Project report

On

MUSIC RECOMMENDATION SYSTEM USING COSINE SIMILARITY

Submitted

In partial fulfilment of the requirements for the degree of

BACHELOR OF SCIENCE

In

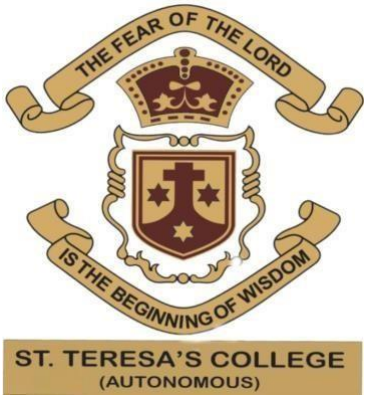
MATHEMATICS

By

ANGELEENA HELEN MARY A A (AB23AMAT005)

Under the supervision of

Smt. SUSAN MATHEW PANAKKAL



DEPARTMENT OF MATHEMATICS AND STATISTICS

ST.TERESA'S COLLEGE (AUTONOMOUS)

ERNAKULAM, KOCHI- 682011

MARCH – 2026

ST.TERESA'S COLLEGE (AUTONOMOUS), ERNAKULAM

ST.TERESA'S COLLEGE (AUTONOMOUS), ERNAKULAM

CERTIFICATE

This is to certify that the dissertation entitled, MUSIC RECOMMENDATION SYSTEM USING COSINE SIMILARITY is a bonafide record of the work done by ANGELEENA HELEN MARY A A under my guidance as partial fulfillment of the award of the degree of Bachelor of Science in Mathematics at St. Teresa's College (Autonomous), Ernakulam affiliated to Mahatma Gandhi University, Kottayam. No part of this work has been submitted for any other degree elsewhere.

Date: 12/12/2025

Place: Ernakulam

ST.TERESA'S COLLEGE (AUTONOMOUS) DEPARTMENT OF MATHEMATICS AND STATISTICS ERNAKULAM

Dr. Susan Mathew Panakkal
Assistant Professor,
Department of Mathematics, St.
Teresa's College (Autonomous),
Ernakulam.

Dr. Elizabeth Reshma M T
Assistant Professor and Head,
Department of Mathematics,
St. Teresa's College (Autonomous)

External Examiners

1:..... 29/01/26

2:.....

I hereby declare that the work presented in this project is based on the original work done by me under the guidance of Dr. Susan Mathew Panakkal, Assistant Professor, Department of Mathematics, St. Teresa's College(Autonomous), Ernakulam and has not been included in any other project submitted previously for the award of any degree.

Place: Ernakulam

Date:.

ANGELEENA HELEN MARY A A (AB23AMAT005)

ACKNOWLEDGEMENT

I must mention several individuals who encouraged us to carry this work. Their continuous invaluable knowledgeable guidance throughout the course of this study helped us to complete the work up to this stage.

I am very grateful to our project guide Dr. Susan Mathew Panakkal for her guidance and support throughout the work. I also want to express my gratitude to my classmates, who assisted me in formulating my research topics by sharing their knowledge. Last but not least, I'd like to express my heartfelt gratitude to God Almighty for always being my emotional rock and teaching us how to confront every challenge with intelligence and confidence.

Place: Ernakulam

Date :

CONTENTS

CERTIFICATE . . . . . *i*

DECLARATION . . . . . *ii*

ACKNOWLEDGEMENT . . . . . *iii*

CONTENTS . . . . . *iv*

1. INTRODUCTION

1.1 Project Overview 7

1.2 Importance and Relevance 8

2. CONVERTING SONGS INTO VECTORS AND FINDING SIMILARITY

2.1 Method 1 - Cosine Similarity 9

2.2 Mathematical Background 11

2.3 Equations Used 13

2.4 Music Data Representation 14

2.5 Feature Extraction 14

2.6 Vectorization Process 15

2.7 Cosine Similarity in Action 16

2.8 Method 2 - Euclidean Method 17

3. PYTHON CODE IMPLEMENTATION

3.1 Overview of Python Code 20

3.2 Code 1 : Song Vectorization 20

3.3 Code 2 : Cosine Similarity Calculation 23

3.4 Challenges in Code Implementation 24

3.5 Testings and Results 25

3.6 Future Improvements 26

4. DESIGN AND DEVELOP A MUSIC RECOMMENTATION SYSTEM 26

5. CONCLUSION 27

6. REFERENCE 28

CHAPTER 1

INTRODUCTION

1.1 Project Overview

i. Relation between Music and Mathematics

Comparing the basic general definitions of mathematics and music implies that they are two very distinct disciplines. Mathematics is a scientific study, full of order, countability and calculability . Music on the other hand is thought to be artistic and expressive. The study of these two disciplines, though seemingly different however are linked and have been for over thousand years. Music itself is indeed very mathematical and mathematics is inherent to many basic ideas in music theory. Music theorists like experts in other disciplines, use mathematics to develop, express and communicate their ideas. Mathematics can describe many phenomena and concepts in music.

ii. Objective

The objectives of the project are :

- To study the method of cosine similarity.
- To study the method of Euclidean and to analyse how it is applicable to music similarity.
- To design and develop a music recommendation system.

The main goal of this project is to design and develop a music recommendation system that can automatically suggest songs similar to a given initial song by analyzing and comparing their features. In today's world, music recommendation systems are a vital part of digital platforms, as they help listeners discover new songs that match their taste. Our project focuses on the mathematical and computational side of this process, where every song is first broken down into measurable features such as energy, bpm, danceability, liveness, chroma and spectral characteristics etc. These features are then represented in numerical form, allowing us to apply mathematical formulas to compare songs. One of the primary techniques we have used is cosine similarity, which measures different feature vectors to determine how alike they are. If the vectors are close, the songs are considered similar; if they are far apart, the songs are less alike. By applying this method, our system can recommend music that has comparable properties to the song chosen by the user. Ultimately, the music recommendation system serves as a simple yet powerful model of how modern music streaming platforms function behind the scenes, making the process of discovering new music more enjoyable, personalized, and efficient by the use of mathematics.

1.2 Importance and Relevance

This project is highly relevant in today's digital era where music consumption is dominated by online platforms and streaming services. With millions of songs available, listeners often face the challenge of discovering music that aligns with their personal taste. Recommendation systems solve this problem by automatically suggesting songs based on a user's preferences, and our project demonstrates the mathematical and computational principles behind such systems.

By using techniques like cosine similarity, our project shows how mathematics can be applied to capture the essence of a song and compare it with others. This approach not only highlights the practical role of mathematics in creative fields like music but also demonstrates the growing influence of Artificial Intelligence (AI) and Machine Learning (ML) in shaping modern entertainment. Recommendation systems powered by AI are already widely used in platforms such as Spotify, YouTube Music, and Apple Music to provide personalized playlists and enhance user experience.

Beyond entertainment, the relevance of this project extends to areas such as music therapy, education, and digital libraries, where identifying similarities between songs can be useful in building mood-based playlists, teaching musical structures, or organizing large music collections. By bridging the gap between music and mathematics, this project provides insight into how AI systems understand human creativity and make technology more personal, engaging, and impactful.

CHAPTER 2

CONVERTING SONGS INTO VECTORS AND FINDING SIMILARITY

2.1 Method 1 – Cosine Similarity

i. Explanation of Cosine Similarity an How it Applies to Music Similarity

Cosine similarity is a mathematical method used to measure how similar two objects are, based on the angle between them in a multi-dimensional space. Instead of looking at the actual values, it focuses on the orientation (or direction) of the data points. The result is a value between 0 and 1, where:

- 1 means the two objects are perfectly similar,
- 0 means they are completely different (no similarity).

ii. Mathematical Definition

If we have two vectors:

$A = (a_1, a_2, a_3, \ldots\ldots, a_n)$

$B = (b_1, b_2, b_3, \ldots\ldots, b_n)$

$A.B$ then cosine similarity is defined as: Cosine Similarity (A, B) $= \frac{}{\|A\|.\|B\|}$ where:

- A .B = $\sum_{i=1}^{n} a_i b_i$ → dot product of vectors
- ||A|| = $\sqrt{\sum_{i=1}^{n} a_i^2}$ → magnitude (length) of vector A
- ||B|| $\sqrt{\sum_{i=1}^{n} b_i^2}$ → magnitude (length) of vector =

Cosine $\frac{\sum_{i=1}^{n} a_i b_i}{\sqrt{\sum_{i=1}^{n} a_i^2} \cdot \sqrt{\sum_{i=1}^{n} b_i^2}}$ similarity(A,B) =

Range of Values

- 1 → Vectors point in the same direction (songs are very similar).
- 0→ Vectors point in opposite directions (completely dissimilar).[4]

iii. Application to Music Similarity

In music, each song can be represented as a feature vector, for example:

Song 1= (tempo, energy, liveness, dancability, loudness, …)

Song 2 = (tempo, energy, liveness, dancability, loudness, …)

By applying cosine similarity, we calculate how close these two vectors are. If the cosine similarity is high (close to 1), the songs share similar musical patterns even if their overall volume or scale differs.

Cosine similarity provides a precise mathematical way to measure how alike two songs are by comparing the angle between their feature vectors. Unlike distance-based measures, it ignores differences in magnitude (like loudness or scale) and focuses on the pattern of features (tempo, valence , energy, bpm, etc.). This makes it highly suitable for music similarity tasks, where two songs can vary in intensity but still share the same structural or stylistic qualities.

2.2 Mathematical Background

i. The Formula for Cosine Similarity

Cosine similarity measures the cosine of the angle between two vectors A and B. It is Cosine Similarity (A, B) $= \frac{A.B}{\|A\|.\|B\|}$

where:

- A.B = $\sum^{n}_{i=1} a_i b_i$ → dot product of vectors
- ||A|| = $\sqrt{\sum^{n}_{i=1} a_i^2}$ → magnitude (length) of vector A
- ||B|| = $\sqrt{\sum^{n}_{i=1} b_i^2}$ → magnitude (length) of vector B

If the value is normalized, it lies between 0 and 1, where 1 = perfectly similar and 0 = no similarity.

ii. Vectors and Their Relation to Music

In mathematics, a vector is simply an ordered list of numbers that represent some characteristics.

Example: A 2D vector = (x, y), a 3D vector = (x, y, z), and in higher dimensions we can have many features.

In music, each song can be converted into a feature vector:

Song = (bpm, energy, loudness, dB, ...)

So, comparing two songs mathematically means comparing their feature vectors in this multi-dimensional space.

iii. Example with 2D Vectors (Simplified)

Let's take two simple vectors:

A = (1, 2), B = (2, 3) Dot

product:

A.B = (1)(2) + (2)(3) = 2 + 6 = 8 Magnitude of

A:

$$||A|| = \sqrt{1^2 + 2^2} = \sqrt{1 + 4} = \sqrt{5}$$

Magnitude of B:

$$||B|| = \sqrt{2^2 + 3^2} = \sqrt{4 + 9} = \sqrt{13}$$

Cosine similarity:

Cos Sim(A,B) = $\frac{8}{\sqrt{5}.\sqrt{13}} \approx \frac{8}{8.06} \approx$ 0.993

This value is very close to 1, which means the two vectors are highly similar.

In music terms, this would mean the two songs have very similar feature patterns, even if their values (like exact tempo or loudness) are slightly different.

iv. 3D Example of Cosine Similarity

Suppose we have two songs, represented as 3D feature vectors:

A = (1, 2, 3), B = (4, 5, 6)

Here each number could represent a simplified feature of the song (e.g., tempo, energy, loudness).

v. Interpretation

The cosine similarity is 0.974, very close to 1.

This means Song A and Song B are highly similar in terms of their features.

Even if their raw values differ (Song B has bigger values), their feature patterns align closely.

2.3 Equations Used

In this project, several mathematical equations are used to calculate the similarity between songs. These equations help us represent songs as numbers (vectors) and then measure how close or far they are from each other. The main equations are:

1. Cosine Similarity Formula

Cosine similarity measures how similar two vectors are by checking the angle between them.

$$\text{Cosine Similarity} = \frac{A.B}{\|A\|.\|B\|}$$

$\vec{A} \cdot \vec{B} \rightarrow$The dot product of the two vectors (multiplying and summing their values).

- $\|\vec{A}\|$ and $\|\vec{B}\| \rightarrow$ The magnitudes (lengths) of the vectors.
- The value always lies between 0 and +1. · +1 → Perfectly similar (same direction).
- 0 → No similarity.

In our case, if two songs have a cosine similarity closer to 1, they are very similar in style or features.

2. Vector Representation of Songs

Each song is converted into a vector of features (like tempo, valence, dancability, energy, etc.). Example:
Song A = [0.7, 0.5, 0.9]
Song B = [0.6, 0.4, 0.8]
These vectors make it possible to apply mathematical formulas like cosine similarity.

3. Normalization Formula

To make fair comparisons, we normalize the vectors so that differences in scale (like loudness or length of song) don't affect the results.

$$\text{Normalized Vector} = \frac{\vec{X}}{\|\vec{X}\|}$$

This ensures all vectors have the same length (magnitude = 1). After normalization, similarity depends only on direction (pattern of features), not on size.

In summary:

- Cosine similarity tells us how close two songs are.
- Vectorization represents each song in numerical form.
- Normalization ensures fair comparison by adjusting different scales.

2.4 Music Data Representation

Songs can be represented as numerical vectors by extracting features from their audio signals. Each feature captures a different characteristic of the song, such as rhythm, harmony, or timbre. Once extracted, these features are combined into a high-dimensional vector.

Music features can be grouped into several categories depending on what aspect of the signal they capture:

1) Bpm (tempo): Indicates the speed of the song, essential for rhythmic comparison.
2) Energy: Measures perceived intensity and activity; higher energy means a more dynamic track.
3) Danceability: Assesses how suitable a song is for dancing based on tempo, rhythm stability, and beat.
4) Loudness: Reflects overall volume; often associated with genre differences.
5) Valence: Represents musical positivity or mood; higher values indicate happier and more positive songs.
6) Speechiness: Quantifies spoken words or vocals in the track.
7) Acousticness: Measures the likelihood a song uses acoustic instruments or sounds.
8) Liveness: Infers if the track is from a live performance.
9) Zero Crossing Rate (ZCR): how often the signal changes sign → relates to noisiness.
10) Spectral Centroid: "brightness" of the sound (average frequency weighted by amplitude).
11) Chroma Features: energy distribution across 12 semitones → captures harmony, chords.

12) Instrumentalness: a measure of how likely a track is to be instrumental rather than containing vocals.

2.5 Feature Extraction

Feature extraction transforms raw audio waveforms into structured features. Songs are represented as vectors by extracting quantitative audio features using specialized algorithms or neural network models. These vectors efficiently encode characteristics of songs, enabling tasks like similarity search, genre classification, and music recommendation systems.

i. How features are extracted

Libraries like Librosa (Python) analyze the waveform to compute descriptors.

Feature extraction from music using mathematical methods involves transforming raw audio signals into meaningful numerical representations. Common techniques often rely on mathematical tools from both the time and frequency domains, using equations and signal-processing algorithms to derive features relevant for analysis, classification, and machine learning tasks.

- Root Mean Square Energy (RMS): Measures the signal's average power, which relates to perceived loudness. Mathematically, for a frame of samples x[n]$x[n]$, RMS energy is:

  RMS=$\sqrt{\frac{1}{N}\sum_{i=0}^{N} x_i^2}$

  where N is the number of samples per frame.

- Zero Crossing Rate (ZCR): Counts how often the signal crosses the zero axis, indicating frequency content and noisiness. It is defined as:

ZCR= $\frac{1}{N-1}\sum_{i=1}^{N-1} 1[y_i . y_{i-1} < 0]$

- Higher ZCR = more noise/ speech

- Spectral Centroid: The "center of mass" of the spectrum, reflecting brightness:

  Centroid = $\Sigma \frac{kfk.|X\leftrightarrow k|}{\Sigma_k |X\leftrightarrow_k|}$

  where *fk* is the frequency at bin k and |Xk| is its magnitude.

  BPM(Tempo) = 60 / period of strongest rhythmic pulse(sec)

$\frac{1}{N}\sum_{i=1}^{N} y_i^2$) Loudness (dB) = 10.log(

$$\text{Danceability} = \frac{\text{mean onset strength}}{max\ onset\ stregth}$$

Liveness = STD(spectral contrast(y))

Valence = normalized centroid

2.6 Vectorization Process

Once features are extracted, they need to be converted into a structured format that algorithms can work with. This is where vectorization comes in.

Step 1: Feature extraction

From each song, we extract many features (for example, spectral centroid, tempo,energy etc.).

A single song may give hundreds or thousands of numbers (because features can change every few milliseconds).

Step 2: Aggregation

To make songs comparable, we summarize these features.

Example: Instead of keeping the spectral feature for every frame of the song, we might take the mean, variance, or median.

Step 3: Normalization

Features may have very different scales (e.g., tempo could be around 100–200, while spectral centroid could be in thousands).

Normalization brings everything to a standard scale, usually between 0 and 1, so no single feature dominates.

Step 4: Dimensionality Reduction

If we have too many features, algorithms may become slow or overfit.

Techniques like PCA (Principal Component Analysis) reduce the number of features while keeping the important information.

i. Example of Vector Representation

Let's say we extract 5 features from Song A and Song B:

- Song A → [0.8, 0.6, 0.7, 0.5, 0.3]
- Song B → [0.7, 0.5, 0.6, 0.4, 0.2]

Here, each number corresponds to a normalized feature. Now each song is just a vector of numbers.

This process transforms "music" into mathematical objects (vectors) that can be compared.

Once features are extracted, they are combined into a feature vector.

Formula: Cosine Similarity(A,B) $= \frac{A.B}{\|A\|.\|B\|}$

2.7 Cosine Similarity in Action

Cosine similarity is a mathematical measure used to determine how similar two non-zero vectors are by calculating the cosine of the angle between them. In the context of music, cosine similarity is widely used to quantify how close two songs are in terms of their features, such as melody, harmony, rhythm, or audio embeddings, rather than factors like loudness or volume.

Formula:

Cosine Similarity(A,B) $= \frac{A.B}{\|A\|.\|B\|}$

i. How Cosine Similarity Works

Cosine similarity evaluates the direction rather than the magnitude of the vectors being compared. If two vectors (for example, song feature representations) point in the same direction, their cosine similarity will be 1 (maximum similarity). If they are orthogonal, the value will be 0 (no similarity). The formula is:

ii. Application to Music Similarity

Music tracks are converted into vector embeddings using techniques like deep learning on audio features (e.g., spectral features, spectrograms).Cosine similarity is used to compare these vectors, focusing on the shape or structure of the acoustic features rather than their energy or volume.

This allows for finding musically similar tracks, even if they differ in loudness or genre, by matching core patterns, harmonics, or rhythm.Modern music recommendation systems and discovery tools use cosine similarity to suggest songs that match a listener's tastes based on audio likeness, not on metadata like artist or album name.

iii. Advantages in Music Analysis

Cosine similarity excels in scenarios where comparing the pattern of audio matter more than absolute levels (e.g., song dynamics).It enables content-based recommendations, identifying lesser-known tracks with similar musical structures to popular favorites, even when little usage data exists .In summary, cosine similarity is a fast, reliable approach to identifying music tracks that sound alike, by comparing the internal structure of their audio features without bias from loudness or popularity.

Using features such as bpm (tempo), energy, danceability, loudness, valence, speechiness, acousticness, and liveness for music similarity with cosine similarity is a common approach, especially for datasets like Spotify's, and is generally effective for capturing broad musical qualities and personal preferences.

iv. Effectiveness of these Features

These features cover major aspects of how songs feel and sound: mood (valence), rhythmic drive (bpm, danceability), intensity (energy, loudness), speech-related aspects (speechiness), acoustic presence (acousticness), and live-performance elements (liveness).

Studies show significant differences between genres and song types along these dimensions, supporting their utility in distinguishing musical styles and listener engagement.Combined, they provide good correlations with listener preferences and can yield high accuracy for recommending similar tracks or grouping music by emotional or functional appeal.

v. Limitations to Consider

While these features are robust for broad similarity (e.g., matching dance music to other dance tracks), they may miss more nuanced musical details, like melodic or harmonic similarity, which requires more specialized audio features (e.g., chroma, spectral).

For very detailed music analysis, adding spectral or tonal features can improve precision, especially when distinguishing tracks within similar genres or identifying covers/remixes.

These features are widely used by Spotify and other streaming platforms, as they balance interpretability with computational efficiency. They work well for personal recommendations, playlist generation, and broad clustering tasks.

In summary, bpm, energy, danceability, loudness, valence, speechiness, acousticness, and liveness form a strong foundation for music similarity estimation with cosine similarity algorithms, capturing the main elements that influence listener experience. They are a widely accepted and effective set of audio features for estimating music similarity using cosine similarity. These features collectively capture aspects of rhythm, mood, intensity, speech content, acoustic characteristics, and live performance presence, all of which contribute meaningfully to how similar two songs will feel to listeners.

However, more nuanced similarity—such as matching melody, harmony, or intricate musical patterns—may require deeper audio features (such as chroma ,spectral).

2.8 Method 2 – Euclidian Method

The Euclidean method measures the straight-line distance between two points (or vectors) in a multidimensional space. In simple terms, it calculates how far apart two songs are based on their feature values.

If we think of each song as a point in space (defined by features like tempo, pitch, energy, etc.), then the Euclidean distance tells us how similar or different those songs are.

Smaller distance → songs are more similar

Larger distance → songs are less similar

Formula for Euclidean Distance

For two vectors A = (a_1, a_2, …, a_n) and B = (b_1, b_2, …,b_n):

$$d(A,B) = \sqrt{(a_1 - b_1)} = \sqrt{(a_1 - b_1)^2 + (a_2 - b_2)^2 + \ldots\ldots + (a_n - b_n)^2}$$

Example

Suppose we have two songs represented by just two features:

Song 1 vector = (3, 4)

Song 2 vector = (6, 8)

$$d = \sqrt{(3-6)^2 + (4-8)^2} = \sqrt{(-3)^2 + (-4)^2} = \sqrt{9+16} = \sqrt{25} = 5$$

So, the Euclidean distance = 5.

CHAPTER 3

PYTHON CODE IMPLEMENTATION

3.1 Overview of Python Code

i. General Explanation

This code is designed to build a simple music recommendation system. It allows to upload multiple songs and then uses the librosa library to extract important musical features from each track, such as tempo (BPM), energy, loudness, danceability, valence (happiness/brightness), acousticness, speechiness, instrumentalness, and other spectral and chroma features. These features act like a unique "profile" for each song. Since the features have different scales, the code normalizes them (except loudness in dB) so they can be fairly compared. Then, it calculates how similar the songs are to each other using two mathematical approaches: cosine similarity (pattern-based similarity) and Euclidean distance (distance between feature points). It uses these similarity scores to recommend the most similar song to a given input song.

ii. Purpose

This code find and recommend songs that sound similar to each other based on their musical features. Normally, when you look at an audio file, it's just raw sound data, which humans can hear but computers cannot directly understand. This code translates songs into measurable features like tempo, energy, and loudness, then compares those features mathematically. By doing this, it makes it possible to group similar songs together and suggest "if you liked this song, you may also like that one." The purpose of this code is to analyze audio files, compare their musical characteristics, and recommend songs that sound alike, giving a basic demonstration of how real-world music recommendation systems are built.

iii. Why can't the value of decibel be normalized?

The value of dB (decibel) is a logarithmic scale, not a linear scale. It doesn't increase evenly — every 10 dB step means the sound is 10 times louder in power. The range of dB values is very wide and uneven, it can go from negative numbers like –60 dB for silence up to +10 dB or more for very loud sounds. If try to normalize dB directly into 0–1, the small changes at higher dB levels won't matter much, but small changes at lower dB levels (like –50 vs –40) will look huge. The scale becomes misleading.

3.2 Song Vectorization

i. Explanation

This code is designed to take a raw audio file, extract its important musical features, and then convert those features into normalized numbers between 0 and 1 (except dB) so they can be compared with other songs. The user uploads a song, which is then loaded as a waveform along with its sampling rate. Using Librosa functions, the code extracts several features: tempo (BPM) from beat tracking, energy using root mean square (RMS), loudness by converting the signal's power to decibels, danceability by measuring beat strength regularity, liveness from changes in spectral contrast, valence from brightness (spectral centroid), acousticness as the inverse of spectral flatness, and speechiness from the zero-crossing rate. These features describe how energetic, bright, rhythmic, or speech-like the track is. Since different features have very different scales, the code applies a normalization function that maps all values into the same 0–1 range (except dB) using predefined limits. The result shows each feature and its normalized value.

ii. Functions

Two libraries are used in this code.

1. Librosa

Library purpose: Librosa is the main library used here to analyze audio signals. It helps turn a raw song into features.

Key functions:

Librosa.load(): loads the audio file and gives the waveform and sampling rate.

Librosa.beat.beat_track(): finds the tempo (BPM) by detecting beats in the music.

Librosa.feature.rms(): calculates energy by measuring loudness strength over time.

Librosa.feature.spectral_centroid(): measures the “center of mass” of frequencies (how bright the sound is).

Librosa.feature.spectral_contrast(): captures differences between peaks and valleys in the spectrum (used for liveness).

Librosa.feature.spectral_flatness(): checks how “noise-like” or “tone-like” the sound is (used for acousticness).

Librosa.feature.zero_crossing_rate(): counts how often the signal crosses zero, which indicates how speechlike or percussive the sound is.

Librosa.onset.onset_strength(): measures the strength of note onsets (used to approximate danceability).

---

2. NumPy

Library purpose: NumPy is used for math operations on the audio data and features.

Key functions:

Np.mean(): takes the average of values (e.g., mean energy, mean spectral centroid).

Np.log10(): converts power to decibels (dB) for loudness.

Np.clip(): keeps normalized values inside the range 0–1.

3.3 Cosine Similarity Calculation

i. Explanation

This code is used to find how similar songs are to each other based on their features using cosine similarity. Each song is represented as a list of numbers (features like energy, bpm, danceability, etc.), and cosine similarity checks how close the “direction” of two such lists is.

A function is created to calculate cosine similarity by dividing the dot product of two vectors by the product of their lengths (norms). The target song’s features are stored, along with a playlist of other songs. The code

compares the target song with every song in the playlist, calculates the similarity values, and stores them. It sorts the results from most similar to least similar. It prints the similarity score for each song and shows which one is the most similar.

ii. Functions

In this code, we only used NumPy.

NumPy

Library purpose: NumPy is the core Python library for numerical operations, arrays, and linear algebra.

Key functions:

Np.dot(a, b): computes the dot product of two vectors.

Np.linalg.norm(a): computes the length (magnitude) of a vector.

3.4 Challenges in Code Implementation

i. Challenges Faced

- Data loading – Songs usually come in big audio files (like MP3). Loading many of them can be slow, and sometimes the files are missing or in different formats.

- Feature extraction – To compare songs, we need numbers (features) like energy, tempo, or spectral centroid. Extracting these takes time and computer power, especially for long songs.

- Different scales of features – Some features are tiny (like 0.05 for speechiness), while others are very large (like 3500 for spectral centroid). If not scaled, the big numbers will dominate the similarity result.

ii. How it Solved

•Data loading → Instead of loading all songs at once, songs are processed one by one. Files are converted to a common format before analysis.

•Feature extraction → To save time, extracted features are stored in a database so we don't need to extract them again later.

•Different scales of features → Features are normalized so that small values (like 0.05) and big values (like 3500) are brought to a similar scale. This way, all features contribute fairly to similarity.

3.5 Testing and Result

```
    np.array([0.2395, 0.5429, 0.0694, 0.8029, 0.3675, 1.0, 0.1093, 0.969]),
]

# Compute similarity for each song in playlist
similarities = []
for i, song in enumerate(playlist, 1):
    sim = cosine_similarity(target_song, song)
    similarities.append((i, sim))

# Sort results by similarity (descending)
similarities.sort(key=lambda x: x[1], reverse=True)

# Print similarity values for all songs
for idx, sim in similarities:
    print(f"Song {idx} similarity: {sim:.4f}")

# Get the most similar song
most_similar_song = similarities[0]
print(f"\nMost similar song is Song {most_similar_song[0]} with a similarity of {most_similar_song[1]:.4f}")
```

```
Song 1 similarity: 0.9986
Song 3 similarity: 0.9963
Song 2 similarity: 0.9777

Most similar song is Song 1 with a similarity of 0.9986
```

Each song in the playlist was compared with the target song, and the program returned cosine similarity scores close to 1 (e.g., 0.9986, 0.9963, 0.9777). In cosine similarity, values close to 1 indicate a very high degree of similarity, which matches the expectation since all the songs in this playlist have feature values that are very close to the target song.

ii. How the Output of the Code Validates the Success of the Project?

The output shown in this code run validates the success of the project because it demonstrates that the similarity computation works correctly and produces meaningful results.Moreover, the output is sorted in descending order, so the most similar songs appear at the top, which shows that the sorting and ranking logic is also functioning properly. This step-by-step matching, scoring, and ranking proves that the code can successfully analyze song vectors, compute similarity using the chosen mathematical approach, and provide interpretable results — fulfilling the project's goal of identifying which songs are most similar to a given track.

3.6 Future Improvements

i. Potential Improvements for Better Result

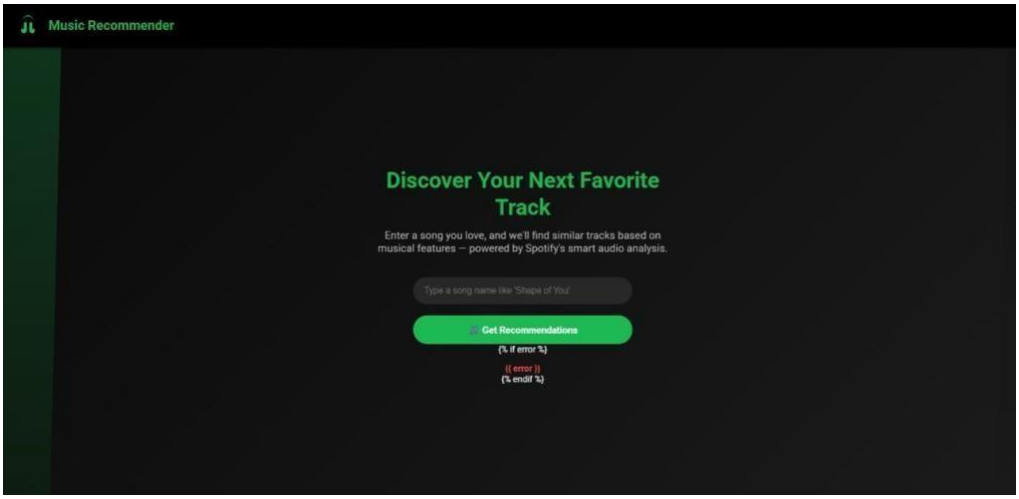
To improve the accuracy and usefulness of the music similarity system, several enhancements can be explored. Richer features such as MFCCs, spectral contrast, and harmonic descriptors can be added, or advanced audio embeddings (e.g., OpenL3, VGGish) can be used for deeper representation of songs. Integrating audio features with metadata (genre, lyrics, artist, or listener behavior) can create a hybrid system that offers more personalized and context-aware music recommendations.

CHAPTER 4

DESIGN AND DEVELOP A MUSIC RECOMMENDATION SYSTEM

In this project, we developed a Music Recommendation System, focusing primarily on designing and coding the user interface. During the development phase, we encountered certain errors and technical issues in the code that prevented us from progressing to the next stage of building a fully functional website. As a result,

the current version serves as a prototype interface, which lays the groundwork for future development. The project can be further enhanced by correcting the existing errors, improving the functionality, and integrating advanced features to create a complete and efficient music recommendation platform.



CHAPTER 5

CONCLUSION

This project successfully demonstrates how mathematical concepts like cosine similarity can be applied to music recommendation. By representing songs as feature vectors derived from their audio characteristics, we were able to quantify similarity between tracks in a precise and meaningful way. The implementation highlights how mathematics and computer science can work together to create practical solutions in the field of music technology.

In our project, we applied cosine similarity to measure the degree of similarity between songs based on their musical attributes such as energy, BPM, loudness (dB), danceability, liveness, valence, speechiness, and acousticness. The cosine similarity approach proved effective because it focuses on the orientation of feature vectors rather than their magnitude, making it well-suited for comparing songs with varying scales of attributes. Our results show that cosine similarity can reliably identify songs that share similar patterns and characteristics, which can be valuable for applications such as music recommendation systems, playlist generation, and genre classification. Overall, this project highlights the practical use of mathematical concepts in analyzing and organizing music data.

In addition to feature extraction and similarity calculation, we also developed a website that allows users to input a song and receive recommendations for similar tracks. This platform provides an

REFERENCE

1) Cosine Similarity for Recommender System: An Honors Research Project on Music Recommendation - Eden Huang

2) An exploration of the relationship between mathematics in music
S Shah - 2010 - eprints.maths.manchester.ac.uk

3) Music recommendation system based on Cosine Similarity and Supervised Genre Classification. (Jamie Mayliana Alyza, Fandy Setyo Utomo, Yuli Purwati, Bagus Adhi Kusuma, Mohd Sanusi Azmi)

4) A methodology combining cosine similarity with classifier for text classification K Park, JS Hong, W Kim - Applied Artificial Intelligence, 2020 - Taylor & Francis