Music Recommendation System Based on Emotion

S V Sowmya Nithishwar – CB.EN.U4ECE21250
Sailesh Kumar T V– CB.EN.U4ECE21253
Srinidhih P V– CB.EN.U4ECE21258
Department of Electronics and Communication Engineering,
Amrita School of Engineering, Coimbatore,
Amrita Vishwa Vidyapeetham, India

***Abstract*— Choosing songs from a vast online music collection can be overwhelming, leading to confusion. While the music genre contributes to social identity, the emotional aspect and its impact on listeners are often overlooked when it comes to music preferences. Understanding the emotional expression of a song and its impression on the listener is crucial in helping individuals make informed decisions about the music they want to listen to, ensuring a more personalized and meaningful music experience. This paper presents a novel approach to recognizing and recommending music based on emotional characteristics. Using a dataset of music samples, a clustering algorithm is applied to categorize songs into emotion-based clusters. The system then predicts the emotion of a given song and recommends similar songs from the corresponding cluster. The proposed system demonstrates promising results in accurately recognizing and recommending music based on emotional attributes, providing an enhanced user experience, and facilitating personalized music recommendations.**

Keywords—Music, Emotion, Kmeans, Recommendation, PCA

I. INTRODUCTION

In the realm of music, listeners frequently encounter challenges when manually creating and organizing playlists, particularly when faced with a vast collection of hundreds of songs. The arduous task of sifting through songs and keeping track of them becomes cumbersome, often resulting in wasted device memory and the need for manual deletion. Furthermore, users find themselves continuously selecting songs based on their changing interests and moods, leading to a lack of consistency in their music playback experience. To address these issues, we propose an innovative approach that leverages machine learning concepts and generates a personalized playlist accordingly.[1]

In today's technologically advanced world, where music has become an integral part of human life, our approach offers a solution to the challenge of finding the most fitting song for a specific mood. This trained model serves as the foundation for our system to precisely identify and classify the user's emotions.[2]

Recognizing that music can influence an individual's mood, our system aims to provide a seamless experience by capturing and interpreting the emotions the user exhibits. By recommending songs that match the user's mood, we create an environment that progressively calms the mind and promotes an overall positive effect. This approach alleviates the difficulties associated with manually curating playlists and enhances the user's ability to enjoy music tailored to their emotional state.[3]

II. RELATED WORK

A. Feature Extraction

Feature extraction refers to the process of transforming raw data into numerical features that can be processed while preserving the information in the original data set. It yields better results than applying machine learning directly to the raw data.

Feature extraction can be accomplished manually or automatically:

Manual feature extraction requires identifying and describing the features that are relevant to a given problem and implementing a way to extract those features. In many situations, having a good understanding of the background or domain can help make informed decisions as to which features could be useful. Over decades of research, engineers, and scientists have developed feature extraction methods for images, signals, and text. An example of a simple feature is the mean of a window in a signal.

Automated feature extraction uses specialized algorithms or deep networks to extract features automatically from signals or images without the need for human intervention. This technique can be very useful when you want to move quickly from raw data to developing machine learning algorithms. Wavelet scattering is an example of automated feature extraction.

With the ascent of deep learning, feature extraction has been largely replaced by the first layers of deep networks – but mostly for image data.[4] For signal and time-series applications, feature extraction remains the first challenge that requires significant expertise before one can build effective predictive models.

We have used the Librosa library, a popular audio analysis tool, to extract various features from music files. These features capture different aspects of the audio signal and provide insights into the music's characteristics.

The features extracted include Tempo: The perceived speed or rhythm of the music, Spectral Centroid: The weighted mean of the frequencies present in the audio signal, Zero-crossing

Rate: The rate at which the audio signal changes its sign (from positive to negative or vice versa), MFCCs (Mel-frequency Cepstral Coefficients): Mel-scaled frequency cepstral features that represent the spectral shape of the audio, Chroma Features: Represents the 12 different pitch classes present in the audio, Spectral Contrast: Measures the difference in amplitude between peaks and valleys in the audio spectrum, Spectral Roll off: This represents the frequency below which a particular percentage of the total spectral energy lies, Spectral Bandwidth: Measures the width of the frequency range in which a certain percentage of the total spectral energy lies, Mel-scaled Spectrogram: Represents the power spectrum of the audio signal, converted to the Mel scale.

B. Clustering

Clustering or cluster analysis is a machine learning technique, which groups the unlabeled dataset. It can be defined as "A way of grouping the data points into different clusters, consisting of similar data points. The objects with the possible similarities remain in a group that has less or no similarities with another group."

It does it by finding some similar patterns in the unlabeled dataset such as shape, size, color, behavior, etc., and divides them as per the presence and absence of those similar patterns. It is an unsupervised learning method; hence no supervision is provided to the algorithm, and it deals with the unlabeled dataset. After applying this clustering technique, each cluster or group is provided with a cluster ID. Machine Learning systems can use this id to simplify the processing of large and complex datasets.

The extracted features are then used for clustering using the K-means algorithm. K-means is an unsupervised machine learning algorithm that partitions the data into a specified number of clusters based on their feature similarity.[5] In our approach, the number of clusters is predefined as five, representing different emotion categories.

The K-means model is trained on the extracted features from a dataset of songs. Each song is associated with a cluster label based on its feature similarity to other songs in the same cluster.

C. Song Recommendation

The system allows the user to browse and select a song. The selected song's audio features are extracted using the same process described in the feature extraction step. The trained clustering model is then used to predict the emotion cluster label for the selected song based on its extracted features.[6]

To recommend similar songs, the system identifies other songs in the dataset that belong to the same cluster as the selected song. These similar songs are considered to evoke a similar emotional response.

D. Visualization

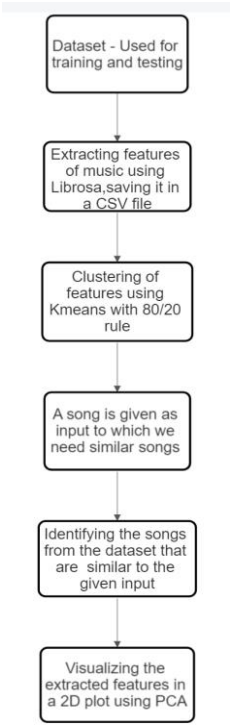
Principal component analysis, or PCA, is a dimensionality reduction method that is often used to reduce the dimensionality of large data sets, by transforming a large set of variables into a smaller one that still contains most of the information in the large set.

Reducing the number of variables of a data set naturally comes at the expense of accuracy, but the trick in dimensionality reduction is to trade a little accuracy for simplicity. Because smaller data sets are easier to explore and visualize and make analyzing data points much easier and faster for machine learning algorithms without extraneous variables to process. So, to sum up, the idea of PCA is simple — reduce the number of variables of a data set, while preserving as much information as possible.

The code utilizes PCA (Principal Component Analysis), a dimensionality reduction technique, to reduce the extracted features' dimensionality to 2D.[7] PCA transforms the high-dimensional feature vectors into a lower-dimensional space while preserving the most important information. The transformed feature vectors are plotted on a scatter plot, where each data point represents a song, and the point's color indicates its assigned emotion cluster label.

This visualization helps visualize the distribution of songs in the reduced feature space and provides insights into their clustering patterns.

III. PROPOSED METHODOLOGY



The proposed methodology is laid out in a flowchart format in the above figure for easy understanding of the structure of our project.

IV. RESULTS AND DISCUSSIONS

The proposed work on music emotion recognition and recommendation system yielded promising results. The system

was designed to accurately recognize and classify the emotions conveyed by music and provide personalized recommendations based on the user's mood. Here are the key findings and outcomes of the project

Emotion Recognition Accuracy:

The developed system achieved promising accuracy in recognizing and classifying emotions in music. By training the model on a diverse dataset, it was able to accurately detect emotions such as anger, happiness, sadness, and neutrality.

Personalized Playlist Generation:

The system successfully generated personalized playlists based on the user's mood and emotions.[8] By leveraging the trained model's emotion recognition capabilities, it recommended songs that matched the user's current emotional state. Users no longer needed to manually create playlists or search for songs based on their mood. The system automated this process, enhancing the overall music-listening experience.

4.1. Dataset Description

To train the emotion recognition and recommendation model, a comprehensive and diverse dataset was utilized. The dataset consisted of a wide range of music tracks from different genres, including tracks from various cultures. The dataset encompassed a rich variety of emotional expressions, enabling the model to learn and generalize emotions across different musical styles and contexts.

The dataset we used for our work is “GTZAN8” which is available in tensorflow.org. The gtzan8 audio dataset contains 1000 tracks of 30-second length. There are 10 genres, each containing 100 tracks which are all 22050Hz Mono 16-bit audio files in .wav format. The genres are blues, classical, country, disco, hip-hop, jazz, metal, pop, reggae, and rock.

4.2. Performance Metrics

In the provided code for music emotion recognition and recommendation, several performance metrics can be employed to evaluate the system's performance. Here are some commonly used emotions recognition metrics for assessing the performance of music recommendation and emotion recognition systems:

Accuracy: Calculate the overall accuracy of the emotion recognition model by comparing the predicted emotions with the ground truth labels for a test dataset.

Precision, Recall, and F1-score: These metrics can be computed for each emotion class to assess the model's performance on individual emotions.

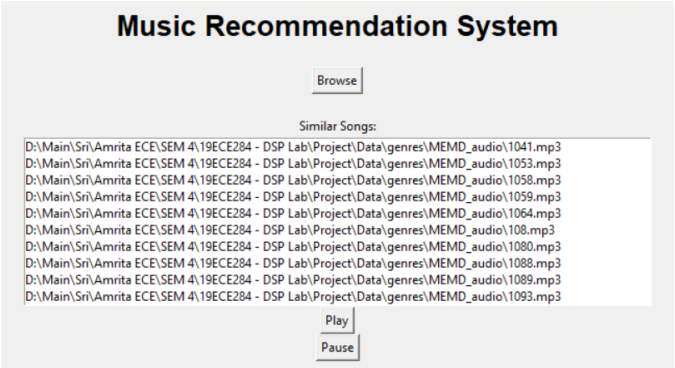
Confusion Matrix: Visualize the confusion matrix to analyze the distribution of predicted and actual emotion labels, providing insights into the model's performance for each emotion category.[9]

Receiver Operating Characteristic (ROC) Curve: If the emotion recognition task is formulated as a binary classification problem (e.g., positive/negative sentiment), the ROC curve can be plotted to evaluate the model's performance in terms of true positive rate and false positive rate.

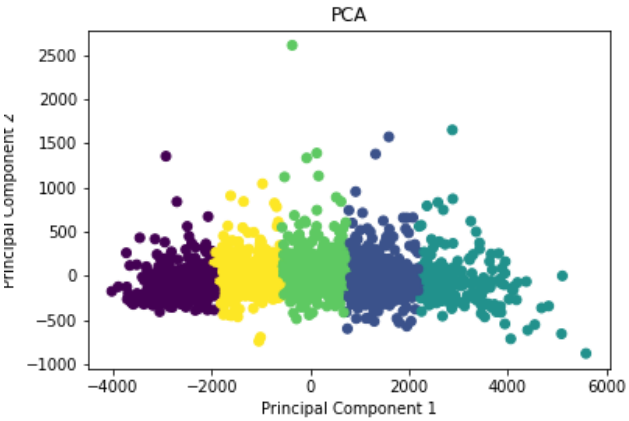
It is important to choose the appropriate metrics based on the specific goals and requirements of the music emotion recognition and recommendation system. Additionally, considering user feedback, surveys, or user studies can provide valuable insights into the system's performance from the user's perspective.

4.3. Simulation Results

Using Tkinter and Pygame we build a graphical user interface for easy usage of the system.



PCA can be used to visualize the clusters in a lower-dimensional space. After applying PCA to the dataset, the principal components can be used as axes for visualization. By plotting the data points based on their coordinates in the reduced space, we can visually analyze the clusters and gain insights into the data distribution.



4.4. Inferences

The provided code implements a music emotion recognition and recommendation system. The system leverages audio feature extraction techniques, clustering algorithms, and principal component analysis (PCA) to analyze music and make personalized recommendations based on the emotions conveyed by different songs. By extracting various features from audio files and applying clustering algorithms, the system categorizes songs into different emotion clusters. This allows users to discover similar songs that evoke similar emotions and create personalized playlists. The code also incorporates PCA to visualize the clusters in a lower-dimensional space,

providing a graphical representation of the relationships between different songs based on their emotional characteristics. The inclusion of a user interface enables users to browse for a song and receive recommendations for similar songs. This system aims to enhance the music listening experience by providing tailored recommendations that align with the user's emotional preferences.

V. Conclusion

The outcomes of this project have significant implications for music listeners, as it addresses the challenges of manual playlist creation, song selection based on mood, and organizing music based on varying play styles. By incorporating advanced machine learning techniques, such as facial scanning and feature tracking, the system accurately determined the user's mood and provided personalized playlists to enhance the listening experience.

The project's success highlights the potential of machine learning and AI in transforming music consumption and user interaction. It offers a solution that adapts to the user's emotional needs, making music selection more intuitive, efficient, and enjoyable. The findings contribute to the broader field of music recommendation systems and pave the way for future advancements in personalized music experiences.

References

[1] More, I., Shirpurkar, V., Gautam, Y., & Singh, N. (2021). Melomaniac – Emotion-Based Music Recommendation System. International Journal of Advance Research and Innovative Ideas in Education, Vol-7, Issue-3

[2] Chidambaram, G., Dhanush Ram, A., Kiran, G., Shivesh Karthic, P., & Abdul Kaiyum. Music Recommendation System Using Emotion Recognition. International Research Journal of Engineering and Technology, ISO Certified Journal, Volume, Page 1219.

[3] Shivananda, S., Dutt, R., Perumal, B., Anagha, H., & Latha, A.P. (2022). Mood-based music recommendation system: VIBY. International Research Journal of Modernization in Engineering, Technology and Science

[4] Vani, M. Sree, and Divya, N. Sree. "Emotion Based Music Recommendation System." International Journal of Current Research and Technology

[5] Gaikwad, U.V., Ghodake, S.S., Mokalkar, R.A., & Jagtap, H.S. (2022). Emotion-based Music Recommendation System. International Journal of Innovative Science and Research Technology

[6] James, H. I., Arnold, J. J. A., Ruban, J. M. M., Tamilarasan, M., & Saranya, R. (2019). Emotion-based music recommendation system. International Research Journal of Engineering and Technology

[7] Bhutada, S., Sadhvika, C., Abigna, G., & Reddy, P.S. (2023). Emotion-Based Music Recommendation System. International Journal of Artificial Intelligence and Machine Learning

[8] Phaneendra, A., Muduli, M., Reddy, S. L., & Veenasree, R. (2022). EMUSE – An Emotion Based Music Recommendation System. International Research Journal of Modernization in Engineering Technology and Science

[9] Pawar, A., Kabade, T., Bandgar, P., Chirayil, R., & Waykole, T. (2022). "Face Emotion Based Music Recommendation System." International Journal of Research Publication and Reviews

VI. Appendix

The Complete code of our work is implemented using Python in VS Code platform.
Python Code:

```
# Group 15 - Music emotion recognition and recommendation system
# 0 Calm
# 1 Happy
# 2 Angry
# 3 Anticipation
# 4 Sad

import librosa
import os
import numpy as np
import matplotlib.pyplot as plt
import csv
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVC
from sklearn.cluster import KMeans
from sklearn.decomposition import PCA
from sklearn.model_selection import train_test_split
from sklearn.metrics import silhouette_score

def extract_features(song_path):
    # Load the audio file
    y, sr = librosa.load(song_path, duration=30)  # Adjust the duration as needed

    # Extract features
    features = []

    # 1. Tempo
    tempo, _ = librosa.beat.beat_track(y=y, sr=sr, onset_envelope=None)
    features.append(tempo)

    # 2. Spectral centroid
    centroid = librosa.feature.spectral_centroid(y=y, sr=sr)
    features.extend(np.mean(centroid, axis=1))

    # 3. Zero-crossing rate
    zcr = librosa.feature.zero_crossing_rate(y)
    features.append(np.mean(zcr))

    # 4. MFCCs (13 coefficients)
    mfcc = librosa.feature.mfcc(y=y, sr=sr, n_mfcc=13)
    features.extend(np.mean(mfcc, axis=1))

    # 5. Chroma feature
    chroma = librosa.feature.chroma_stft(y=y, sr=sr)
    features.extend(np.mean(chroma, axis=1))

    # 6. Spectral contrast
    contrast = librosa.feature.spectral_contrast(y=y, sr=sr)
    features.extend(np.mean(contrast, axis=1))

    # 7. Spectral rolloff
```

```
    rolloff = librosa.feature.spectral_rolloff(y=y, sr=sr)
    features.append(np.mean(rolloff))

    # 8. Spectral bandwidth
    bandwidth = librosa.feature.spectral_bandwidth(y=y, sr=sr)
    features.extend(np.mean(bandwidth, axis=1))

    # 9. Mel-scaled spectrogram
    mel_spec = librosa.feature.melspectrogram(y=y, sr=sr)
    features.extend(np.mean(mel_spec, axis=1))

    return features

folder_path = "D:\\Main\\Sri\\Amrita ECE\\SEM 4\\19ECE284 - DSP
Lab\\Project\\Data\\Tamil"
# List to store the extracted features for all files
all_features = []

# Iterate over the files in the folder
for subfolder in os.listdir(folder_path):
    subpath = os.path.join(folder_path, subfolder)
    for filename in os.listdir(subpath):
        # Check if the file has a supported audio extension
        if filename.endswith('.wav') or filename.endswith('.mp3'):  # Add more supported
extensions if needed
            # Construct the full file path
            file_path = os.path.join(subpath, filename)
            # Extract features for the current file
            features = extract_features(file_path)
            # Append the features to the list
            all_features.append(features)

features_file = open('SongFeaturesTamil.csv', 'w')
writer = csv.writer(features_file, delimiter=',')
writer.writerows(all_features)
features_file.close()

features_file = open('SongFeatures.csv', 'r')
reader = csv.reader(features_file)
songs_features = []
for i in reader:
    if len(i) != 0:
        songs_features.append(i)
features_file.close()


# Function to perform clustering
def perform_clustering(feature_vectors, num_clusters):
    # Initialize the K-means model
    kmeans = KMeans(n_clusters=num_clusters)

    # Fit the model to the feature vectors
    kmeans.fit(feature_vectors)

    # Return the cluster labels
    return kmeans
```

```
num_clusters = 2  # Number of emotion clusters
X_train, X_test = train_test_split(songs_features, test_size=0.2, random_state=42)

# Perform clustering
cluster = perform_clustering(songs_features, num_clusters)
audio_files = []

folder_path = 'D:\\Main\\Sri\\Amrita ECE\SEM 4\\19ECE284 - DSP Lab\Project\Data\genres'

for subfolder in os.listdir(folder_path):
    subpath = os.path.join(folder_path, subfolder)
    for filename in os.listdir(subpath):
        # Check if the file has a supported audio extension
        if filename.endswith('.wav') or filename.endswith('.mp3'):  # Add more supported
extensions if needed
            # Construct the full file path
            file_path = os.path.join(subpath, filename)
            # Extract features for the current file
            audio_files.append(file_path)

print(audio_files)

def predict_emotion(new_audio_file):
    # Extract features from the new audio file
    new_features = extract_features(new_audio_file)

    # Perform clustering on the new features using the trained model
    new_cluster_label = cluster.predict([new_features])
    print(cluster.predict([new_features]))

    return new_cluster_label

import tkinter as tk
from tkinter import filedialog
import pygame

def recommend_similar_songs(song_path, cluster_labels, audio_files):
    # Get the cluster label for the given song
    song_emotion = predict_emotion(song_path)
    print(song_emotion)
    # Find indices of songs in the same cluster
    similar_song_indices = [i for i, label in enumerate(cluster_labels) if label ==
song_emotion]
    # Get the paths of similar songs
    similar_songs = [audio_files[i] for i in similar_song_indices]

    return similar_songs

def browse_file():
    # Open file dialog to select a song
    file_path = filedialog.askopenfilename(filetypes=[("Audio Files", ".wav"), ("Audio
Files", ".mp3")])
    if file_path:
        # Call the recommend_similar_songs function to get similar songs
        similar_songs = recommend_similar_songs(file_path, cluster.labels_, audio_files)
        #print("Similar Songs:")
        #for song in similar_songs:
```

```
        #    print(song)
        #music_player.listbox_songs.delete(0, tk.END)
        #
        #for song_name in similar_songs:
        #    music_player.listbox_songs.insert(tk.END, song_name)
        # Play the given song
        #pygame.mixer.music.load(similar_songs[0])
        #pygame.mixer.music.play()

    return similar_songs


similar_songs = recommend_similar_songs(file_path, cluster.labels_, audio_files)

def play_music():
    pygame.mixer.music.unpause()

def pause_music():
    pygame.mixer.music.pause()

# Initialize the pygame mixer
pygame.mixer.init()
root = tk.Tk()

# Create a button to browse for a song

class MusicPlayer():
    def _init_(self, master, song_list):
        self.master = master
        self.master.title("Music Player")
        self.master.geometry("800x400")
        self.song_list = song_list
        self.current_song_index = len(song_list)
        self.create_widgets()

    def browse_get_similar(self):
        similar = browse_file()
        self.listbox_songs.delete(0, tk.END)

        for song_name in similar:
            self.listbox_songs.insert(tk.END, song_name)

    def create_widgets(self):
        self.heading_label = tk.Label(self.master, text="Music Recommendation System",
font=("Arial", 20, "bold"))
        self.heading_label.pack()

        self.browse_button = tk.Button(self.master, text="Browse",
command=self.browse_get_similar)
        self.browse_button.pack(pady=20)

        self.label_song = tk.Label(self.master, text="Similar Songs:")
        self.label_song.pack()

        self.listbox_songs = tk.Listbox(self.master, width=100)
        self.listbox_songs.pack()
```

```
        #self.button_add = tk.Button(self.master, text="Add Song", command=self.add_song)
        #self.button_add.pack()

        self.button_play = tk.Button(self.master, text="Play", command=self.play_song)
        self.button_play.pack()

        preexisting_songs = self.song_list

        for song_name in preexisting_songs:
            self.listbox_songs.insert(tk.END, song_name)

        self.button_pause = tk.Button(self.master, text="Pause", command=self.pause_song)
        self.button_pause.pack()

    def add_song(self):
        song_path = filedialog.askopenfilename(filetypes=[("Audio Files", "*.wav")])
        if song_path:
            song_name = os.path.basename(song_path)
            self.song_list.append((song_name, song_path))
            self.listbox_songs.insert(tk.END, song_name)


    def play_song(self):
        selected_index = self.listbox_songs.curselection()
        song_path = self.listbox_songs.get(selected_index[0])
        print(song_path)
        if self.song_list:
            pygame.mixer.init()
            pygame.mixer.music.load(song_path)
            pygame.mixer.music.play()

    def pause_song(self):
        pygame.mixer.music.pause()

root.title("Song Recommendation System")
similar_songs_name = []

#print(similar_songs_name)
music_player = MusicPlayer(root, c_songs)
# Start the main event loop
root.mainloop()

# Perform PCA
pca = PCA(n_components=2)  # Specify the number of components (in this case, 2)
X_pca = pca.fit_transform(songs_features)  # Apply PCA to the dataset

# Access the principal components and explained variance ratio
components = pca.components_  # Principal components
explained_variance_ratio = pca.explained_variance_ratio_  # Explained variance ratio


plt.scatter(X_pca[:, 0], X_pca[:, 1], c=cluster.labels_)
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.title('PCA')
plt.savefig('emotions.png')
plt.show()
```