Pengembangan Sistem Rekomendasi Lagu untuk Playlist Menggunakan Cosine Similarity

Mahesa Fadhillah Andre 13523140
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13523140@itb.ac.id, mahesa0208@gmail.com

***Abstrak*—Aplikasi-aplikasi musik seperti Spotify, Apple Music, dan Sound Cloud memiliki sistem rekomendasi lagu yang disediakan bagi penggunanya. Sistem rekomendasi lagu ini bermanfaat untuk membantu pengguna sugesti-sugesti lagu yang cocok dengan seleranya. Pada ilmu Aljabar Linier dan Geometri, terdapat rumus bernama cosine similarity yang, secara teori, dapat meningkatkatkan akurasi dari sistem rekomendasi.**

***Keywords*—Rekomendasi, lagu, similarity**

I. PENDAHULUAN

Musik sudah menjadi bagian integral dalam kehidupan sehari-hari banyak orang. Sensasi yang didapatkan dari musik tidak hanya untuk hiburan, melainkan juga untuk menenangkan pikiran, meningkatkan focus, dan bentuk terapi. Platform seperti Spotify dan Apple Music memudahkan akses kita untuk mendengar musik di mana pun dan kapanpun.

Setiap platform yang menyediakan akses terhadap musik umumnya menyediakan fitur *playlist* yang memberi penggunanya kemampuan untuk menyimpan dan menyusun sejumlah lagu berdasarkan preferensi pengguna platform tersebut. Pengguna diberi kebebasan dalam menentukan dasar penyusunan dari sebuah playlist (contoh: Berdasarkan tahun rilis, genre, dan artis).

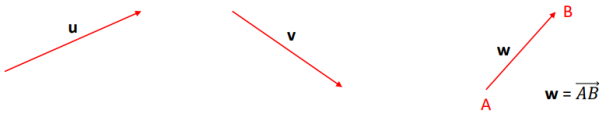
Setiap playlist yang dibuat memiliki isi lagu yang bervariasi. Semua lagu yang terdapat pada sebuah playlist dapat menggambarkan karakteristik dari sebuah playlist. Pengguna pada umumnya, apabila ingin menambahkan sebuah lagu baru pada playlist yang dimilikinya, ingin lagu yang dipilih memiliki karakteristik yang mirip dengan karakteristik playlistnya. Oleh karena itu, diperlukan sebuah sistem yang dapat merekomendasi lagu yang cocok dengan playlist yang dimiliki pengguna.

Makalah ini bertujuan untuk mengembangkan sistem rekomendasi lagu berbasis *cosine similarity* untuk memberi rekomendasi terukur dan sesuai dengan susunan playlist pengguna. Fokus pada karakteristik musik yang digunakan adalah pada komponen tempo, energi, dan danceablitity.

II. LANDASAN TEORI

A. *Vektor dan Ruang Vektor*

Vektor adalah sebuah konsep dalam matematika dan fisika yang merepresentasikan suatu arah yang memiliki arah. Vektor dilambangkan oleh huruf kecil yang dicetak tebal atau memiliki tanda panah di atas hurufnya.



Gambar 2.1 Vektor pada ruang 2D
(Sumber: https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2024-2025/Algeo-11-Vektor-di-Ruang-Euclidean-Bag1-2024.pdf)

Vektor terdiri 2 komponen, yaitu besar (magnitudo) dan arah. Besar vektor menyatakan kuat atau seberapa besar sebuah vektor, sedangkan arah menyatakan arah dari vektor itu sendiri. Contoh konsep vektor yang sering dijumpai sehari-hari adalah kecepatan yang merupakan kuantifikasi seberapa cepat suatu objek bergerak dalam sebuah arah.

Sebuah vektor memiliki sebuah dimensi. Vektor berdimensi dua berarti vektor tersebut memiliki dua elemen di dalamnya (**u** = (U1, U2)). Vektor berdimensi 3 memiliki 3 elemen di dalamnya, dan seterusnya.

Ruang vektor atau ruang Euclidean merupakan ruang tempat vektor didefinisikan. Sebuah ruang vektor memiliki dimensi. Ruang vektor berdimensi dua berarti ruang vektor tersebut hanya dapat terdiri dari vektor berdimensi dua. Hukum ini berlaku untuk semua ruang vektor berdimensi n.

B. *Cosine Similarity*

Cosine similarity adalah metode pengukuran kesamaan antara dua buah vektor dalam ruang berdimensi Metode ini menghitung kemiripan berdasarkan nilai cosinus yang berada dalam rentang -1 hingga 1. Nilai cosinus mendekati satu menandakan kedua vektor memiliki kemiripan (besar sudut antara kedua vektor mendekati 0). Sedangkan nilai cosinus yang mendekati 0 menandakan kedua vektor tidak memiliki kesamaan (sudut antara kedua vektor adalah 90

derajat). Jika nilai vektor mendekati nilai -1, maka kedua vektor memiliki sifat berlawanan (sudut antara kedua vektor 180 derajat). Rumus cosine similarity antara dua vektor adalah:

Gambar 2.2 Rumus cosine similarity
(Sumber: https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2023-2024/Algeo-14-Aplikasi-dot-product-pada-IR-2023.pdf)

Q · D adalah perkalian dot product dari dua vektor, dan ‖Q‖ dan ‖D‖ adalah norma dari kedua vektor.

Dot product adalah hasil penjumlahan dari perkalian antar elemen-elemen bersesuaian pada dua vektor. Misal terdapat vektor A dengan elemen [A1, A2] dan vektor B dengan elemen [B1, B2], maka dot product kedua vektor tersebut adalah $A \cdot B = A1 \cdot B1 + A2 \cdot B2$. Norma dari sebuah vektor adalah panjang dari vektor itu sendiri. Panjang vektor dapat dihitung dengan menjumlahkan hasil pangkat kuadrat setiap elemen dalam vektor tersebut, lalu menghitung akar kuadrat dari hasil tersebut.

$$\|A\| = \sqrt{(A1^2 + A2^2 + ... + An^2)}$$

Dengan menggabungkan dua kommponen di atas, cosine similarity dua buah vektor dapat ditemukan dengan menghitung rasio antara dot product dua vektor tersebut dengan hasil kali norma masing-masing vektor.

C. *Karakteristik Musik*

Karakteristik musik adalah fitur/atribut yang dimiliki sebuah lagu yang menggambarkan sebuah aspek dari lagu tersebut. Setiap musik memiliki beberapa karakteristik yang umum digunakan.

1. Tempo

   Tempo adalah ritme dari sebuah lagu yang diukur dengan beats per minute (BPM). Lagu dengan tempo yang cepat cenderung memberi suasana yang energik dan bersemangat. Sedangkan lagu dengan tempo yang rendah cenderung memberi suasana yang tenang.

2. Energi

   Energi menggambarkan intensitas yang diberi dari musik. Nilai energi sebuah lagu berguna untuk menentukan suasana yang diberi oleh lagu. Lagu dengan energi yang tinggi biasanya memiliki tempo yang cepat, beat yang kuat yang memberi suasana energik. Sedangkan lagu dengan energi yang rendah cenderung memiliki volume dan beat yang rendah sehingga memberikan suasana yang lebih tenang.

3. Danceability

   Danceability adalah atribut dari sebuah lagu yang menilai seberapa baik lagu dapat memberikan suasana yang cocok untuk menari berdasarkan elemen ritme, tempo, dan beat yang konsisten. Lagu yang memiliki danceability yang tinggi memiliki beat dan tempo yang cocok untuk suasana menari. Sebaliknya, lagu dengan danceability memberi suasana yang kurang mendukung untuk melakukan aktivitas menari.

4. Valence

   Valence sebuah lagu adalah ukuran yang menunjukan tingkat emosi positif dan negatif yang dihasilkan dari lagu. Valence pada umumnya diukur dengan skala berentang 0 hingga 1 (0% hingga 100%). Valence yang tinggi cenderung memberi suasana yang ceria dan menyenangkan. Contohnya seperti lagu "Happy" oleh Pharrel Williams. Sedangkan, lagu dengan valence rendah cenderung memberi suasana yang gelap, suram, dan muram. Contohnya seperti lagu "Hurt" oleh Johnny Cash.

5. Acousticness

   Acousticness sebuah lagu adalah ukuran yang menunjukkan tingkat kemungkinan sebuah lagu menggunakan alat non elektrik (akustik). Skala yang dipakai acousticness berada pada rentang 0 hingga 1 (0 hingga 100%) Acousticness dengan nilai yang tinggi menyatakan alat musik akustik lebih mendominasi pada lagu. Sebaliknya, nilai acousticness yang rendah menyatakan alat music non akustik/elektrik lebih mendominasi pada lagu.

D. *MPEG-1 Audio Layer 3 (MP3)*

MP3 atau MPEG-1 Audio Layer 3 adalah format code untuk sebuah audio digital yang menggunakan kompresi lossy (sebuah metode kompresi yang menghilangkan beberapa bagian dari file untuk mengurangi ukuran file). MP3 merupakan format file yang sangat umum digunakan untuk menyimpan audio karena dan kualitas ukuran audio dapat disimpan dalam ukuran yang efisien dan kualitas audio yang tersimpan masih cukup baik. File MP3 sering dijumpai pada berbagai platform, seperti servis streaming lagu dan aplikasi penyimpanan audio.

III. PERCOBAAN

A. *Pemilihan Dataset*

Dataset yang digunakan untuk eksperimen sistem rekomendasi lagu untuk playlist dengan pencarian berbasis cosine similarity Bernama "Million Song Dataset + Spotify + Last.fm" yang diambil dari website Kaggle.com. Dataset tersebut terdiri dari tiga subset, yaitu Music Info, User Listening History, dan MP3-Example.

Subset Music Info merupakan file .csv yang berisi track_id, name, artist, spotify_preview_url, spotify_id, tags, genre, year, duration_ms, dan danceability. Subset Music Info memiliki 50.683 lagu. Subset ini berguna untuk mengetahui informasi umum dari sebuah lagu.

Subset User Listening History merupakan file .csv yang berisi track_id, user_id, dan playcount. File ini melacak total waktu user mendengar sebuah lagu.

Subset terakhir adalah MP3-Example yang berisi 1500 file MP3 yang telah dikelompokkan pada folder berdasarkan genre masing-masing. Subset ini memiliki total folder sejumlah 15 dengan setiap folder berisi 100 lagu/file MP3. Pada percobaan ini, dataset yang digunakan hanyalah subset MP3-Example karena algoritma untuk sistem rekomendasi mencari fitur-fitur karakteristik musik dengan mengekstraksi dan mengolah atribut yang terdapat pada file MP3.

Untuk dataset yang digunakan sebagai query/playlist, data yang digunakan adalah subset dari database sendiri. Jumlah file MP3 yang berada dalam playlist pada percobaan adalah n jumlah file dengan n > 10 karena percobaan akan mengambil 10 lagu terakhir yang di-add oleh user sebagai query yang digunakan untuk sistem rekomendasi. Pertimbangan untuk hanya menggunakan 10 lagu terakhir dari playlist disebabkan oleh sistem rekomendasi diinginkan untuk memberi rekomendasi lagu yang paling relevan dengan selera lagu yang disukai oleh pemilik playlist pada saat pencarian lagu.

B. *Representasi Vektor*

Vektor yang akan digunakan pada algoritma adalah vektor berdimensi lima. Setiap elemen pada vektor mewakili nilai dari sebuah karakteristik musik. Karakteristik musik yang digunakan dan direpresentasikan dalam vektor adalah tempo, energi, danceability, valence, dan acousticness. Setiap file MP3 pada dataset dan percobaan akan diubah menjadi bentuk vektor yang sama.

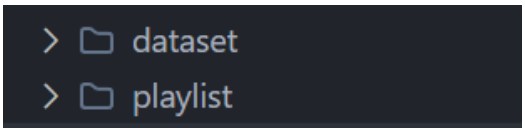
C. *Algoritma*

Algoritma dibuatkan dalam program yang menggunakan bahasa python dengan versi 3.12.4. Python digunakan sebagai bahasa pemrograman dalam pembuatan algoritma karena bahasa python memiliki bahasa code yang simple untuk dimengerti dan juga memiliki library yang lengkap untuk membantu pembuatan algoritma. Algoritma memanfaatkan library numpy untuk membantu perhitungan algoritma, library librosa untuk mengekstraksi fitur dari file MP3, library os untuk load file, dan library scipy untuk menghitung cosine similarity.

```
import os
import numpy as np
import librosa
from scipy.spatial.distance import cosine
```

Gambar 3.1 Library yang digunakan
(Sumber: Dokumentasi pribadi)

Percobaan akan menggunakan dua folder. Folder pertama berisi seluruh dataset file MP3 yang digunakan sebagai tempat penyimpanan database. Folder kedua diisi dengan n buah file MP3 yang dipilih secara acak untuk mewakilkan sebuah playlist.



Gambar 3.4 Folder dataset dan playlist
(Sumber: Dokumentasi pribadi)

Setiap file MP3 akan melalui beberapa tahap pemrosesan. Setiap file MP3 dalam dataset dan playlist akan di-load ke dalam program terlebih dahulu. Proses load file memanfaatkan fungsi os.path.join dari library OS. Proses load dijalankan dengan fungsi load_folder yang menerima folder sebuah folder path.

```
# Function to load all audio files from a folder
def load_folder(folder_path):
    try:
        # Return a list of all .mp3 files in the folder
        return [os.path.join(folder_path, f) for f in os.listdir(folder_path) if f.endswith('.mp3')]
    except Exception as e:
        # Handle errors during folder loading
        print(f"Error loading folder {folder_path}: {e}")
        return []
```

Gambar 3.5 Fungsi load_folder
(Sumber: Dokumentasi pribadi)

Setelah semua file di-load, program akan mengekstraksi vektor fitur dari 10 lagu yang paling terakhir ditambahkan ke dalam playlist (dalam hal ini, program mengambil 10 file MP3 terakhir dalam folder playlist).

Setiap file MP3 akan diekstrak fiturnya menggunakan fungsi extract_features yang menerima file path ke file MP3. Program menyimpan amplitudo dari file MP3. Amplitudo akan digunakan sebagai representasi dari file MP3 yang digunakan untuk mendapatkan semua fitur dalam file MP3.

```
# Load the audio file
y, sr = librosa.load(track, sr=None)
```

Gambar 3.6 Program me-load amplitudo file MP3
(Sumber: Dokumentasi pribadi)

Program akan memastikan sampling rate setiap

pemrosesan adalah “None” untuk menjaga akurasi dalam analisis audio.

Setelah itu, program mengekstraksi tempo dengan fungsi librosa.beat.beat_track untuk mendapatkan beat dalam lagu.

```
# Ekstraksi fitur
tempo, _ = librosa.beat.beat_track(y=y, sr=sr)
tempo = float(tempo)  # Pastikan tempo adalah float tunggal
```

Gambar 3.7 Program mengekstraksi tempo dari file MP3
(Sumber: Dokumentasi pribadi)

Program memastikan tempo merupakan float tunggal agar tidak terjadi error karena perbedaan type pada pemrosesan. Beat yang diekstrak akan digunakan untuk mengukur tempo berdasarkan beats per minute (BPM) untuk mendeteksi kecepatan lagu.

```
# Extract energy as the root mean square of the audio signal
energy = np.mean(librosa.feature.rms(y=y))
```

Gambar 3.8 Program mengekstraksi energy dari file MP3
(Sumber: Dokumentasi pribadi)

Selanjutnya, program akan mendapatkan fitur energy dengan menghitung rerata dari root mean square (RMS) sinyal audio dengan bantuan fungsi librosa.feature.rms.

```
# Compute onset strength as a proxy for danceability
danceability = np.mean(librosa.onset.onset_strength(y=y, sr=sr))
```

Gambar 3.9 Program mengekstraksi danceability dari file MP3
(Sumber: Dokumentasi pribadi)

Fitur danceability didapatkan dengan menghitung onset strength audio. Onset strength berguna untuk mengukur intensitas transien dari sinyal audio untuk mendapatkan hubungan beat dan ritme yang digunakan untuk mengukur danceability sendiri.

Valence dari lagu didapatkan dengan menghitung rerata spectral centroid lagu. Spectral centroid lagu dapat menunjukkan tingkat kecerahan suara.

```
# Calculate spectral centroid as an indicator of valence
valence = np.mean(librosa.feature.spectral_centroid(y=y, sr=sr))
```

Gambar 3.10 Program mengekstraksi valence dari file MP3
(Sumber: Dokumentasi pribadi)

Kecerahan suara yang tinggi menunjukk valence yang cenderung positif. Sebaliknya kecerahan suara yang rendah menunjukkan valence yang cenderung negatif.

```
# Measure spectral bandwidth for acousticness
acousticness = np.mean(librosa.feature.spectral_bandwidth(y=y, sr=sr))
```

Gambar 3.11 Program mengekstraksi acousticness dari file MP3
(Sumber: Dokumentasi pribadi)

Acousticeness diukur dengan menghitung rerata dari spectral bandwidth. Lebar spectral bandwidth memberikan informasi terkait rentang frekuensi yang mencakup sebagian besar energi spektral. Nilai yang rendah berarti suara yang dihasilkan lagu cenderung murni/akustik.

Setelah semua pemrosesan file dijalankan, fungsi extract_features akan mengembalikan vektor fitur dalam bentuk array yang berisi lima elemen. Setiap vektor fitur MP3 akan di-append ke dalam sebuah array bernama features_list pada fungsi extract_features_from_playlist.

```
return np.array(features, dtype=np.float64)
```

Gambar 3.12 Program mengembalikan array numpy
(Sumber: Dokumentasi pribadi)

```
# Fungsi untuk ekstraksi fitur dari track audio
def extract_features(track):
    try:
        # Load file audio (seluruh durasi)
        y, sr = librosa.load(track, sr=None, duration=30)

        if len(y) == 0:
            raise ValueError("File audio kosong atau tidak valid")

        # Ekstraksi fitur
        tempo, _ = librosa.beat.beat_track(y=y, sr=sr)
        tempo = float(tempo)  # Pastikan tempo adalah float tunggal

        energy = float(np.mean(librosa.feature.rms(y=y)))
        danceability = float(np.mean(librosa.onset.onset_strength(y=y, sr=sr)))
        valence = float(np.mean(librosa.feature.spectral_centroid(y=y, sr=sr)))
        acousticness = float(np.mean(librosa.feature.spectral_bandwidth(y=y, sr=sr)))

        # Validasi semua fitur adalah float
        features = [tempo, energy, danceability, valence, acousticness]
        if any(f is None or not isinstance(f, float) or np.isnan(f) for f in features):
            raise ValueError("Fitur yang diekstraksi tidak valid")

        return np.array(features, dtype=np.float64)
    except Exception as e:
        print(f"Error processing {track}: {e}")
        return None
```

Gambar 3.13 Fungsi mengekstraksi fitur dari MP3
(Sumber: Dokumentasi pribadi)

Setelah 10 lagu terakhir dalam playlist diolah menjadi list vektor fitur, program memanggil fungsi calculate_average_vector untuk menghitung rata-rata dari seluruh vektor fitur playlist. Vektor rerata ini akan dipakai sebagai query pada sistem rekomendasi dari database.

```
# Fungsi untuk menghitung rata-rata vektor dari playlist
def calculate_average_vector(feature_vectors):
    if feature_vectors.size == 0:
        raise ValueError("Tidak ada vektor fitur dalam playlist.")

    # Menghitung rata-rata vektor dari semua vektor fitur dalam playlist
    average_vector = np.mean(feature_vectors, axis=0)

    # Validasi bahwa hasilnya tetap 5 dimensi
    if len(average_vector) != 5:
        raise ValueError("Vektor rata-rata yang dihitung tidak memiliki 5 dimensi.")

    return average_vector
```

Gambar 3.14 Fungsi menghitung rerata vektor
(Sumber: Dokumentasi pribadi)

Proses pencarian menggunakan find_top_similar_songs yang menerima dua parameter, yaitu database dan vektor target. Fungsi ini mengembalikan list bernama similarities yang berisi 10 lagu dengan nilai kemiripan tertinggi dengan query vektor fitur yang telah dirata-ratakan.

```python
# Fungsi untuk memproses audio di database dan mencari similarity dengan vektor rata-rata playlist
def find_similarity(database_folder, avg_vector_playlist):
    audio_files = load_folder(database_folder)
    features_database = process_audio_files(audio_files)

    similarities = []

    for idx, song_vector in enumerate(features_database):
        similarity = 1 - cosine(avg_vector_playlist, song_vector)
        similarities.append((os.path.basename(audio_files[idx]), similarity))  # Menggunakan nama file saja

    similarities.sort(key=lambda x: x[1], reverse=True)

    return similarities[:10]
```

Gambar 3.15 Fungsi sistem rekomendasi
(Sumber: Dokumentasi pribadi)

Fungsi find_top_similar_songs memproses semua lagu dalam database menjadi bentuk vektor fitur, sama seperti pemrosesan semua lagu dalam playlist. Namun, setelah diubah menjadi vektor fitur, fungsi ini mencari nilai cosine similarity vektor lagu database, dengan vektor query (vektor fitur 10 lagu terakhir playlist yang telah dirata-ratakan). Cosine similarity dihitung dengan rumus 1 – cosinus dari target vektor dan vektor lagu dari database. Rumus ini mengurangi nilai 1 dengan cosinus karena fungsi *cosine* dalam scipy menghitung jarak cosine (bukan kesamaan). Hasil dari perhitungan cosine similarity disimpan dalam variable similarity yang akan di-append pada list similarities yang telah diinisialisasi pada awal fungsi. Proses ini dilakukan berulang kali hingga semua lagu dalam database telah didapatkan cosines similarity-nya dengan vektor fitur query.

Setelah semua lagu dalam database telah diproses, fungsi akan sortir list similarities berdasarkan nilai kesamaan dalam urutan menurun. Jadi, list akan dimulai dari lagu dengan nilai cosine similarity tertinggi. Setelah pengurutan, fungsi akan mengembalikan list 10 elemen pertama dari list similarities sebagai rekomendasi lagu untuk pemilik playlist.

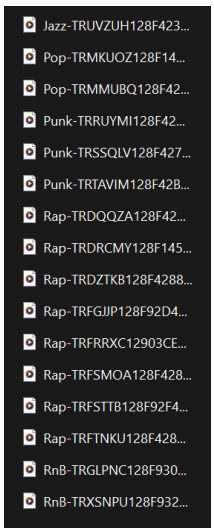
D. Hasil Percobaan

Percobaan menggunakan dataset yang terdiri dari 88 file MP3 karena algoritma kurang efisien jika dipakai untuk memproses data set yang besar.

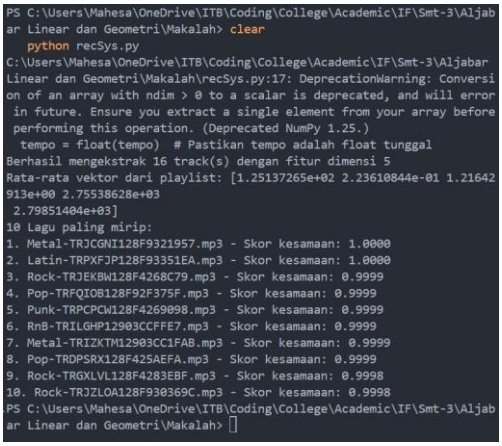
```python
import os
import numpy as np
import librosa
from scipy.spatial.distance import cosine

# Fungsi untuk ekstraksi fitur dari track audio
def extract_features(track):
    try:
        # Load file audio (seluruh durasi)
        y, sr = librosa.load(track, sr=None, duration=30)

        if len(y) == 0:
            raise ValueError("File audio kosong atau tidak valid")

        # Ekstraksi fitur
        tempo, _ = librosa.beat.beat_track(y=y, sr=sr)
        tempo = float(tempo)  # Pastikan tempo adalah float tunggal

        energy = float(np.mean(librosa.feature.rms(y=y)))
        danceability = float(np.mean(librosa.onset.onset_strength(y=y, sr=sr)))
        valence = float(np.mean(librosa.feature.spectral_centroid(y=y, sr=sr)))
        acousticness = float(np.mean(librosa.feature.spectral_bandwidth(y=y, sr=sr)))

        # Validasi semua fitur adalah float
        features = [tempo, energy, danceability, valence, acousticness]
        if any(f is None or not isinstance(f, float) or np.isnan(f) for f in features):
            raise ValueError("Fitur yang diekstraksi tidak valid")

        return np.array(features, dtype=np.float64)
    except Exception as e:
        print(f"Error processing {track}: {e}")
        return None

# Fungsi untuk memuat file audio dari folder
def load_folder(folder_path):
    try:
        audio_files = [os.path.join(folder_path, f) for f in os.listdir(folder_path) if f.endswith('.mp3')]
        if not audio_files:
            print(f"Folder {folder_path} is empty or cannot be loaded.")
        return audio_files
    except Exception as e:
        print(f"Error loading folder {folder_path}: {e}")
        return []

# Fungsi untuk memproses file audio yang sudah dimuat dan mengekstraksi fitur
def process_audio_files(audio_files):
    feature_list = []

    for track in audio_files:
        print(f"Processing {track}...")
        features = extract_features(track)
        if features is not None and len(features) == 5:
            feature_list.append(features)
        else:
            print(f"Skipping {track}: fitur tidak valid")

    if len(feature_list) == 0:
        print("Tidak ada fitur yang berhasil diekstraksi.")
        return np.empty((0, 5))

    return np.array(feature_list)

# Fungsi untuk menghitung rata-rata vektor dari playlist
def calculate_average_vector(feature_vectors):
    if feature_vectors.size == 0:
        raise ValueError("Tidak ada vektor fitur dalam playlist.")

    # Menghitung rata-rata vektor dari semua vektor fitur dalam playlist
    average_vector = np.mean(feature_vectors, axis=0)

    # Validasi bahwa hasilnya tetap 5 dimensi
    if len(average_vector) != 5:
        raise ValueError("Vektor rata-rata yang dihitung tidak memiliki 5 dimensi.")

    return average_vector

# Fungsi untuk memproses audio di database dan mencari similarity dengan vektor rata-rata playlist
def find_similarity(database_folder, avg_vector_playlist):
    audio_files = load_folder(database_folder)
    features_database = process_audio_files(audio_files)

    similarities = []

    for idx, song_vector in enumerate(features_database):
        similarity = 1 - cosine(avg_vector_playlist, song_vector)
        similarities.append((os.path.basename(audio_files[idx]), similarity))  # Menggunakan nama file saja

    similarities.sort(key=lambda x: x[1], reverse=True)

    return similarities[:10]

# Main execution block
if __name__ == "__main__":
    # Path ke folder playlist dan database
    playlist_folder = "C:/Users/Mahesa/OneDrive/ITB/Coding/College/Academic/IF/Smt-3/Aljabar Linear dan Geometri/Makalah/playlist"
    database_folder = "C:/Users/Mahesa/OneDrive/ITB/Coding/College/Academic/IF/Smt-3/Aljabar Linear dan Geometri/Makalah/sm-dataset"

    # Muat file audio dari folder playlist
    playlist = load_folder(playlist_folder)
    # Ekstraksi fitur dari file audio di folder playlist
    features_playlist = process_audio_files(playlist)

    # Output hasil ekstraksi playlist
    if features_playlist.shape[0] > 0:
        print(f"Berhasil mengekstrak {features_playlist.shape[0]} track(s) dengan fitur dimensi {features_playlist.shape[1]}")

        # Hitung rata-rata vektor dari playlist
        average_vector_playlist = calculate_average_vector(features_playlist)
        print(f"Rata-rata vektor dari playlist: {average_vector_playlist}")

        # Ekstraksi fitur dari folder database dan temukan similarity
        top_similar_songs = find_similarity(database_folder, average_vector_playlist)
        print("10 Lagu paling mirip:")
        for idx, (song, score) in enumerate(top_similar_songs, start=1):
            print(f"{idx}. {song} - Skor kesamaan: {score:.4f}")
    else:
        print("Tidak ada fitur yang dapat digunakan di playlist.")
```

Gambar 3.16 Keseluruhan algoritma sistem rekomendasi yang dibangun dalam bahasa python
(Sumber: Dokumentasi pribadi)

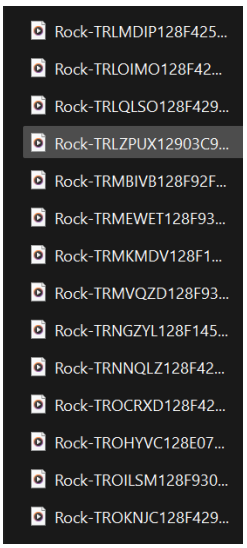
Dilakukan 3 percobaan. Setiap percobaan akan memiliki isi playlist yang berbeda. Pada percobaan pertama, 10 lagu terakhir dalam playlist adalah lagu dengan genre rap dan RnB. Percobaan kedua, 10 lagu terakhir adalah genre rock saja. Percobaan ketiga, 10 lagu terakhir adalah genre pop dan rock.



Gambar 3.17 Folder playlist yang digunakan sebagai query pada percobaan 1
(Sumber: Dokumentasi pribadi)

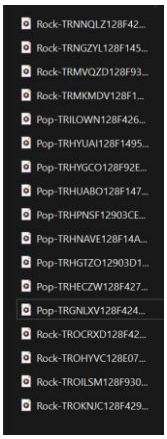


Gambar 3.18 Output program pada percobaan 1
(Sumber: Dokumentasi pribadi)

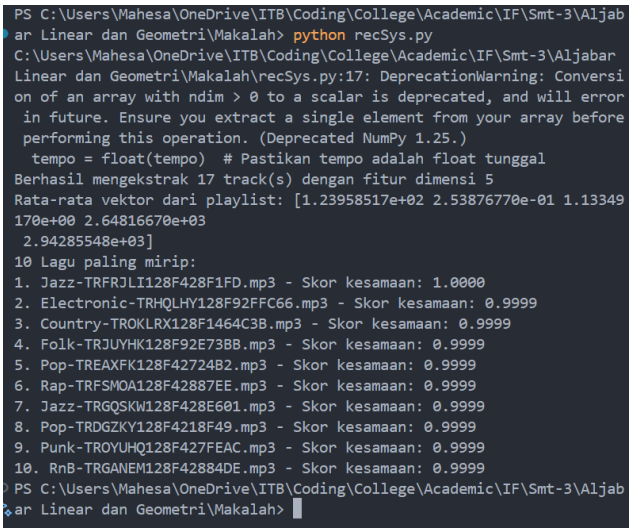


Gambar 3.19 Folder playlist yang digunakan sebagai query pada percobaan 2
(Sumber: Dokumentasi pribadi)

Gambar 3.20 Output program pada percobaan 2
(Sumber: Dokumentasi pribadi)



Gambar 3.21 Folder playlist yang digunakan sebagai query pada percobaan 3
(Sumber: Dokumentasi pribadi)



Gambar 3.22 Output program pada percobaan 3
(Sumber: Dokumentasi pribadi)

IV. KESIMPULAN

Hasil dari percobaan menunjukkan bahwa output yang diberi dari sistem rekomendasi belum cukup akurat dalam mencari lagu-lagu yang sesuai dengan karakteristik

playlist. Hasil dari query dengan 10 lagu rock menghasilkan output berupa lagu-lagu dengan genre selain rock (diharapkan setidaknya genre lagu sama), bahkan tidak ada satupun lagu dengan genre rock pada output. Sistem rekomendasi berbasis cosine similarity belum dapat dinyatakan sebagai sistem yang efektif dalam memberi rekomendasi lagu yang terpersonalisasi.

V. UCAPAN TERIMA KASIH

Pertama, puji syukur dipanjatkan kepada Tuhan Yang Maha Esa karena atas izinnya, penulis dapat menyelesaikan makalah ini dengan kondisi sebaik mungkin. Penulis juga berterima kasih kepada orang tua, saudara, dan teman-teman yang telah mendukung penulis dalam pembuatan makalah ini. Ucapan terima kasih juga penulis beri kepada Dr. Ir. Rinaldi Munir, Dr. Judhi Santoso, M.Sc., dan Bapak Arrival Dwi Sentosa, S.Kom., M.T., selaku dosen mata kuiliah Aljabar Linier dan Geometri yang telah memberikan banyak ilmu yang bermanfaat yang penulis gunakan pada makalah ini dan juga pada perkuliahan. Terakhir, penulis memohon maaf apabila terdapat salah kata dalam penulisan makalah ini, segala hal yang telah ditulis dalam makalah bertujuan untuk memberi ilmu yang bermanfaat bagi pembaca. Penulis berharap makalah ini dapat bermanfaat bagi banyak orang.

REFERENSI

[1] https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2023-2024/Algeo-14-Aplikasi-dot-product-pada-IR-2023.pdf diakses pada 27 November 2024

[2] https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2024-2025/Algeo-11-Vektor-di-Ruang-Euclidean-Bag1-2024.pdf diakses pada 28 November 2024

[3] https://www.kaggle.com/datasets/undefinenull/million-song-dataset-spotify-lastfm?resource=download diakses pada 28 Novermber 2024

[4] https://www.goodaudio.org/blog/what-is-an-mp3-file-and-how-do-i-open-it diakses pada 29 November 2024

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 27 Desember 2024

Mahesa Fadhillah Andre 13523140