist frequency (double the recorded range) to improve resolution and signal-to-noise ratio [25]. For all the audio in this project, I keep their sampling rates at 20khz for the best representations of Mel-spectrograms.

ii. Trimming

Trimming is an essential step in audio preprocessing that involves removing unwanted silence or noise from the beginning and end of an audio file. Many raw audio recordings contain leading or trailing silence, which can interfere with feature extraction and affect the performance of machine learning models. By eliminating these silent or low-energy segments, trimming ensures that the processed audio focuses only on relevant sound information, reducing computational load and improving classification accuracy. This is particularly useful in music processing, where silence at the start or end of a track does not contribute to mood recognition. Automatic trimming can be performed using energy-based thresholding, where samples below a certain amplitude are discarded.

iii. Zooming

Zooming in audio processing refers to enhancing specific sections of an audio waveform by focusing on frequency ranges, time intervals, or signal details. This technique is useful when analyzing key features such as beats, pitch variations, or transient sounds that may be less noticeable in the full waveform. By zooming into relevant portions of an audio signal, researchers can perform detailed spectrogram analysis or fine-tune feature extraction, particularly for applications like speech recognition, music emotion classification, and sound event detection. Zooming can be implemented by time windowing or frequency filtering, where only specific segments or frequency bands are selected for further analysis. In this project, we zoom in by duration, from the 30th second to 2:15 minutes for all songs.

iv. Denoising (Spectral Subtraction)

Denoising is a crucial step in audio preprocessing that removes unwanted background noise to enhance the clarity of the signal. Few denoising techniques were tested for stationary and non-stationary noise such as spectral gating, Wiener Filtering, and spectral subtraction. Spectral subtraction is tested to be the most effective noise reduction technique for this project, particularly for stationary or predictable noise, as it does not over remove informative sounds.

This method works by estimating the noise spectrum from a silent or low-energy segment of the audio file, then subtracting this noise estimate from the overall signal. The process is typically performed in the frequency domain, using Short-Time Fourier Transform (STFT) to analyze and modify the spectral components. Spectral subtraction is widely used in speech enhancement, music preprocessing, and environmental sound classification, as it significantly improves signal quality without severely distorting the original audio.

v. Loudness normalization

To normalize loudness of songs, first we measure the loudness of each audio file, then calculate a global target loudness as the average loudness across all files. Then, loudness of each audio files is adjusted to match the global target.

vi. Short-time Fourier transform

Fourier transform (FT) is a mathematical expression to turn data from time domain to frequency domain. Fourier transform decomposes a signal into a combination of sine wave of different amplitudes and frequencies. FT is one of the most basic signal processing tools [26].

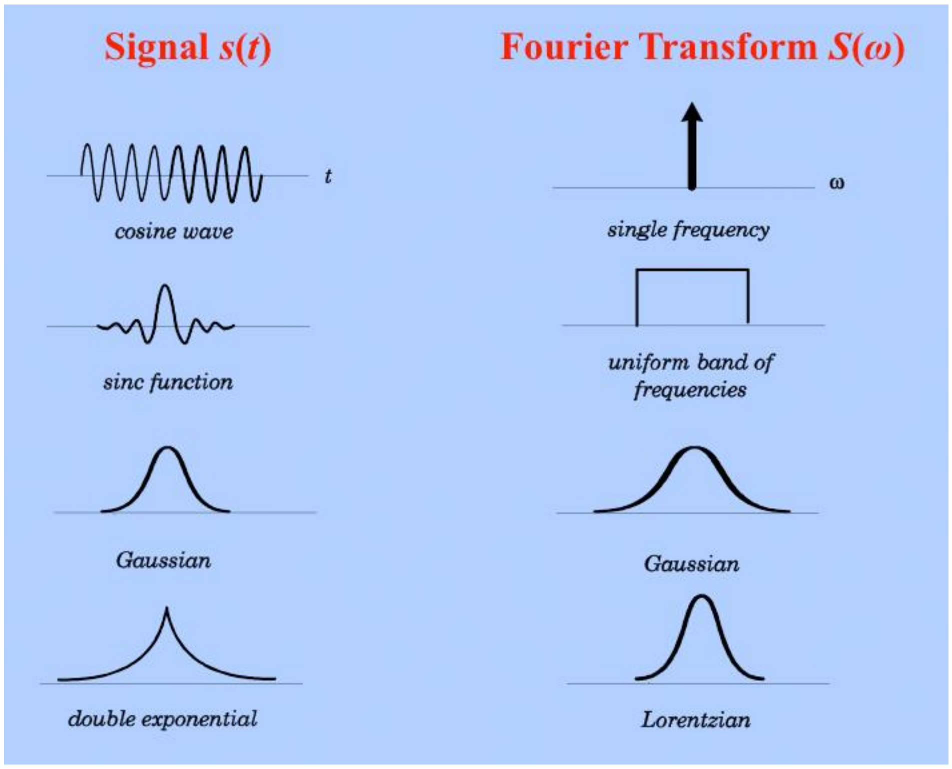


Figure 6.6.1: Fourier transform from time domain to frequency domain [46]

Fourier demonstrated that any periodic signal *s(t)* can be represented as a combination of sine waves with different amplitudes, frequencies, and phase shifts [46].

$$s(t) = a_0 + a_1 \sin(wt + \emptyset_1) + a_2 \sin(2wt + \emptyset_2) + \cdots \text{ [46]}$$

where a_i represent the amplitudes, $\emptyset_i$ denote the phase shifts, and w_i is the fundamental frequency. The multiples of this frequency, such as 2w, and so on, are referred to as harmonics.

To transform to frequency domain S(w), we apply Fourier transform on *s(t):*

$$S(w) = \int_{-\infty}^{\infty} s(t)e^{-iwt}dt$$

Fourier Transform takes a whole audio signal and breaks it down into its frequency components, but it does this for the whole signal at once, so we know which frequencies exist, but not when they happen. FT assumes the signal is infinite and stationary.

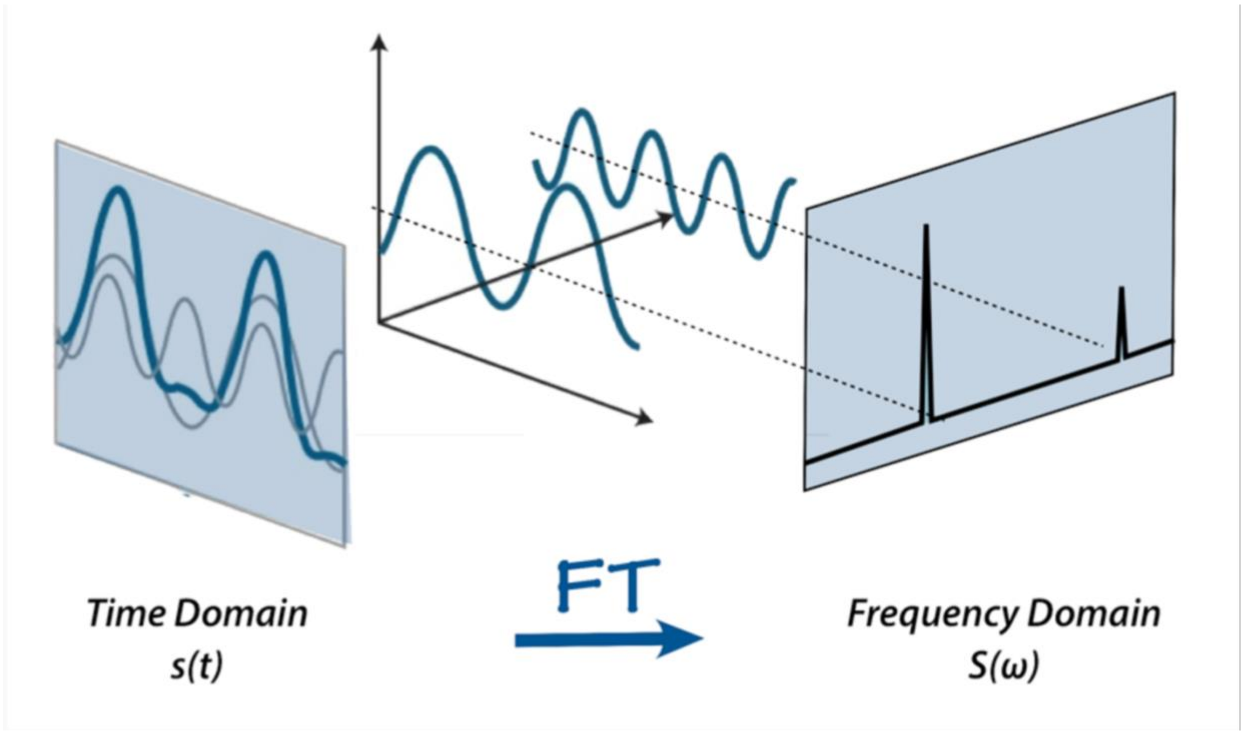


Figure 6.6.2: Visualizing the Fourier Transform: From Time Domain to Frequency Domain [46]

Discrete Fourier Transform (DFT) computes the FT on signals. DFT analyzes a finite segment of the signal instead of infinite length and returns a set of frequency components present in that segment.

Short-time Fourier transform (STFT) is deviated from Fourier transform. The procedure is to divide a longer time signal into shorter and overlapping segments of equal length, which is called time windows. Each window is multiplied with a windowing function that helps smooth the edges of each small segment as we divide signal into short chunks which can cause sharp transitions. Then DFT is applied separately on each shorter segment. It provides a detailed view of both time and frequency, making it useful for non-stationary signals where frequency content changes over time. Fourier Transform lacks time localization, making it difficult to analyze non-stationary signals. STFT can fix this problem by adding a time parameter to the base function [27].

$$X(\tau, w) = \int_{-\infty}^{\infty} x(t)w(t-\tau)e^{-iwt}\mathrm{dt}$$

Where $X(\tau, w)$ is STFT; x(t)w(t−τ) is phase and magnitude of the signal over time and frequency; w(τ) is a window function; x(t) is signal [37].

We use STFT in audio processing, specifically mood classification because of its advantage providing insights for temporal dynamics, frequency evolution, and rich features. Firstly, moods in music often depend on changes in intensity, rhythm, and pitch across time. STFT preserves this temporal variation. Secondly, music moods may be influenced by the buildup of frequencies, for example, rising strings in a sad mood or sharp beats in an energetic track. STFT analyze how the frequency content of a signal evolves over time. Lastly, STFT allows time-frequency features extraction like Mel-spectrogram, Chroma Features, and spectral centroid. STFT is computed through DFT – Discrete Fourier Transform. The output of STFT is a matrix, and often visualized into Spectrogram or Mel-spectrogram in form of heatmap.

VII. Models

Feature Selection and Dimensionality Reduction for Clustering and Neural Network

To prepare the dataset for machine learning, we first removed the mood_encoded variable, which serves as the target label, ensuring that only relevant input features were retained. Since raw features can have different scales, we applied StandardScaler – a Python function to normalize the data, transforming it into a standardized distribution with a mean of 0 and a standard deviation of 1. This step is essential for optimizing the performance of machine learning models, particularly those relying on gradient-based optimization methods, as it prevents features with larger magnitudes from dominating those with smaller scales.

To reduce computational complexity and eliminate redundant information, we performed Principal Component Analysis (PCA). Initially, we fitted PCA without specifying the number of components to analyze the cumulative explained variance and determine how much variance each principal component contributed. Based on this, we selected the optimal number of components that retained at least 95% of the variance, ensuring that the most significant patterns in the dataset were preserved while discarding noise. By reducing dimensionality, PCA improved model efficiency and mitigated potential issues related to overfitting and high computational costs. After applying PCA, the input feature space was reduced from approximately 500,000 variables to 339 principal components. The resulting data has a shape of (388, 339), where 388 rows represent individual songs after audio preprocessing, and 339 columns correspond to the principal components. This reduced data is stored in a 2D matrix.

i. Neural Network Architecture and Hyperparameter Selection

After feature extraction, we implemented a feedforward neural network for multi-class mood classification. The architecture consisted of

1. An input layer matching the number of principal components obtained from PCA.
2. Three hidden layers with 64, 32, and 16 neurons, respectively, each using the ReLU (Rectified Linear Unit) activation function and He initialization.
3. A dropout layer (30%) after the first hidden layer to reduce overfitting by preventing the network from relying too heavily on specific neurons.
4. An output layer with a softmax activation function, responsible for classifying songs into one of the predefined mood categories.

We selected ReLU as the activation function for the hidden layers because it introduces non-linearity while avoiding the vanishing gradient problem common in sigmoid and tanh activations [43]. Unlike these alternatives, ReLU does not saturate for positive values, allowing for faster training and improved performance in deep networks. Additionally, its sparsity-inducing behavior (outputting 0 for negative inputs) enhances computational efficiency [43].

For the output layer, we used softmax activation, which converts raw output scores into probabilities across the mood categories. Softmax ensures that the sum of probabilities across all output classes equals 1, making it ideal for multi-class classification as it allows the model to assign confidence scores to each mood category.

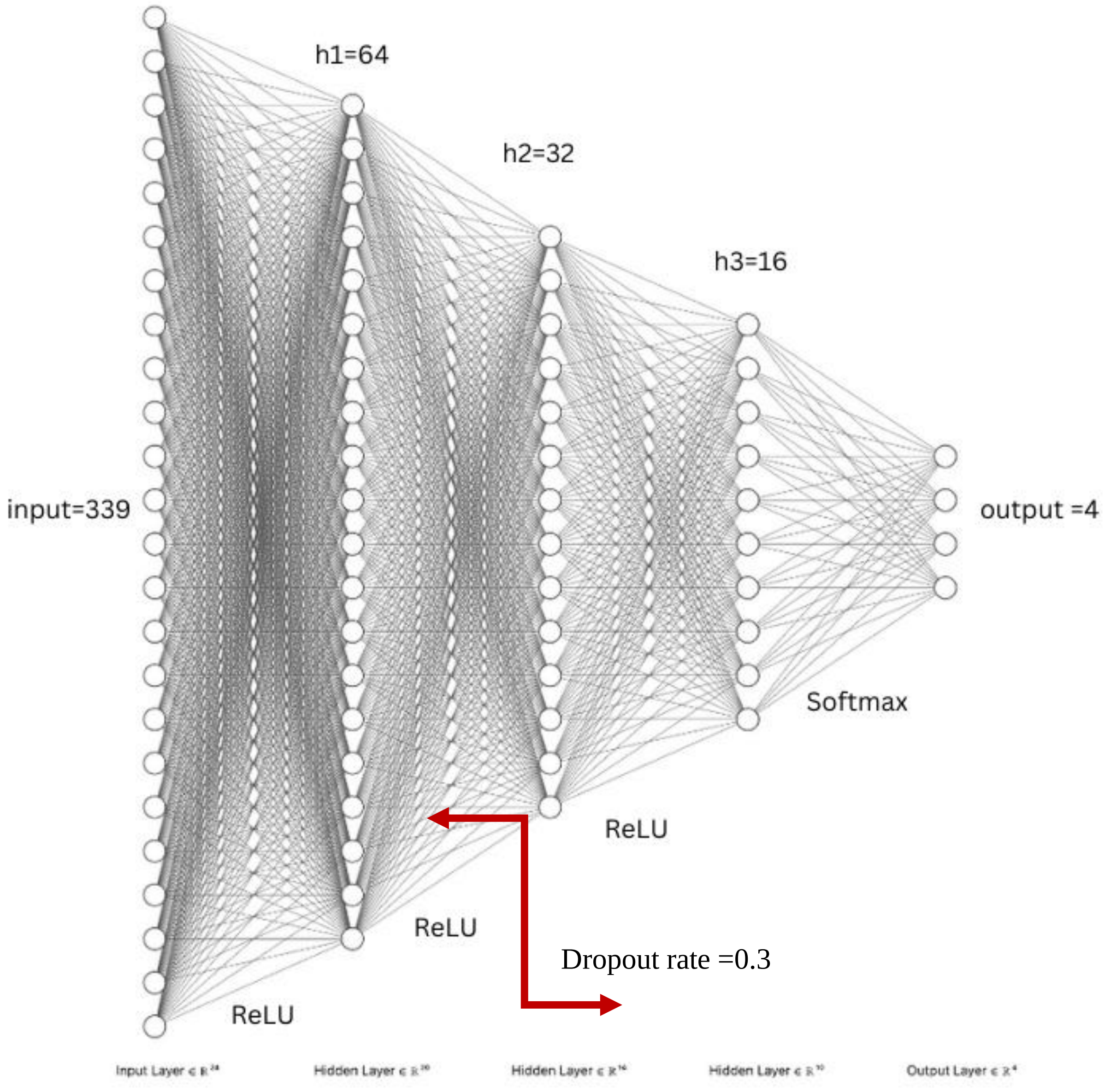


Figure 7.1.1: Visual representation of neural network model, created using website NN SVG

Values h1, h2, h3 are the number of neurons in each hidden layer. Notice we apply the dropout rate = 0.3 after the first layer to prevent overfitting.

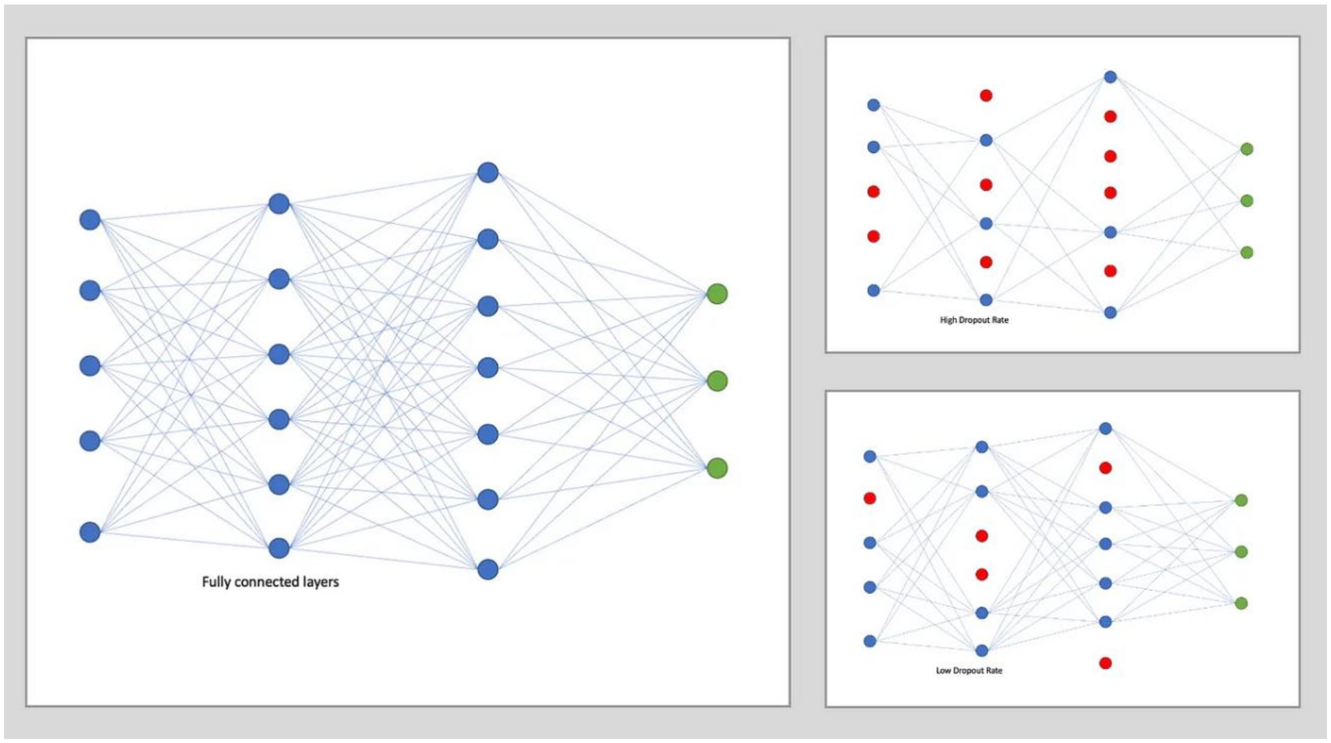


Figure 7.1.2: Drop out layer example [49]

Above is an example visual representation of dropout in neural network. With a dropout rate = 0.3, random 30% of hidden layer 1 (h1=64) is deactivated, leaving 70% of neuron activated – around 45 neurons. The second hidden layer receives input from only about 45 active neurons instead of all 64.

Optimization and Training Strategy: Train the model using Stochastic Gradient Descent, calculate errors using sparse categorical cross-entropy, and report accuracy after each epoch.

The model was compiled using Stochastic Gradient Descent (SGD) as the optimizer. SGD was chosen due to its ability to efficiently handle large datasets while maintaining generalization. Unlike adaptive optimizers such as Adam, which can lead to overfitting by adapting learning rates dynamically, SGD provides better convergence properties and encourages a more stable learning process [44]. Additionally, SGD is computationally less expensive, making it a suitable choice for training deep networks on moderate-sized datasets [44].

In this model, Stochastic Gradient Descent (SGD) demonstrated better performance compared to Adam. SGD consistently achieved an accuracy of around 0.44 across various training sets, while Adam's performance was less stable, fluctuating around 0.30 and exhibiting higher variance between epochs. As an iterative optimization algorithm that updates model weights using mini-batches, SGD provided a more stable and reliable learning process for the dataset used.

Given the relatively small size of the dataset, an epoch range between 10 and 50 is generally considered optimal [39]. After experimenting with various configurations, the model achieved its best performance with 15 epochs and a batch size of 64. This means the model updated its internal parameters after every 64 training samples and completed 15 full passes through the training set.

To prevent overfitting, dropout regularization was tested. Dropout values typically range from 0.2 to 0.5, where 0 implies no dropout and 1 disables all units [38]. A dropout rate of 0.3 produced the highest test accuracy, suggesting a good balance between retaining useful features and preventing over-reliance on any single neuron.

The neural network architecture consists of three hidden layers with 64, 32, and 16 neurons, respectively. These layer sizes were selected based on the reduced dimensionality of the input (after PCA) and the small sample size, aiming to balance model complexity with generalization.

For loss calculation, the model employed sparse categorical cross-entropy, a suitable choice for multi-class classification tasks when labels are integers rather than one-hot encoded vectors [18]. This loss function measures how well the predicted probability distribution aligns with the true class labels. The primary evaluation metric was accuracy, including Accuracy: percentage of correct classifications on the training set; Loss: training error; Validation Accuracy: correct classifications on the test set; Validation Loss: error on the test set

ii. Fuzzy Clustering Hyperparameter Tuning

In this project, we applied Fuzzy C-Means (FCM) clustering to identify natural groupings within the data. To optimize the model, we tuned two key hyperparameters: the number of clusters (c) and the fuzziness exponent (m). The fuzziness exponent controls how strongly data points belong to multiple clusters. We tested values of m = 1.5, 2.0, and 2.5, and evaluated performance using the Fuzzy Partition Coefficient (FPC), which measures the clarity of cluster boundaries. Across all values of m, the FPC remained stable at approximately 0.25, suggesting that the level of fuzziness had minimal impact on the model's performance. Therefore, we selected m = 2.0, the standard value used in most fuzzy clustering applications.

Next, we varied the number of clusters from 2 to 32 to determine the most appropriate value of c. This range was chosen based on SoundCloud's 64 genre and mood categories, with 32 clusters used as an estimate for the number of potential mood categories, acknowledging that the exact count is unknown. The Fuzzy Partition Coefficient (FPC) reached its highest value of 0.5 when c = 2, indicating the presence of two well-defined fuzzy clusters in the data. As the number of clusters increased, the FPC steadily declined, suggesting a reduction in clustering quality. Based on this analysis, we selected c = 2 as the optimal number of clusters.

The error threshold is 0.005, meaning the model stops running when the improvement becomes very small — specifically, if the change is less than 0.005. This helps save time without giving up too much accuracy. The maximum number of steps was set to 1000 to make sure the algorithm has enough time to find the best cluster positions, especially if the data is complicated or takes longer to settle.

iii. K-Means Hyperparameter Tuning

To choose the best number of clusters for K-Means, we used the Elbow Method. We first standardized the data using StandardScaler so that all features had the same scale. Then, we tested different values of clusters from 1 to 14 and recorded the inertia, which measures how tightly the data points are grouped within each cluster. The elbow plot showed a sharp drop in inertia up to about 10 clusters, then began to flatten out. The “elbow point” appeared at k = 12, where adding more clusters did not improve the result by much. Based on this, we selected 12 clusters as the most suitable choice. We then visualized the results using the first two scaled features, with cluster centers shown to highlight how the data was grouped.

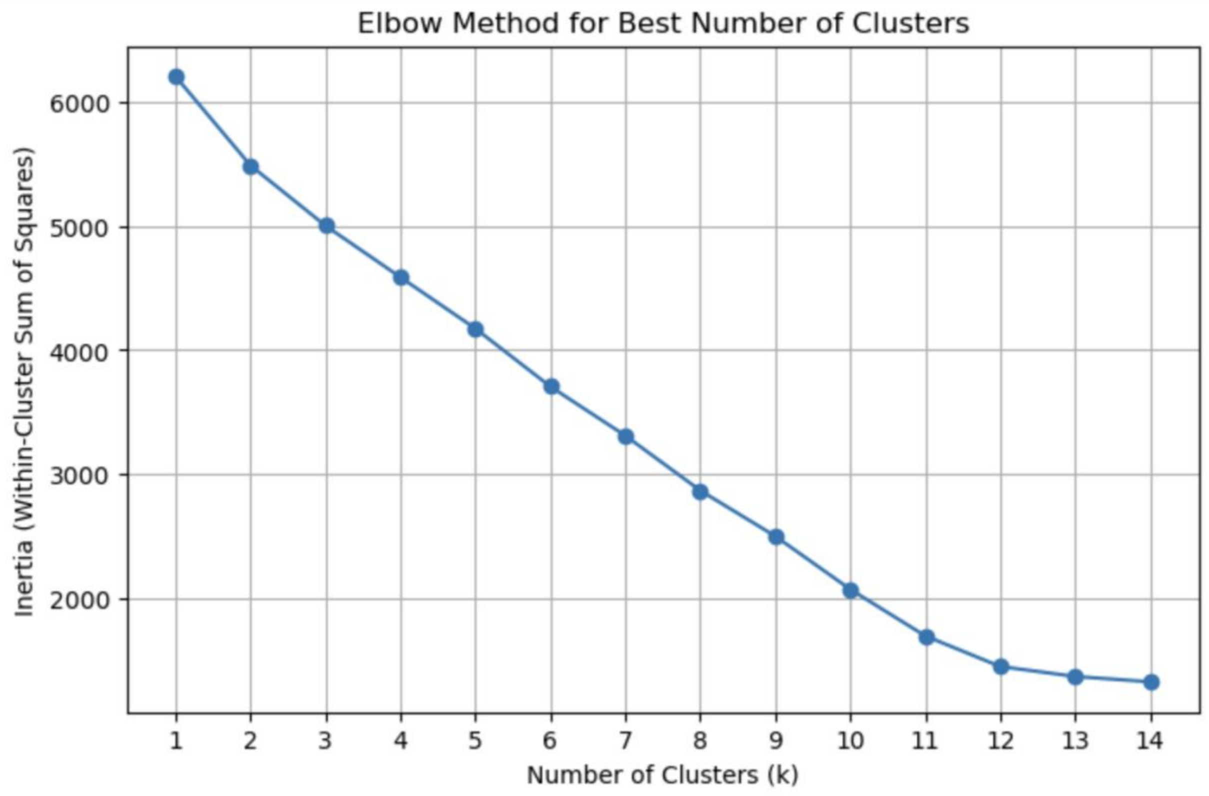


Figure 7.3.1: Elbow Method on K-Means clustering

VIII. Results

i. Clustering Results

a. Fuzzy Clustering including Mel-spectrogram vectors

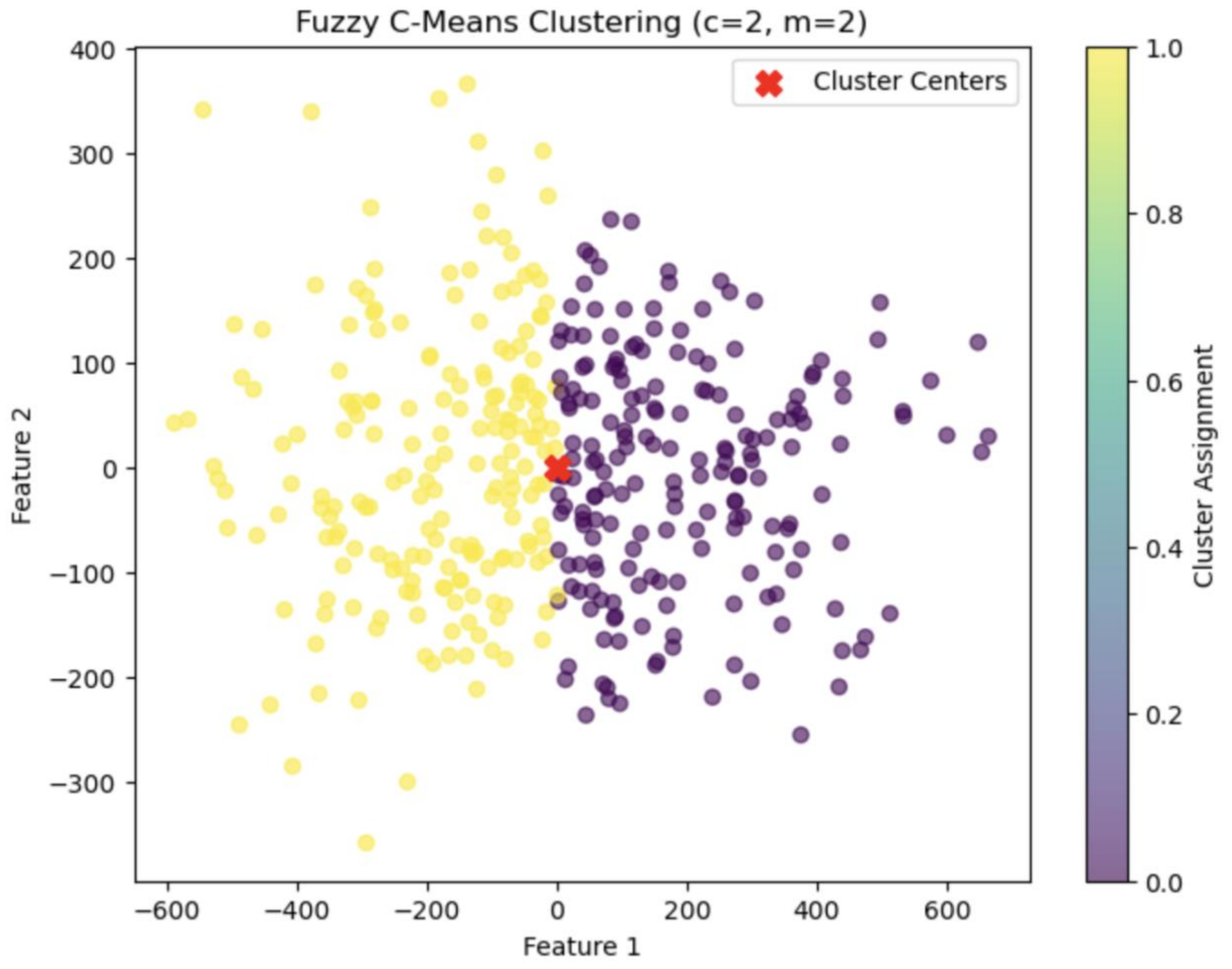


Figure 8.1.1: Fuzzy C-Means Clustering

The clustering result reveals that the data points are distributed across two clusters with very similar membership values. For example, the fuzzy membership matrix for the first five data points shows values very close to 0.5 for each cluster, such as [0.50000036, 0.49999964] for the first data point. This pattern is consistent across the sample, indicating that each point belongs to both clusters almost equally. Such uniform membership values suggest that the data points lie near the decision boundary between the clusters, or that the clusters themselves are highly overlapping in the reduced two-dimensional space.

The scatter plot supports this interpretation. It shows a strong concentration of points near the cluster centers, with minimal visible separation between the two groups. Although the data has been projected into two principal components for visualization, the lack of clear distinction between clusters may be a consequence of information loss during dimensionality reduction, or of insufficient feature variance in the original dataset. The silhouette score, which measures the clarity of clustering, is also low at 0.0668. This further supports the observation that the clusters are not well-separated and that the model is assigning nearly equal membership probabilities to many data points.

While Fuzzy C-Means is designed to handle overlapping clusters and provide soft assignments, the current results suggest limited cluster definition. This may be due to the PCA transformation reducing discriminative power, a suboptimal feature space, or natural homogeneity within the dataset. To address this, further feature engineering or the use of additional PCA components may help preserve more variance. Moreover, testing alternative distance metrics or fuzzy clustering algorithms could provide improved separation. Finally, while the selected number of clusters (c = 2) produced the highest Fuzzy Partition Coefficient (FPC), experimenting with other values of c or adjusting the fuzziness exponent m may lead to more meaningful structure in the data.

To further assess the model, we calculated the silhouette score using the hard cluster assignments derived from the fuzzy membership matrix. The silhouette score was approximately 0.0668, which is very low and indicates that the clusters are not well-separated in the reduced two-dimensional PCA space. This suggests that while fuzzy clustering captures some structure in the data, the overall separability between clusters is weak.

In conclusion, the best-performing FCM model used c = 2 clusters and a fuzziness exponent of m = 2.0. Although the model achieved the highest possible FPC value at this setting, the low silhouette score highlights the limitations of the dataset’s cluster structure. Nonetheless, the soft clustering results provided by FCM still offer valuable insight into the degrees of membership for each data point and may be useful in applications where overlapping group membership is important.

b. K-Means Clustering using Music Features

After selecting 12 clusters using the Elbow Method, we applied K-Means clustering to a dataset of musical features to group songs by mood. The model assigned each data point to one of the 12 clusters, with the largest cluster containing 74 songs and the smallest containing 13. We examined the average characteristics of each cluster, which included features such as BPM (beats per minute), key strength, spectral centroid, musical key, and whether the song was in a minor scale. These cluster centers suggest meaningful differences in musical qualities across groups. For example, Cluster 2 had a high average BPM and key strength, while Cluster 7 showed the lowest values. Some clusters were also associated with specific musical keys, such as Cluster 0 with key C and Cluster 1 with key D. The clustering structure was visualized in two dimensions using PCA, where each point represents a song, and the red X marks show the centroids of each group. The Silhouette Score for this clustering was 0.43, indicating moderate separation between clusters and suggesting that the model captured useful structure in the data, though some overlap among groups may still exist.

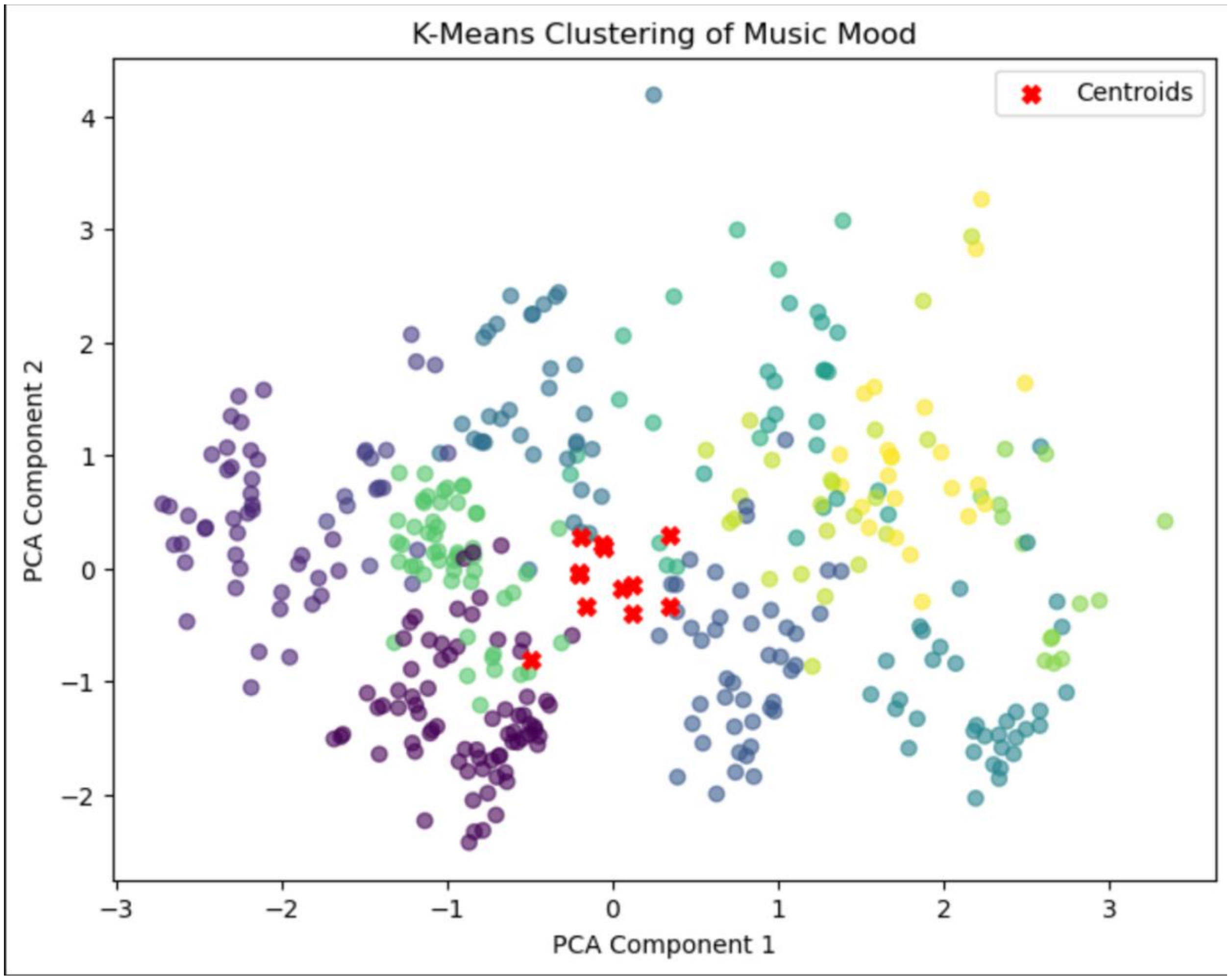


Figure 8.1.2 K-Means Clustering of Music

The scatter plot shows the results of the clustering in two dimensions, with each point representing a song and the red X markers showing the cluster centers. This visualization helps us see how the songs are grouped based on their characteristics.

Cluster	Cluster member count	BPM	Spectral Centroid	Main Key	Minor Scale %
0	74	113.87	2402.81	C	44.6
1	41	113.68	2590.2	D	68.3

2	17	118.33	2336.95	B	76.5
3	44	110.15	2489.65	F	29.5
4	35	124.11	2552.11	E	65.7
5	34	110.51	2426.93	G	41.2
6	19	116.66	2781.73	F#	73.7
7	13	102.85	2602.49	C#	53.8
8	52	124.02	2488.82	NA	73.1
9	14	110.26	2689.48	Eb	42.9
10	22	118.25	2924.83	Ab	27.3
11	23	111.14	2695.38	Bb	56.5

Figure 8.1.3: K-Means summaries

Cluster summaries show how each group of songs differs. For example, Cluster 0 has the most songs (74) and includes tracks with a moderate tempo (113.87 bpm), a balanced spectral centroid, and a strong presence of the C key. About 45% of the songs in this group are in a minor scale, suggesting a neutral or slightly emotional mood. Cluster 1 has songs mainly in the key of D, with a lower tempo and fewer minor key songs, possibly reflecting a calmer mood. Cluster 2 includes a variety of keys and has the highest percentage of minor scale songs (76.5%), which may indicate a more emotional or sad mood. Cluster 3 shows the highest tempo and spectral centroid and is dominated by the Ab key, likely representing high-energy or intense tracks.

The remaining clusters (4 to 11) also have their own musical traits, such as different keys, tempo ranges, and minor scale usage. This allows for grouping songs based on mood-related features.

The Silhouette Score for the model was 0.43, which suggests that the clusters are moderately well-defined, although there may still be some overlap. Overall, the clustering provides useful insights into how music can be grouped by mood, and it can help with tasks like playlist creation or mood-based music recommendations.

ii. Neural Network Results.

Epoch	Train Accuracy	Train Loss	Val Accuracy	Val Loss
1	0.2879	45.3368	0.2949	2.7058
2	0.347	2.8175	0.4359	1.6532
3	0.3597	1.6541	0.4359	1.5722
4	0.3812	1.657	0.4487	1.5306
5	0.3874	1.509	0.4487	1.4203
6	0.4257	1.3686	0.4487	1.4636
7	0.4275	1.4829	0.4359	1.3627
8	0.4106	1.3805	0.4359	1.365
9	0.4319	1.3387	0.4359	1.3746
10	0.4355	1.3201	0.4359	1.3699
11	0.4028	1.403	0.4359	1.393
12	0.4507	1.3152	0.4359	1.4203
13	0.4457	1.3393	0.4359	1.4202
14	0.4323	1.3305	0.4359	1.3853
15	0.4201	1.3412	0.4359	1.3969

Figure 8.2.1: Neural Network Evaluation Metrics

During training, the model showed a significant improvement in both training and validation performance after the first epoch. Initially, the training loss was very high at 45.34, and accuracy was low at 28.79%, indicating that the model was untrained and unconfident in its predictions. However, by Epoch 2, the training loss dropped to 2.82, and validation accuracy rose to 43.59%, marking a considerable learning gain.

Between Epochs 3 to 5, the model continued to improve gradually. Training accuracy increased to 38.74%, while the validation accuracy stabilized around 44.87%, suggesting the model was starting to generalize. Validation loss also decreased to 1.42 by Epoch 5.

From Epochs 6 through 15, training accuracy continued to rise modestly, peaking at 45.07% in Epoch 12, while training loss gradually decreased. However, validation accuracy plateaued at 43.59% from Epoch 6 onward, and validation loss hovered between 1.36 and 1.42, suggesting the model had reached its generalization capacity for this configuration.

The final test accuracy remained at 43.59%, showing that although the model improved on training data, its ability to generalize to unseen data was limited, possibly due to model capacity, data size, or feature complexity.

Based on the results shown across the 15 training epochs, the model demonstrates moderate learning progress but struggles with generalization and optimization. In Epoch 1, it begins with a very high training loss of 45.34 and a low accuracy of 28.79%, which is expected from an untrained neural network. By Epoch 2, however, the model adapts quickly, dropping training loss to 2.82 and improving accuracy to 34.70%. This sharp early decline suggests that the model is effectively learning basic patterns in the data during the initial stages.

As training continues, the model's performance improves more gradually. Training accuracy peaks at 45.07% by Epoch 12, with loss decreasing to around 1.32. This indicates that the model is capturing structure in the training data. However, validation accuracy remains flat at 43.59% from Epoch 2 onward, and validation loss fluctuates between 1.3 and 1.6. These signs point toward overfitting: while the model is optimizing on the training set, it is failing to improve on unseen data, limiting its ability to generalize.

The narrow margin between training and validation accuracy—roughly 1.5%—suggests mild overfitting. This plateau hints that the model may have reached the learning limits imposed by the current architecture, input features, or dataset size. Moreover, the lack of a consistent downward trend in validation loss indicates that the model's predictions on the validation set are not improving as training progresses.

IX. Discussion

i. Model comparison

This project compared the performance of unsupervised clustering methods (Fuzzy C-Means and K-Means) with a supervised classification model (Neural Network) to analyze and group songs based on musical mood.

The Fuzzy C-Means (FCM) model was designed to allow songs to belong to multiple clusters with varying degrees of membership. While this method achieved a high Fuzzy Partition Coefficient (FPC = 0.5) at c = 2, the resulting clusters had nearly equal membership values for all data points. This suggests that the clusters were not clearly separated, which was further supported by a low silhouette score (0.0668). Therefore, although FCM offered soft clustering capabilities, it was less effective in revealing meaningful structure in the data.

In contrast, the K-Means model produced better clustering results. Using the Elbow Method, 12 clusters were selected. The silhouette score for this model was 0.43, indicating moderate separation between clusters. Cluster analysis showed clear differences in features like BPM, key, and scale, which aligned well with different musical moods. The use of PCA helped visualize these clusters, making K-Means more interpretable and effective than FCM in this context.

The Neural Network model was used for multi-class mood classification. It was trained on PCA-reduced data and reached a validation accuracy of 43.59%. While the model showed early improvement, it plateaued quickly and struggled to improve performance beyond a certain point. Compared to K-Means, the neural network required more computation and showed limited gains, likely due to the small dataset size and feature limitations.

Overall, K-Means provided the most useful and interpretable results, followed by the Neural Network. Fuzzy C-Means was less effective in this application, due to the overlapping nature of the resulting clusters and weak separability in the data.

ii. Discussion

Each model used in this study served a unique purpose and offered different insights into the structure and classification of music mood data. The Fuzzy C-Means (FCM) clustering model was selected for its ability to assign soft membership values, allowing songs to partially belong to multiple clusters. This is especially useful when dealing with abstract concepts like mood. However, FCM showed limited effectiveness in this application. One possible reason is that mood in music often lacks sharp boundaries, and when combined with dimensionality reduction through PCA, the model struggled to separate groups clearly. Although soft clustering can be powerful in theory, in this case, the result highlighted the difficulty of applying it to a relatively small and highly compressed dataset.

The K-Means model offered a more practical approach to clustering. It provided clear assignments, making it easier to interpret and analyze song groupings. One strength of K-Means is its simplicity and speed, which made it suitable for exploring patterns in the reduced feature space. However, K-Means assumes that clusters are spherical and non-overlapping, which may not fully capture the nuance of musical mood. It also requires specifying the number of clusters ahead of time, which depends heavily on subjective decisions or estimation methods like the Elbow Method. While it worked well in this context, it may not perform as effectively on more complex or less structured datasets.

The neural network model was built for supervised mood classification using reduced features from PCA. This architecture allowed for non-linear relationships and deeper learning from the data. While the model demonstrated some learning progress during training, its performance was limited by the size and structure of the dataset. With fewer than 400 examples, the network likely did not have enough information to generalize well. Additionally, the reduction in input dimensions may have led to a loss of important information. Overfitting was also a concern, despite the use of dropout regularization. These limitations suggest that while the model was well-designed, its full potential could not be realized under the current conditions.

One of the key limitations of this study is the size and diversity of the dataset. The dataset consists of a relatively small number of songs (50-60 per genre) spanning only eight genres, which may not fully capture the wide spectrum of musical compositions across different cultures, styles, and time periods. As a result, the model may be biased toward the specific genres included, such as Blues, Electronic, and Hip-Hop, and may not generalize well to Classical, Folk, or Experimental music, where different musical structures and emotional cues are prevalent. Additionally, some moods may be underrepresented, leading to imbalanced training and difficulties in accurately classifying certain emotional tones.

Another major limitation is the subjectivity of mood perception in music. Emotional responses to music vary widely across individuals due to personal experiences, cultural influences, and listening contexts. A song perceived as "relaxing" by one listener may evoke "melancholy" in another. The model relies on fixed relationships between musical features (e.g., tempo, pitch, spectral centroid) and mood, which may not always align with human interpretations. Moreover, external factors such as the listening environment, lyrics, and personal associations with a song are not considered, which could lead to classification errors in real-world applications.

The feature representation used in this study is also simplified and may not fully capture the complexity of musical emotions. The analysis of spectrograms and Mel-spectrograms was limited to fixed time frames, potentially missing key emotional shifts that occur later in longer or evolving compositions. Additionally, other important aspects of music, such as lyrics, chord progressions, harmonic transitions, and instrumental timbre variations, were not included in the feature extraction process. This omission could reduce the model’s ability to recognize subtle mood variations and limit its effectiveness in distinguishing between similar emotional tones.

Finally, the model's performance is constrained by overfitting and generalization. The Neural Network results showed a significant gap between training accuracy (54.65%) and validation accuracy (43.59%), with validation loss fluctuating instead of consistently decreasing. This suggests that the model memorizes training data rather than learning generalizable patterns, likely due to the high dimensionality of extracted features and the relatively small dataset size. Regularization techniques such as dropout and L2 weight decay were not extensively tested, which could have helped mitigate overfitting [45]. Additionally, the clustering analysis revealed that the centroids were relatively close, indicating that some mood categories overlapped due to the continuous nature of emotions in music, making strict classification inherently difficult.

iii. Future Work

To address these limitations, future research should focus on expanding the dataset to include a broader range of music genres and a larger number of songs, ensuring better representation of diverse emotional tones. A more extensive dataset would allow the model to learn richer musical patterns and improve generalization, reducing the risk of genre-specific biases. Additionally, incorporating multilingual datasets that account for different cultural perspectives on music and mood could enhance the robustness of the classification system.

Further improvements can be made by integrating additional audio features, such as lyrics analysis, chord progressions, rhythmic patterns, and harmonic transitions, to provide a more comprehensive understanding of a song's mood. Since lyrics play a crucial role in emotional expression, their inclusion through Natural Language Processing (NLP) techniques could significantly improve classification accuracy. Additionally, dynamic spectrogram analysis that captures changes in mood over time rather than a fixed segment could lead to more accurate temporal mood tracking, particularly for genres with complex structures like progressive rock, classical, and jazz.

On the modeling side, implementing advanced deep learning architectures such as Convolutional Neural Networks (CNNs) for spectrogram-based classification or Recurrent Neural Networks (RNNs) networks for sequential dependencies in audio features could enhance the model's ability to recognize mood variations [28]. Hybrid models that combine CNNs for spectrogram feature extraction for capturing temporal changes could further improve classification accuracy. Additionally, fine-tuning hyperparameters, applying dropout regularization, and experimenting with different activation functions could help reduce overfitting and enhance model performance.

For clustering, alternative methods such as Gaussian Mixture Models (GMMs) could be explored to better capture the continuous and overlapping nature of mood categories [29]. Another potential improvement is to develop a hybrid recommendation system that combines supervised mood classification with unsupervised clustering, enabling classification based on a user’s listening history and emotional state.

Lastly, future research could explore context-aware music recommendation, where mood classification is influenced by user behavior, activity, time of day, or location. By integrating reinforcement learning and adaptive AI models, the algorithm could learn and adjust recommendations based on real-time user feedback, creating a truly dynamic and personalized music experience. These enhancements would significantly improve the accuracy, flexibility, and usability of the Songs Mood Recognition System, making it a powerful tool for music streaming services, emotional AI applications, and therapeutic music interventions.

References

[1] "Pitch and Frequency," *The Physics Classroom*. [Online]. Available: https://www.physicsclassroom.com/class/sound/lesson-2/pitch-and-frequency

[2] J. M. Kua, T. Thiruvaran, M. H. Nosratighods, E. Ambikairajah, and J. R. Epps, "Investigation of Spectral Centroid Magnitude and Frequency for Speaker Recognition," in *Proceedings of Odyssey 2010: The Speaker and Language Recognition Workshop*, Brno, Czech Republic, Jun. 2010. [Online]. Available: https://www.researchgate.net/publication/264986258

[3] T. Giannakopoulos and A. Pikrakis, "Audio Features," in *Introduction to Audio Analysis: A MATLAB® Approach*, 1st ed. Amsterdam, Netherlands: Academic Press, 2014, ch. 4. [Online]. Available: https://www.sciencedirect.com/science/article/abs/pii/B9780080993881000042

[4] I. Pasha, "Centroiding Tutorial," *Python for Astronomers*, 2025. [Online]. Available: https://prappleizer.github.io/Tutorials/Centroiding/centroiding_tutorial.html

[5] D. Mitrović, M. Zeppelzauer, and C. Breiteneder, "Features for Content-Based Audio Retrieval," *Advances in Computers*, vol. 78, pp. 71–150, 2010, doi: 10.1016/S0065-2458(10)78003-7

[6] X. Hu and J. S. Downie, "When Lyrics Outperform Audio for Music Mood Classification: A Feature Analysis," in *Proc. 11th Int. Soc. Music Inf. Retrieval Conf. (ISMIR 2010)*, Utrecht, Netherlands, Aug. 2010, pp. 619–624. [Online]. Available: https://ismir2010.ismir.net/proceedings/ismir2010-106.pdf

[7] "ML | Fuzzy Clustering," *GeeksforGeeks*, Sep. 10, 2019. [Online]. Available: https://www.geeksforgeeks.org/ml-fuzzy-clustering/

[8] V. R. Madgazin, "The Information Theory of Emotions of Musical Chords," *arXiv preprint arXiv:0909.3976*, Sep. 2009. [Online]. Available: https://arxiv.org/abs/0909.3976

[9] E. G. Schellenberg and S. E. von Scheve, "Emotional Connotations of Major and Minor Tonality: One or More Origins?," *Musicae Scientiae*, vol. 18, no. 3, pp. 307–313, Sep. 2014. [Online]. Available: https://journals.sagepub.com/doi/abs/10.1177/1029864914542842

[10] M. Plewa and B. Kostek, "A Study on Correlation Between Tempo and Mood of Music," *AES Convention Paper 8725*, Oct. 2012. [Online]. Available: https://www.researchgate.net/publication/292853487

[11] A. M. Ikotun, A. E. Ezugwu, L. Abualigah, B. Abuhaija, and J. Heming, "K-means Clustering Algorithms: A Comprehensive Review, Variants Analysis, and Advances in the Era of Big Data," *Information Sciences*, vol. 622, pp. 178–210, 2023. doi: 10.1016/j.ins.2022.11.139

[12] A. Géron, "Introduction to Artificial Neural Networks," in *Neural Networks and Deep Learning*, 1st ed. Sebastopol, CA, USA: O'Reilly Media, 2018, ch. 1. [Online]. Available: https://learning.oreilly.com/library/view/neural-networks-and/9781492037354/ch01.html

[13] "Fuzzy Clustering in R," *GeeksforGeeks*. [Online]. Available: https://www.geeksforgeeks.org/fuzzy-clustering-in-r/

[14] "Softmax Activation Function: Everything You Need to Know," *Pinecone*. [Online]. Available: https://www.pinecone.io/learn/softmax-activation/

[15] J. Brownlee, "A Gentle Introduction to the Rectified Linear Activation Function for Deep Learning Neural Networks," *Machine Learning Mastery*, Jan. 8, 2019. [Online]. Available: https://www.machinelearningmastery.com/rectified-linear-activation-function-for-deep-learning-neural-networks/

[16] A. Jain, "ML | Stochastic Gradient Descent (SGD)," *GeeksforGeeks*, Aug. 21, 2019. [Online]. Available: https://www.geeksforgeeks.org/ml-stochastic-gradient-descent-sgd/

[17] I. T. Jolliffe, *Principal Component Analysis*, 2nd ed. New York, NY, USA: Springer, 2002.

[18] "Sparse Categorical Crossentropy vs. Categorical Crossentropy," *GeeksforGeeks*, Jan. 9, 2025. [Online]. Available: https://www.geeksforgeeks.org/sparse-categorical-crossentropy-vs-categorical-crossentropy/

[19] B. Murauer and G. Specht, "Detecting Music Genre Using Extreme Gradient Boosting," in *Companion Proc. The Web Conf. 2018*, Lyon, France, Apr. 2018, pp. 1893–1897. doi: 10.1145/3184558.3191822

[20] B. G. Patra, D. Das, and S. Bandyopadhyay, "Unsupervised Approach to Hindi Music Mood Classification," in *Mining Intelligence and Knowledge Exploration*, R. Prasath and T. Kathirvalavakumar, Eds., vol. 8284, *Lecture Notes in Computer Science*. Cham, Switzerland: Springer, 2013, pp. 57–67. doi: 10.1007/978-3-319-03844-5_7

[21] M. B. Er and I. B. Aydilek, "Music Emotion Recognition by Using Chroma Spectrogram and Deep Visual Features," *International Journal of Computational Intelligence Systems*, vol. 12, no. 2, pp. 1622–1634, Dec. 2019, doi: 10.2991/ijcis.d.191216.001.

[22] A. Sen, "Development as Freedom," in *The Elgar Companion to Development Studies*, P. B. Clark, Ed. Cheltenham, UK: Edward Elgar, 2014, pp. 49–54. doi: 10.1016/B978-0-08-099388-1.00004-2.

[23] J. Lian, "An Artificial Intelligence-Based Classifier for Musical Emotion Expression in Media Education," *PeerJ Computer Science*, vol. 9, p. e1472, Jul. 2023, doi: 10.7717/peerj-cs.1472.

[24] B. M. Nema and A. A. Abdul-Kareem, "Preprocessing Signal for Speech Emotion Recognition," *Al-Mustansiriyah Journal of Science*, vol. 28, no. 3, pp. 157–165, Jul. 2018, doi: 10.23851/mjs.v28i3.48.

[25] Analog Devices, "Sampling Rate," *Analog Devices Glossary*. [Online]. Available: https://www.analog.com/en/resources/glossary/sampling-rate.html

[26] D. Goyal and B. S. Pabla, "Condition Based Maintenance of Machine Tools—A Review," *CIRP Journal of Manufacturing Science and Technology*, vol. 10, pp. 24–35, Aug. 2015, doi: 10.1016/j.cirpj.2015.05.004.

[27] D. Alpay, A. De Martino, K. Diki, and D. C. Struppa, "Short-Time Fourier Transform and Superoscillations," *Integral Transforms and Special Functions*, vol. 32, no. 9, pp. 734–753, 2021, doi: 10.1080/10652469.2021.1914427.

[28] C. F. G. dos Santos and J. P. Papa, "Avoiding Overfitting: A Survey on Regularization Methods for Convolutional Neural Networks," *arXiv preprint arXiv:2201.03299*, Jan. 2022. [Online]. Available: https://arxiv.org/abs/2201.03299

[29] R. Thiruvengatanadhan, "Music Genre Classification Using GMM," *International Research Journal of Engineering and Technology (IRJET)*, vol. 5, no. 10, pp. 1061–1063, Oct. 2018. [Online]. Available: https://www.irjet.net/archives/V5/i10/IRJET-V5I10198.pdf

[30] P. Knees and M. Schedl, *Music Similarity and Retrieval*, Cham, Switzerland: Springer, 2016. doi: 10.1007/978-3-662-49722-7.

[31] S. Dredge, "SoundCloud Gets Mood and Genre Filters for Listener Libraries," *Music Ally*, Sep. 25, 2024. [Online]. Available: https://musically.com/2024/09/25/soundcloud-gets-mood-and-genre-filters-for-listener-libraries/

[32] "Musical Key Characteristics," *Western Michigan University*, 2024. [Online]. Available: https://legacy.wmich.edu/mus-theo/courses/keys.html

[33] J. Davis, "The Evolution of Music Production Technology: From Analog to Digital and Beyond," *Illustrate Magazine*, Jan. 2023. [Online]. Available: https://illustratemagazine.com/the-evolution-of-music-production-technology-from-analog-to-digital-and-beyond/

[34] "Prehistoric Music," *Wikipedia*, Mar. 27, 2025. [Online]. Available: https://en.wikipedia.org/wiki/Prehistoric_music#cite_note-1

[35] A. Pang, "Algorithmic Symphonies: How Spotify Strikes the Right Chord," *illumin — USC Viterbi School of Engineering*, vol. 23, no. 1, Feb. 2022. [Online]. Available: https://illumin.usc.edu/algorithmic-symphonies-how-spotify-strikes-the-right-chord/

[36] S. Maesawa *et al.*, "Changes in Musical Frequency Influence Physiological and Psychological States," *Frontiers in Psychology*, vol. 13, Jan. 2022. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8854572/

[37] Mixing Monster, "EQ Frequency Ranges – The Ultimate Guide," *Mixing Monster*, Apr. 2, 2023. [Online]. Available: https://mixingmonster.com/eq-frequency-ranges/

[38] P. Kashyap, "Understanding Dropout in Deep Learning: A Guide to Reducing Overfitting," *Medium*, Jul. 14, 2023. [Online]. Available: https://medium.com/@piyushkashyap045/understanding-dropout-in-deep-learning-a-guide-to-reducing-overfitting-26cbb68d5575

[39] "How to choose Batch Size and Number of Epochs when fitting a model?," *GeeksforGeeks*, Aug. 4, 2021. [Online]. Available: https://www.geeksforgeeks.org/how-to-choose-batch-size-and-number-of-epochs-when-fitting-a-model/

[40] N. Kulkarni and Vinayak Bairagi, "Use of Complexity Features for Diagnosis of Alzheimer Disease," *Elsevier eBooks*, pp. 47–59, Jan. 2018, doi: https://doi.org/10.1016/b978-0-12-815392-5.00004-6.

[41] "Covariance Matrix," *GeeksforGeeks*, Sep. 19, 2023. [Online]. Available: https://www.geeksforgeeks.org/covariance-matrix/

[42] “Eigen Values,” *GeeksforGeeks*, Jan. 27, 2023. [Online]. Available: https://www.geeksforgeeks.org/eigen-values/

[43] “ReLU Activation Function in Deep Learning,” *GeeksforGeeks*, Aug. 17, 2023. [Online]. Available: https://www.geeksforgeeks.org/relu-activation-function-in-deep-learning/

[44] F. Dijkinga, “The SGD Optimizer,” *Medium*, Jan. 31, 2022. [Online]. Available: https://medium.com/@fernando.dijkinga/the-sgd-optimizer-4764cc0f0493

[45] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, Cambridge, MA, USA: MIT Press, 2016, ch. 7. [Online]. Available: https://www.deeplearningbook.org/contents/regularization.html

[46] T. A. Hope, “Fourier Transform (FT),” *MRI Questions*, 2023. [Online]. Available: https://mriquestions.com/fourier-transform-ft.html

[47] "Short-time Fourier transform," *Wikipedia*, Mar. 26, 2024. [Online]. Available: https://en.wikipedia.org/wiki/Short-time_Fourier_transform

[48] Д. Ильин, *Signal sampling representation. The continuous signal S(t) is represented with a green colored line while the discrete samples are indicated by the blue vertical lines*, 2023. [Online]. Available: Wikipedia.

[49] C. Daniela, *Dropout Layer Explained in the Context of CNN*, Medium, Sep. 9, 2020. [Online]. Available: https://cdanielaam.medium.com/dropout-layer-explained-in-the-context-of-cnn-7401114c2c

[50] T. Raitio, *Spectrogram and the STFT*. Speech Processing Book, Aalto University. [Online]. Available: https://speechprocessingbook.aalto.fi/Representations/Spectrogram_and_the_STFT.html

Appendix

R code for K-means example:

```
iris_data <- iris[, 1:4]
set.seed(123)
kmeans_result <- kmeans(iris_data, centers = 3)

# Add column cluster
iris$Cluster <- as.factor(kmeans_result$cluster)

# make a dataframe name centroids that stores k_means centers
centroids <- as.data.frame(kmeans_result$centers)
centroids$Cluster <- as.factor(1:3)

# Plot and add shapes as misclassification
ggplot(iris, aes(x = Sepal.Length, y = Sepal.Width, color = Cluster, shape = Species)) +
  geom_point(size = 3, alpha = 0.7) +
  geom_point(data = centroids, aes(x = Sepal.Length, y = Sepal.Width),
             shape = 4, color = "black", size = 5, stroke = 2) +
  labs(title = "K-Means Clustering of Iris Data",
       subtitle = "Color = Cluster, Shape = Actual Species",
       x = "Sepal Length", y = "Sepal Width") +
  theme_minimal()
```

Python code for Audio Preprocessing

```
y,sr = librosa.load('FMAaudio/Blues/Cullah - Open Your Eyes! (To The World).mp3', sr=44100,
mono = 1)
#standardize sampling rate, and ensure monophonic by mono=1
```

Noise Reduction first test

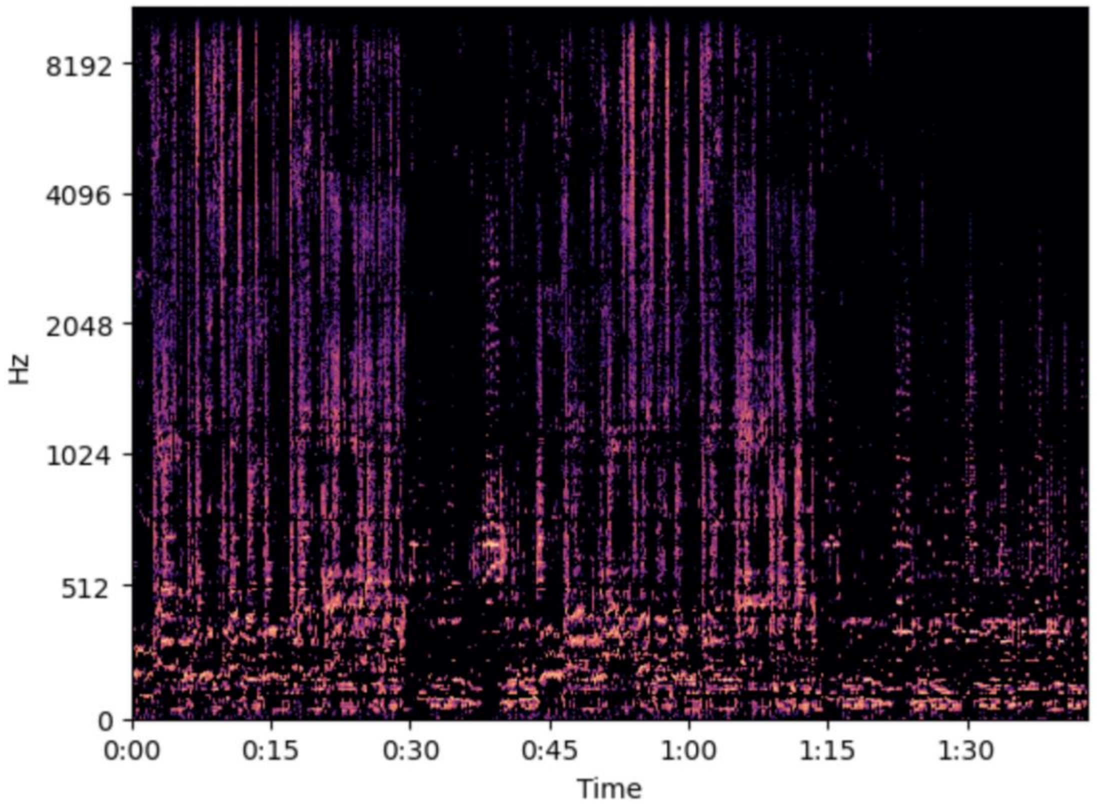
```
!pip install noisereduce
import noisereduce as nr
reduced_noise = nr.reduce_noise(y=y, sr=sr)
D = librosa.amplitude_to_db(np.abs(librosa.stft(reduced_noise)), ref=np.max)
librosa.display.specshow(D,x_axis='time', y_axis='mel')
#By the function reduce_noise from noisereduce package, which applied spectral gatting.
Assumption: Noise is quieter than the actual signal across most frequency bands.
```

Noise Reduction second test

```
audio_signal, sr = librosa.load(' FMAaudio/Denoise/Cullah - Open Your Eyes! (To The
World).mp3', mono = 1)
# Short-time Fourier Transform (STFT)
stft = librosa.stft(audio_signal)
magnitude, phase = librosa.magphase(stft)
# Estimate noise profile (e.g., from a silent section)
noise_profile = np.mean(magnitude[:,50 :140], axis=1, keepdims=True)
```

# Subtract noise from magnitude
magnitude_denoised = np.maximum(magnitude - noise_profile, 0)

# Reconstruct the denoised audio to test for denoise quality
stft_denoised = magnitude_denoised * phase
denoised_signal = librosa.istft(stft_denoised)
noise_subtract = librosa.amplitude_to_db(np.abs(stft_denoised), ref=np.max)
librosa.display.specshow(noise_subtract,x_axis='time', y_axis='mel')
noise_subtract = librosa.amplitude_to_db(np.abs(stft_denoised), ref=np.max)
librosa.display.specshow(noise_subtract,x_axis='time', y_axis='mel')



#Convert from spectrogram back to audio file
import IPython.display as ipd
ipd.Audio(reduced_noise, rate=sr)
ipd.Audio(denoised_signal, rate = sr)
#After testing between spectral gating and noise subtraction, noise subtraction seems to work better, as it reduces perceived noise without over-reducing like spectral gating. We will apply this method with all other songs with the window between 1 to 2 mins.

Neural Network model build

```
model = keras.Sequential([
    layers.Input(shape=(X_train.shape[1],)),  # Input layer (feature size from PCA)
    layers.Dense(64, activation="relu", kernel_initializer=keras.initializers.HeNormal()),
# First hidden layer
```

```
    layers.Dropout(0.3),  # Dropout layer to prevent overfitting
    layers.Dense(32, activation="relu", kernel_initializer=keras.initializers.HeNormal()),
# Second hidden layer
    layers.Dense(16, activation="relu", kernel_initializer=keras.initializers.HeNormal()),
# Third hidden layer
    layers.Dense(len(np.unique(Y)), activation="softmax")
# Output layer (softmax for multi-class classification)
])
```

Find best epoch (based on highest validation accuracy or lowest validation loss)

```
# Get validation accuracy and loss from history
val_acc = history.history['val_accuracy']
val_loss = history.history['val_loss']
best_epoch_acc = np.argmax(val_acc) + 1
best_epoch_loss = np.argmin(val_loss) + 1
print(f"Best Epoch based on Validation Accuracy: {best_epoch_acc} (val_accuracy =
{val_acc[best_epoch_acc - 1]:.4f})")
print(f"Best Epoch based on Validation Loss: {best_epoch_loss} (val_loss =
{val_loss[best_epoch_loss - 1]:.4f})")
```

K-Means

```
# Normalize the data (important for K-Means)
scaler = StandardScaler()
X_scaled = scaler.fit_transform(Y_cluster)

# Choose the number of clusters
n_clusters = 12

# K-Means clustering
kmeans = KMeans(n_clusters=n_clusters, random_state=42, n_init=10)
Y_cluster.loc[:, "Cluster"] = kmeans.fit_predict(X_scaled)
# Cluster counts
print(Y_cluster["Cluster"].value_counts())

from sklearn.decomposition import PCA

# Reduce data to 2D using PCA for visualization
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X_scaled)

# Extract cluster labels
clusters = Y_cluster["Cluster"]
```

```
# Create scatter plot of clusters
plt.figure(figsize=(8, 6))
plt.scatter(X_pca[:, 0], X_pca[:, 1], c=clusters, cmap="viridis", alpha=0.6)
plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1],
            marker="X", s=50, c="red", label="Centroids")
plt.title("K-Means Clustering of Music Mood")
plt.xlabel("PCA Component 1")
plt.ylabel("PCA Component 2")
plt.legend()
plt.show()

cluster_summary = Y_cluster.groupby("Cluster").mean()
print(cluster_summary)

#Evaluation Metrix
from sklearn.metrics import silhouette_score
score = silhouette_score(X_scaled, Y_cluster["Cluster"])
print("Silhouette Score:", score)
```

#Elbow curve to choose number of clusters

```
value = []

number_of_clusters = range(1, 15)

for num_clusters in number_of_clusters:
    kmeans_model = KMeans(n_clusters=num_clusters, random_state=42, n_init=10)
    kmeans_model.fit(X_Scaled)
    value.append(kmeans_model.inertia_)  # Sum of squared distances to cluster centers

# Plot the elbow curve
plt.figure(figsize=(8, 5))
plt.plot(number_of_clusters, value, marker='o')
plt.title('Elbow Method for Determining Optimal Number of Clusters')
plt.xlabel('Number of Clusters (k)')
plt.ylabel('Inertia (Within-Cluster Sum of Squares)')
plt.xticks(number_of_clusters)
plt.grid(True)
plt.show()
```

Fuzzy model (using PCA components for analyzing both audio and spectral features)

```
X_fcm = X_pca_final.T  # Transpose for skfuzzy input format
```

```
# hyperparameters
n_clusters = 2
m = 2.0

# Run Fuzzy
cntr, u, _, _, _, _, fpc = fuzz.cluster.cmeans(
    X_fcm, c=n_clusters, m=m, error=0.005, maxiter=1000, init=None
)

# soft memberships to hard assignments
hard_clusters = np.argmax(u, axis=0)

# Plot
fig, ax = plt.subplots(figsize=(8, 6))
scatter = ax.scatter(X_pca_final[:, 0], X_pca_final[:, 1],
                     c=hard_clusters, cmap='viridis', alpha=0.6)
ax.scatter(cntr[:, 0], cntr[:, 1], c='red', marker='X', s=100, label='Cluster Centers')

ax.set_title('Fuzzy C-Means Clustering (c=2, m=2)')
ax.set_xlabel('Feature 1')
ax.set_ylabel('Feature 2')
ax.legend()
plt.colorbar(scatter, label='Cluster Assignment')
plt.show()

# Silhouette Score
sil_score = silhouette_score(X_pca_final, hard_clusters)
print(f"Silhouette Score (approximate): {sil_score:.4f}")

# Tuning m
print("\nFPC for different m:")
for m_test in [1.5, 2.0, 2.5]:
    _, _, _, _, _, _, fpc = fuzz.cluster.cmeans(
        X_fcm, c=n_clusters, m=m_test, error=0.005, maxiter=1000
    )
    print(f"FPC with m={m_test}: {fpc:.4f}")

# Tuning numbers of clusters
print("\nFPC for different clusters:")
for c in range(2, 32):
    _, _, _, _, _, _, fpc = fuzz.cluster.cmeans(
        X_fcm, c=c, m=2.0, error=0.005, maxiter=1000
    )
    print(f"Clusters: {c}, FPC: {fpc:.4f}")
```