Penerapan *Greedy Algorithm* dalam Sistem Rekomendasi Lagu: Studi Kasus Fitur *Recommended Songs* pada Spotify

Rafen Max Alessandro - 13523031
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail: rafen.max@gmail.com , 13523031@std.stei.itb.ac.id

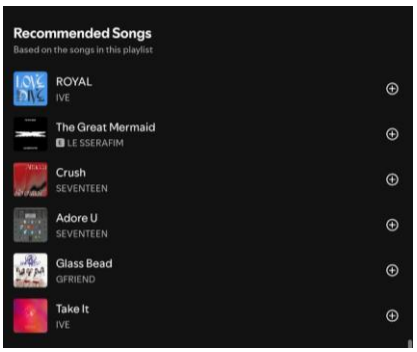
***Abstract*—Spotify merupakan salah satu *platform streaming* musik yang paling banyak digunakan secara global, dan menyediakan fitur *recommended songs* untuk menyarankan lagu-lagu yang dinyatakan sesuai untuk ditambahkan ke dalam *playlist* pengguna. Makalah ini mengkaji fitur *recommended songs* pada Spotify menggunakan pendekatan *greedy algorithm* untuk memilih kandidat lagu dengan nilai kemiripan tertinggi terhadap seluruh lagu dalam *playlist* berdasarkan atribut audio numerik. Studi kasus dilakukan melalui simulasi pemodelan menggunakan *cosine similarity* yang menunjukkan kinerja efisien dan relevan dalam menghasilkan rekomendasi lagu. Sebagai pengembangan lebih lanjut, beberapa tambahan dimensi dapat digunakan untuk memberikan hasil yang lebih dinamis, meliputi *diversity-aware*, *time-aware*, dan *cluster-based*. Ketiga dimensi ini memberikan peningkatan kualitas dan fleksibilitas dalam memberikan rekomendasi lagu berbasis *greedy algorithm* pada aplikasi *platform streaming* lagu.**

Keywords—Cosine Similarity; Greedy Algorithm; Recommended Songs; Sistem Rekomendasi; Spotify

I. PENDAHULUAN

Platform streaming musik secara digital telah menjadi komponen integral dalam kehidupan sehari-hari masyarakat modern, memungkinkan pengguna untuk mengakses jutaan lagu dari berbagai artis dan genre secara mudah dan fleksibel dalam genggaman tangan. Salah satu *platform* yang paling umum digunakan secara global adalah Spotify. Tidak hanya menyediakan layanan pemutaran lagu, Spotify juga mengintegrasikan sistem rekomendasi untuk memberikan pengalaman mendengarkan musik secara personal bagi setiap pengguna. Sistem ini bekerja dengan menyarankan lagu-lagu yang dinilai relevan dengan preferensi dan selera musik pengguna melalui fitur seperti *Discover Weekly, Release Radar,* dan *Recommended Songs*.

Fitur *Recommended Songs* dirancang untuk memberikan saran lagu yang relevan dan koheren untuk ditambahkan ke dalam *playlist* pengguna. Sistem ini mengandalkan informasi dari lagu-lagu yang telah ada di dalam *playlist* untuk menentukan kesamaan secara musikal dari rekomendasi kandidat lagu dengan lagu-lagu tersebut. Perhitungan nilai kesamaan memanfaatkan analisis atribut audio, meliputi *danceability, energy, valence,* dan atribut-atribut lainnya yang telah ditentukan sebelumnya dan dapat diperoleh melalui situs resmi *Spotify Audio Features API*. Proses ini memungkinkan sistem untuk secara otomatis menyusun daftar kandidat lagu yang dinilai sesuai dan relevan untuk berikan sebagai opsi bagi pengguna.



Gambar 1. Fitur *Recommended Songs* pada Spotify
Sumber: dokumentasi pribadi

Salah satu pendekatan yang dapat digunakan untuk menyusun rekomendasi secara efisien adalah *greedy algorithm*, yaitu pendekatan algoritma yang memilih solusi terbaik secara lokal pada setiap langkah, dengan tujuan mencapai solusi optimal secara keseluruhan. Dalam konteks fitur *Recommended Songs*, pendekatan dengan *greedy algorithm* dapat dimanfaatkan untuk memilih lagu dengan hasil perhitungan nilai kesamaan tertinggi terhadap seluruh lagu dalam playlist. Pendekatan ini memberikan keunggulan dalam kesederhanaan implementasi dan efisiensi waktu proses untuk *pool* yang kompleks, tetapi memiliki keterbatasan dalam menjaga keberagaman dan adaptivitas hasil rekomendasi, khususnya terhadap kondisi solusi sub-optimal.

Makalah ini mengkaji penerapan *greedy algorithm* dalam sistem rekomendasi lagu yang digunakan Spotify pada fitur *Recommended Songs*. Melalui studi kasus berbasis simulasi, *greedy algorithm* menggunakan *cosine similarity* diimplementasikan untuk menghasilkan daftar lagu yang direkomendasikan secara otomatis. Hasil studi kasus kemudian dianalisis untuk mengidentifikasi keunggulan dan sejumlah keterbatasan dari pendekatan menggunakan *greedy algorithm*. Analisis ini menjadi dasar konsiderasi pengembangan melalui tiga dimensi tambahan, yaitu *diversity-aware*, *time-aware*, dan

cluster-based. Penambahan ketiga dimensi dalam pendekatan *greedy algorithm* dapat memperluas kemampuan sistem untuk memberikan rekomendasi lagu yang lebih dinamis, tidak terkurung dalam kondisi solusi sub-optimal, dan dalam batas yang tepat secara personalisasi bagi pengguna.

II. LANDASAN TEORI

A. Sistem Rekomendasi

Sistem rekomendasi merupakan salah satu cabang dari sistem penyaringan informasi (*information filtering systems*) yang bertujuan untuk menyajikan informasi dengan dimensi subjektivitas berdasarkan data, pola interaksi, dan preferensi historis sebuah entitas subjek [1]. Sistem ini secara luas diadopsi pada berbagai *platform* digital untuk meningkatkan keterlibatan dan kualitas pengalaman pengguna, termasuk dalam bidang *e-commerce*, media sosial, hingga layanan *streaming* musik dan video.

Secara umum, sistem rekomendasi dibagi ke dalam tiga pendekatan utama [1], yaitu:

1. *Content-based filtering*
   Menyarankan item berdasarkan kemiripan karakteristik dengan item yang pernah disukai oleh pengguna.
2. *Collaborative filtering*
   Menyarankan item berdasarkan kesamaan perilaku antar pengguna.
3. *Hybrid methods*
   Menggabungkan kedua pendekatan di atas dalam membentuk hasil rekomendasi.

Sistem rekomendasi memegang peranan besar dalam membangun pengalaman pengguna yang personal, erat, dan kerap relevan. Sistem umumnya melakukan analisis terhadap fitur item dan perilaku pengguna untuk dapat menyarankan item baru yang tetap selaras, koheren, dan sesuai dengan preferensi pengguna.

B. *Greedy Algorithm*

Algoritma greedy merupakan metode yang populer dan sederhana untuk memecahkan persoalan optimasi. Algoritma ini umumnya digunakan untuk memecahkan persoalan optimasi (*optimization problems*), baik maksimasi (*maximization*) maupun minimasi (*minimization*) [2]. Algoritma greedy didefinisikan sebagai algoritma yang memecahkan persoalan secara langkah per langkah (*step by step*) sedemikian sehingga pada setiap langkah:

1. Diambil pilihan yang terbaik yang dapat diperoleh pada saat itu tanpa memperhatikan konsekuensi pengambilan pilihan untuk langkah ke depan (prinsip *taking what you can get now*)

2. Menganggap bahwa dengan dengan memiliki ‘optimum lokal’ pada setiap langkah, keseluruhan langkah akan berakhir dengan ‘optimum global’

Konsiderasi pemilihan pilihan terbaik didasarkan terhadap elemen-elemen yang dimiliki oleh algoritma *greedy* [2], meliputi:

1. Himpunan kandidat (*C*)
   Himpunan berisi kandidat yang akan dipilih pada setiap langkah.

2. Himpunan solusi (*S*)
   Himpunan berisi kandidat yang telah dipilih.

3. Fungsi Solusi
   Fungsi yang menentukan apakah himpunan kandidat (*C*) yang dipilih telah memberikan solusi terhadap persoalan.

4. Fungsi seleksi (*selection function*)
   Fungsi yang memilih kandidat berdasarkan strategi *greedy* tertentu sesuai dengan heuristik yang telah ditentukan sebelumnya.

5. Fungsi kelayakan (*feasible*)
   Fungsi yang memeriksa apakah kandidat yang dipilih dapat dimasukkan ke dalam himpunan solusi, apakah kandidat layak atau tidak.

6. Fungsi obyektif
   Fungsi yang menyatakan tujuan keseluruhan dari persoalan, yaitu memaksimumkan atau meminimumkan.

Menggunakan elemen-elemen tersebut, maka algoritma *greedy* dapat dinyatakan secara formal sebagai algoritma yang melibatkan pencarian sebuah himpunan bagian *S* dari himpunan kandidat *C*; yang dalam hal ini, *S* harus memenuhi beberapa kriteria yang ditentukan, yaitu *S* menyatakan suatu solusi dan dioptimasi oleh fungsi obyektif [2].

C. *Cosine Similarity*

Cosine similarity adalah pengukuran kemiripan (*similarity measure*) antara dua vektor non-nol menggunakan inner product. Secara geometris, *cosine similarity* diartikan sebagai nilai kosinus dari sudut antara dua vektor. Ukuran ini umum digunakan dalam melakukan analisis data, *text mining*, sistem rekomendasi, serta berbagai bidang *machine learning* lainnya untuk mengukur tingkat kesamaan antara dua entitas yang direpresentasikan sebagai vektor.

Untuk dua vektor, **A** dan **B**, yang masing-masing merepresentasikan suatu entitas dalam ruang berdimensi *n*. Kedua vektor dapat dinyatakan dalam bentuk

$$X = [X_1, X_2, \dots, X_n] \quad (1)$$

sehingga *cosine similarity* antara vektor **A** dan vektor **B**, dinyatakan dalam notasi *sim*(**A, B**), kemudian didefinisikan sebagai

$$sim(A,B) = \frac{A \cdot B}{\|A\|\|B\|} \quad (2)$$

dengan pembilang adalah produk skalar (*dot product*) dari vektor **A** dan vektor **B** yang dihitung sebagai jumlah perkalian elemen-elemen yang bersesuaian sesuai dimensi dari kedua vektor

$$A \cdot B = \sum_{i=1}^{n} A_i B_i = A_1 B_1 + A_2 B_2 + \cdots + A_n B_n \quad (3)$$

dan penyebut adalah panjang (*magnitudo*) Euclidean dari masing-masing vektor yang dihitung sebagai akar kuadrat dari jumlah kuadrat setiap elemennya.

$$\|X\| = \sqrt{\sum_{i=1}^{n} {X_i}^2} = \sqrt{{X_1}^2 + {X_2}^2 + \cdots + {X_n}^2} \quad (4)$$

Hasil nilai perhitungan *cosine similarity* akan berada dalam rentang [-1, 1]. Nilai semakin mendekati -1 menunjukkan bahwa kedua vektor memiliki arah yang berlawanan dan cenderung bertolak belakang atau memiliki perbedaan yang tinggi, sedangkan nilai semakin mendekati 1 menunjukkan bahwa kedua vektor bersifat searah dan cenderung mirip atau memiliki kesamaan yang tinggi. *Cosine similarity* juga dapat menunjukkan tidak adanya korelasi antara kedua vektor yang dianalisis, ditunjukkan oleh nilai mendekati 0, baik dari sisi negatif maupun positif.

Perhitungan menggunakan *cosine similarity* dalam menentukan tingkat kesamaan memberikan keunggulan, khususnya dalam menangani kondisi tidak diperhitungkannya panjang vektor dan jumlah dimensi yang tinggi. *Cosine similarity* hanya bergantung terhadap arah vektor dan tidak terhadap panjang vektor, sehingga dua buah item dengan arah yang berdekatan tetapi panjang berbeda akan tetapi memiliki nilai *cosine similarity* yang tinggi. Lalu, *cosine similarity* efektif dalam ruang vektor berdimensi tinggi yang umumnya menjadi kelemahan terhadap perbandingan menggunakan jarak Euclidean. Kedua hal tersebut mengakibatkan *cosine similarity* dapat secara efektif diterapkan secara luas di berbagai bidang untuk menyatakan tingkat kesamaan.

D. Fitur *Recommended Songs* pada Spotify

Salah satu fitur keunggulan Spotify sebagai aplikasi streaming musik dalam ranah global adalah *Recommended Songs*, yaitu sebuah sistem rekomendasi yang secara otomatis memberikan lagu-lagu baru yang dapat pengguna masukkan ke dalam playlist berdasarkan preferensi pengguna, interaksi pengguna dengan trek-trek tertentu, serta pola pendengaran jutaan penggunaan lainnya [4].

Fitur ini dirancang untuk memperkaya pengalaman pengguna dalam mendengarkan musik dengan menghadirkan konten yang personal, memperkenalkan pengguna kepada artis atau genre baru, sekaligus menjaga relevansi dan keberlanjutan dari interaksi pengguna dengan *platform* [4].

III. Analisis

Memanfaatkan natur dari *greedy algorithm* yang umum digunakan dalam permasalahan optimasi karena implementasinya yang simpel dalam mengambil keputusan lokal terbaik secara bertahap, *greedy algorithm* dapat digunakan dalam konteks sistem rekomendasi *Recommended Songs* pada Spotify untuk memilih kandidat lagu yang paling relevan terhadap lagu-lagu dalam sebuah *playlist*. Sebagai elemen fungsi seleksi dari *greedy algorithm*, dapat dilakukan perhitungan menggunakan *cosine similarity* untuk menyatakan tingkat kesamaan atau kedekatan antar lagu dalam memilih kandidat lagu untuk dimasukkan ke dalam solusi. Maka, untuk melakukan hal tersebut, setiap lagu harus dapat direpresentasikan sebagai sebuah vektor dengan banyak dimensi terlebih dahulu.

Spotify menyatakan sejumlah fitur audio numerik yang disediakan melalui Spotify Audio Features API [3] untuk setiap lagu, dengan setiap nilai telah dinormalisasi untuk berada dalam rentang [0, 1] [3]. Merujuk kepada *website* tersebut, beberapa fitur audio yang dapat dinyatakan dengan baik sebagai dasar perhitungan kemiripan adalah.

1. *Acousticness* (tingkat akustik)
   Ukuran pengukuran terhadap sifat akustik dari sebuah trek. Nilai semakin tinggi menunjukkan trek lagu yang lebih bersifat akustik.

2. *Danceability* (tingkat dansa)
   Ukuran pengukuran terhadap seberapa cocok sebuah lagu untuk menari berdasarkan kombinasi elemen musik, meliputi tempo, stabilitas ritme, kekuatan ketukan, dan keteraturan keseluruhan. Nilai semakin tinggi menunjukkan trek lagu semakin cocok untuk ditarikan.

3. *Energy* (tingkat energi)
   Ukuran pengukuran terhadap persepsi intensitas dan aktivitas. Umumnya trek yang energik terasa cepat, keras, dan berisik. Misalnya, trek dengan genre *death metal* memiliki energi yang tinggi, sementara *prelude* Bach memiliki nilai energi yang rendah. Nilai ini ditentukan oleh banyak faktor, meliputi kedinamisan rentang lagu, kenyaringan, timbre, dan entropi umum dari lagu.

4. *Instrumentalness* (tingkat instrumental)
   Ukuran pengukuran terhadap keberadaan vokal, termasuk *adlibs* seperti ‘ooh’ atau ‘aah’. Nilai semakin tinggi menunjukkan trek lagu lebih tidak memiliki konten vokal, dengan nilai lebih dari 0.5

sudah cukup untuk mengindikasikan bahwa trek merupakan trek instrumental.

5. *Liveness* (tingkat kehidupan)
   Ukuran pengukuran terhadap keberadaan penonton dalam trek. Nilai semakin tinggi menunjukkan kemungkinan yang lebih besar bahwa lagu tersebut dibawakan secara langsung, dengan nilai lebih dari 0.8 sudah cukup untuk mengindikasikan bahwa trek merupakan rekaman langsung dari sebuah lagu.

6. *Speechiness* (tingkat ucapan)
   Ukuran pengukuran terhadap keberadaan kata-kata yang diucapkan dalam sebuah trek. Semakin eksklusif rekaman tersebut terhadap sebuah ucapan (seperti *talkshow* atau buku audio) akan memberikan nilai semakin tinggi. Nilai di atas 0.66 cukup untuk mengindikasikan trek sepenuhnya terdiri atas pengucapan, nilai di antara 0.33 dan 0.66 mengindikasikan trek yang merupakan campuran musik dan ucapan, dan nilai di bawah 0.33 cukup untuk mengindikasikan trek sepenuhnya berupa music

7. *Valence* (tingkat valensi)
   Ukuran pengukuran terhadap kepositifan musikal yang disampaikan oleh sebuah lagu. Lagu dengan valensi tinggi terdengar lebih positif (gembira, ceria, euforia), sedangkan lagu dengan valensi rendah terdengar lebih negatif (sedih, tertekan, marah).

Dengan demikian, sebuah lagu dapat direpresentasikan menjadi sebuah vektor dengan 7 dimensi. Sebagai contoh, vektor yang dapat digunakan perhitungan adalah

$$V = [0.33, 0.14, 0.2, 0.56, 0.12, 0.34, 0.98] \quad (5)$$

yang merepresentasikan sebuah lagu dengan atribut audio sebagai berikut

TABLE I. ATRIBUT AUDIO

Atribut Audio	***Nilai***
Acousticness	0.33
Danceability	0.14
Energy	0.2
Instrumentalness	0.56
Liveness	0.12
Speechiness	0.34
Valence	0.98

Nilai-nilai atribut audio dapat digunakan sebagai dasar perhitungan *cosine similarity* untuk menentukan tingkat kesamaan keseluruhan antara dua buah lagu. Perhitungan dapat memperhitungkan semua lagu yang telah berada di dalam *playlist* untuk memastikan koherensi antar lagu dalam hasil akhir *playlist* tetap terjaga, sehingga perhitungan nilai didapatkan dengan

$$score(c) = \frac{1}{n}\sum_{i=1}^{n} sim(c, s_i) \quad (6)$$

dengan c adalah kandidat lagu dan $s_1, s_2, \ldots, s_n$ adalah lagu-lagu yang telah berada di dalam *playlist*.

IV. STUDI KASUS

Berdasarkan landasan teori dan pembahasan yang telah disampaikan mengenai pendekatan *greedy algorithm* menggunakan *cosine similarity* untuk sistem rekomendasi *Recommended Songs*, dapat dibentuk suatu pemodelan implementasi *greedy algorithm* yang menggambarkan bagaimana cara kerja dan kinerja sistem rekomendasi dalam memberikan kandidat lagu untuk dimasukkan ke dalam *playlist* berdasarkan tingkat kesamaannya dengan lagu-lagu di dalam *playlist*.

1. Implementasi vektor sebagai representasi lagu

Implementasi vektor pada Python digunakan menggunakan *library* NumPy untuk membentuk *array* yang terdefinisi secara konkrit. Sebagai contoh studi kasus, dibentuk 20 lagu sebagai yang dapat dipilih oleh pengguna untuk dimasukkan ke dalam *playlist* sebagai dasar perhitungan *cosine similarity*, serta 80 lagu sebagai *pool* kandidat lagu yang akan dipilih oleh sistem untuk direkomendasikan berdasarkan hasil perhitungan tingkat kesamaan.

```
# Buat 100 vector lagu dengan fitur random (rentang [0, 1])
np.random.seed(42)
song_vectors = np.round(np.random.rand(100, 7), 4)
song_titles = [f"Lagu #{i+1}" for i in range(len(song_vectors))]
```

Gambar 2. Inisiasi 100 vektor sebagai representasi 100 lagu
Sumber: dokumentasi pribadi

2. Aplikasi fungsi selektif sebagai elemen *greedy algorithm*

Proses perhitungan akan dilakukan dengan menghitung *cosine similarity* terhadap kandidat lagu dengan keseluruhan lagu yang telah dimasukkan ke dalam *playlist*. Proses ini diiterasi sebanyak enam kali sesuai dengan tampilan *Recommended Songs* pada Spotify yang menampilkan enam kandidat lagu kepada pengguna. Hasil dari setiap iterasi disimpan di dalam *array recommended_indices* yang kemudian diabaikan di iterasi berikutnya.

```
while True:
    recommended_indices = []
    for step in range(6):
        best_score = -1
        best_index = -1

        for i in range(len(song_vectors)):
            if any(i == rec[0] for rec in recommended_indices) or i in playlist_index:
                continue
            avg = np.mean(cosine_similarity([song_vectors[i]], playlist)[0])
            if avg > best_score:
                best_score = avg
                best_index = i

        recommended_indices.append([best_index, best_score])
```

Gambar 3. Tahapan pengaplikasian fungsi seleksi dalam memilih komponen solusi optimal
Sumber: dokumentasi pribadi

3. Aplikasi fungsi selektif sebagai elemen *greedy algorithm*

Setelah pengguna menerima kandidat lagu yang direkomendasikan sistem untuk dimasukkan ke dalam *playlist*, pengguna kemudian dapat memilih untuk memasukkan lagu-lagu tersebut ke dalam *playlist*. Sistem kemudian akan melakukan lagi keseluruhan algoritma dan kembali memberikan enam rekomendasi kandidat lagu sesuai dengan keadaan *playlist* saat itu.

Keseluruhan program kemudian dijalankan dengan hasil akhir sebagai berikut.

```
Masukkan jumlah lagu yang akan dimasukkan ke playlist awal (1-20): 3
Masukkan 3 indeks lagu (pisahkan dengan spasi) (1-20): 2 6 18

=== LAGU DALAM PLAYLIST ===
1. Lagu  #2: [0.8662 0.6011 0.7081 0.0206 0.9699 0.8324 0.2123]
2. Lagu  #6: [0.8084 0.3046 0.0977 0.6842 0.4402 0.122  0.4952]
3. Lagu #18: [0.5393 0.8074 0.8961 0.318  0.1101 0.2279 0.4271]

=== 6 LAGU YANG DIREKOMENDASIKAN ===
1. Lagu #14: [0.7132 0.7608 0.5613 0.771  0.4938 0.5227 0.4275]
   Rata-rata similarity: 0.8691

2. Lagu #52: [0.5858 0.9402 0.5755 0.3882 0.6433 0.4583 0.5456]
   Rata-rata similarity: 0.8654

3. Lagu #32: [0.7465 0.6496 0.8492 0.6576 0.5683 0.0937 0.3677]
   Rata-rata similarity: 0.8581

4. Lagu #31: [0.5487 0.6919 0.652  0.2243 0.7122 0.2372 0.3254]
   Rata-rata similarity: 0.8575

5. Lagu #77: [0.9966 0.5554 0.769  0.9448 0.8496 0.2473 0.4505]
   Rata-rata similarity: 0.8559

6. Lagu #17: [0.9297 0.8081 0.6334 0.8715 0.8037 0.1866 0.8926]
   Rata-rata similarity: 0.8558


Masukkan indeks (1-6) lagu yang ingin ditambahkan ke playlist (0 untuk selesai):
```

Gambar 4. Pengguna memasukkan *playlist* dan mendapatkan rekomendasi kandidat lagu
Sumber: dokumentasi pribadi

```
=== PLAYLIST AKHIR ===
1. Lagu  #2
2. Lagu  #6
3. Lagu #18
 --- recommended songs ---
4. Lagu #14
5. Lagu #52
6. Lagu #27
7. Lagu #17
8. Lagu #40

Rata-rata playlist awal : [0.738  0.571  0.5673 0.3409 0.5067 0.3941 0.3782]
Rata-rata playlist akhir: [0.7851 0.7013 0.5289 0.564  0.616  0.4509 0.5407]
Perbedaan rata-rata: [ 0.0471  0.1303 -0.0384  0.2231  0.1093  0.0568  0.1625]
```

Gambar 5. Pengguna memasukkan rekomendasi lagu dengan hasil *playlist* tetap koheren
Sumber: dokumentasi pribadi

Dapat dilihat bahwa menggunakan *greedy algorithm*, sistem rekomendasi *Recommended Songs* memberikan rekomendasi kandidat lagu yang koheren dan selaras dengan lagu-lagu dalam *playlist*. Sehingga, setelah pengguna memasukkan beberapa rekomendasi tersebut, nilai rata-rata dari atribut audio untuk *playlist* tidak mengalami perubahan yang signifikan, sehingga tetap menghasilkan *playlist* yang dekat dan akrab dengan preferensi dan selera musik pengguna.

V. Analisis

Melalui studi kasus implementasi sistem rekomendasi *Recommended Songs* berbasis pendekatan *greedy algorithm* menggunakan *cosine similarity*, dapat dilakukan pengkajian terhadap kelebihan dan kekurangan dari penggunaan *greedy algorithm* dalam membentuk suatu sistem rekomendasi.

A. Keunggulan

Implementasi *greedy algorithm* dalam sistem rekomendasi *Recommended Songs* menawarkan sejumlah keunggulan. Pertama, strategi ini efisien secara komputasi, sehingga cocok digunakan untuk pemrosesan cepat dan interaktif, terutama dalam sistem yang membutuhkan respons *real-time* secara langsung seperti Spotify. Selain itu, pendekatan *greedy* dapat dengan mudah diimplementasikan menggunakan fungsi seleksi sederhana seperti *cosine similarity* yang digunakan dalam studi kasus sebelumnya. Penggunaan fungsi seleksi lainnya juga dapat dengan mudah digunakan karena dapat langsung memanfaatkan data dari API resmi Spotify dan *library* analisis perhitungan seperti NumPy atau scikit-learn tanpa desain arsitektur model yang kompleks.

Pendekatan menggunakan *greedy algorithm* turut memberikan keunggulan dalam hal relevansi dengan konteks lokal *playlist* yang dibangun. Proses rekomendasi menggunakan *greedy algorithm* mempertimbangkan tingkat kesamaan langsung antara kandidat lagu dengan lagu-lagu yang telah berada di dalam *playlist*, bukan berdasarkan faktor-faktor eksternal seperti profil pengguna secara keseluruhan. Dengan demikian, sistem rekomendasi dapat menghasilkan saran yang lebih kontekstual dan dapat dipastikan koheren dengan suasana atau tema dari *playlist* tertentu.

B. Keterbatasan

Namun, penggunaan *greedy algorithm* sebagai pendekatan dalam membentuk sistem rekomendasi turut memiliki sejumlah keterbatasan. Salah satu kelemahan utama adalah kurangnya keberagaman (*diversity*) dalam hasil rekomendasi kandidat lagu. Lagu-lagu dengan tingkat kesamaan tinggi akan cenderung lebih dipilih oleh sistem, sehingga potensi eksplorasi terhadap variasi lagu menjadi rendah, yang mungkin menjadi pertimbangan dalam memberikan rekomendasi bagi pengguna.

Selain itu, *greedy algorithm* tidak mempertimbangkan konteks temporal, yaitu urutan waktu penambahan lagu ke

dalam *playlist*. Sistem rekomendasi akan memperhitungkan seluruh lagu dalam sebuah *playlist*, termasuk lagu yang telah dimasukkan bertahun-tahun sebelumnya, dalam membentuk rekomendasi kandidat lagu. Walaupun hal ini tidak sepenuhnya keliru, umumnya pengguna hanya menginginkan lagu berdasarkan preferensi dan selera musik belakangan ini. Sistem ini juga rentan terhadap bias konten, terutama jika *playlist* hanya terdiri dari lagu-lagu yang sangat homogen. Hal ini dapat menyebabkan hasil rekomendasi menjadi monoton dan kurang menarik dalam jangka panjang.

VI. Pengembangan Lebih Lanjut

Untuk menangani keterbatasan yang dimiliki oleh pendekatan *greedy algorithm*, dapat digunakan dimensi tambahan sebagai faktor dalam perhitungan tingkat kesamaan yang dapat dipertimbangkan dalam sistem rekomendasi: *diversity, temporal dynamics,* dan *content bias mitigation* [5][6][7].

1. *Diversity-aware*

Sistem rekomendasi dengan dimensi *diversity-aware* menggunakan dimensi *diversity* untuk tidak hanya memberikan rekomendasi yang relevan sesuai dengan preferensi pengguna, tetapi turut memberikan konsiderasi terhadap keberagaman hasil rekomendasi agar tidak monoton dan menjenuhkan bagi pengguna. Eksistensi dari dimensi *diversity* mendorong sistem untuk merekomendasikan lagu-lagu yang bervariasi, bahkan jika tingkat kemiripannya sedikit lebih rendah, untuk memberikan pengalaman mendengarkan yang lebih eksploratif [5].

```
sim_to_playlist = similarities[idx]
sim_to_selected = 0

if selected:
    sim_to_selected = np.mean(
        [cosine_similarity([candidates[idx]], [candidates[j]])[0][0] for j in selected]
    )

# Penalti jika terlalu mirip dengan kandidat yang sudah dipilih sesuai nilai alpha
score = sim_to_playlist - alpha * sim_to_selected
```

Gambar 6. Implementasi dimensi *diversity* pada pendekatan *diversity-aware*
Sumber: dokumentasi pribadi

Nilai alpha digunakan untuk memberikan konsiderasi terhadap kemiripan kandidat lagu yang direkomendasikan dengan lagu-lagu yang telah dipilih untuk direkomendasikan. Pengurangan nilai tersebut memberikan hasil rekomendasi yang lebih *diverse*, sehingga semakin besar nilai alpha, maka perhitungan nilai akhir akan lebih mengutamakan diversitas dengan semakin mengabaikan opsi yang terlalu homogen.

2. *Time-aware*

Sistem rekomendasi dengan dimensi *time-aware* mempertimbangkan dimensi waktu sebagai bagian dari konteks dalam proses rekomendasi. Hal ini melakukan konsiderasi terhadap preferensi pengguna yang bersifat dinamis dan kecenderungan pemutaran lagu di waktu tertentu untuk memastikan rekomendasi kontekstual dengan situasi dan tidak bersifat stagnan [6].

```
# Fungsi scoring dengan time-aware
def score_with_time(candidate, playlist, last_played_time, alpha=0.5):
    sim = np.mean(cosine_similarity([candidate], playlist))
    days_ago = (now - last_played_time).days
    time_score = 1 / (1 + days_ago)  # semakin lama, semakin kecil
    final_score = alpha * sim + (1 - alpha) * time_score
    return final_score, sim, time_score
```

Gambar 7. Implementasi dimensi *temporal dynamics* pada pendekatan *time-aware*
Sumber: dokumentasi pribadi

Untuk implementasi dari pendekatan dengan dimensi *time-aware*, dibentuk fungsi *score_with_time* yang memperhitungkan kapan terakhir kali suatu lagu terakhir dimainkan, dengan *heuristik* bahwa pengguna sedang menggemari lagu yang baru-baru saja dimainkan. Sama seperti *diversity-aware*, digunakan nilai alpha sebagai besaran konsiderasi kontekstual temporal, dengan semakin besar nilai alpha, maka perhitungan akan lebih mengutamakan lagu-lagu yang baru saja dimainkan oleh pengguna.

3. *Cluster-based*

Cluster-based merupakan dimensi yang memetakan lagu-lagu ke dalam kelompok berdasarkan kemiripan fitur. Pengelompokan ini dapat dilakukan ke dalam beberapa klaster menggunakan teknik *unsupervised learning* seperti K-Means. Setelah klaster terbentuk, sistem hanya merekomendasikan lagu dari klaster yang paling mirip dengan konten dalam *playlist* pengguna. Dengan demikian, sistem rekomendasi lebih efisien dalam menentukan rekomendasi kandidat lagu, menjaga konsistensi genre, dan menghindari perbandingan tanpa heuristik terlebih dahulu yang umumnya tidak efisien [7].

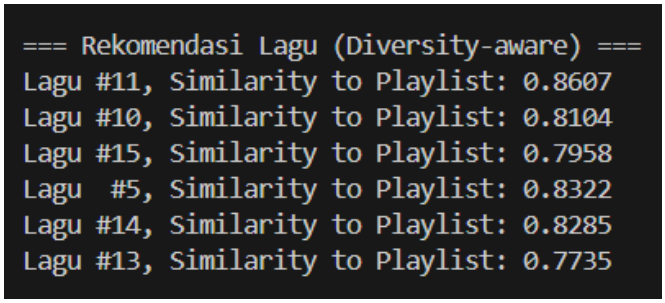
```
# Filter kandidat dari klaster dominan
filtered_candidates = []
filtered_indices = []
for i in range(len(candidates)):
    cluster_idx = cluster_labels[len(playlist) + i]
    if cluster_idx == dominant_cluster:
        filtered_candidates.append(candidates[i])
        filtered_indices.append(i)
```

Gambar 8. Implementasi dimensi *content bias mitigation* pada pendekatan *cluster-based*
Sumber: dokumentasi pribadi

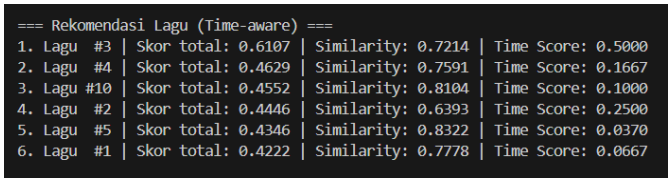
Pencarian lagu untuk dijadikan sebagai rekomendasi kandidat diutamakan dari klaster dominan, yaitu klaster yang dibentuk berdasarkan lagu-lagu yang telah berada di dalam *playlist*. Dengan demikian, rekomendasi akhir disaring dengan skor *similarity* tertinggi terhadap klaster dominan untuk menghasilkan rekomendasi lagu dengan keseragaman karakteristik.

Ketiga dimensi tersebut memiliki kelebihan dan kekurangannya masing-masing dalam perannya sebagai tambahan konsiderasi untuk menghasilkan rekomendasi yang lebih kontekstual dan eksploratif dibandingkan menggunakan pendekatan *greedy algorithm* saja. Hasil dari masing-masing

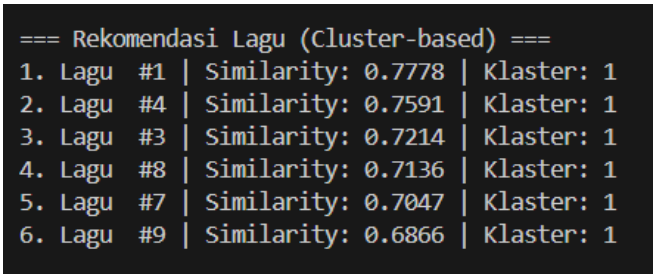
dimensi terhadap studi kasus yang telah dilakukan sebelumnya adalah sebagai berikut.



Gambar 9. Hasil pendekatan *diversity-aware* terhadap studi kasus
Sumber: dokumentasi pribadi



Gambar 10. Hasil pendekatan *time-aware* terhadap studi kasus
Sumber: dokumentasi pribadi



Gambar 11. Hasil pendekatan *cluster-based* terhadap studi kasus
Sumber: dokumentasi pribadi

Berdasarkan hasil dari implementasi ketiga dimensi untuk studi kasus sebelumnya, dapat dilihat bahwa ketiga dimensi memberikan hasil rekomendasi yang berbeda-beda. Oleh karena itu, pemilihan dimensi dalam implementasi aktual dari sistem rekomendasi perlu disesuaikan dengan konteks dan tujuan spesifik dari aplikasi yang digunakan. Tidak dapat ditentukan secara mutlak dimensi mana, atau bahkan kombinasi dimensi, yang lebih optimal untuk digunakan karena efektivitas masing-masing pendekatan sangat bergantung terhadap kebutuhan pengguna, karakteristik data, serta prioritas sistem terhadap keberagaman data, konteks temporal, atau konsistensi pengelompokan karakteristik data.

VII. Kesimpulan

Spotify sebagai salah satu *platform streaming* musik terbesar secara global menawarkan berbagai fitur berbasis sistem rekomendasi, salah satunya adalah *Recommended Songs* yang menyarankan lagu-lagu untuk ditambahkan ke sebuah *playlist* berdasarkan isi lagu-lagu di dalamnya. Makalah ini menjelaskan bagaimana sebuah sistem rekomendasi dapat menggunakan pendekatan *greedy algorithm* untuk memilih lagu-lagu yang memiliki tingkat kemiripan tinggi dengan seluruh lagu yang berada di dalam *playlist*. Penggunaan *greedy algorithm* menggunakan representasi vektor dari atribut yang diterapkan oleh Spotify kepada setiap trek audio, seperti *acousticness*, *energy*, dan *valence*, yang memungkinkan perhitungan *cosine similarity* untuk menentukan tingkat kesamaan antar lagu, yang kemudian dijadikan dasar pemilihan kandidat terbaik dalam setiap iterasi.

Simulasi berbasis data buatan menunjukkan bahwa *greedy algorithm* mampu menghasilkan hasil rekomendasi yang relevan dan cepat, sekaligus menjaga koherensi karakteristik lagu-lagu yang disarankan untuk tetap koheren dengan *playlist*. Meskipun pendekatan menggunakan *greedy algorithm* efektif dalam menyesuaikan konteks lokal dari *playlist*, terdapat keterbatasan dalam hal keberagaman dan konteks semantik dari *playlist*. Maka, makalah melakukan perluasan dari pendekatan tersebut dengan melibatkan tiga dimensi untuk meningkatkan performa sistem rekomendasi: *diversity-aware*, *time-aware*, dan *cluster-based*. Ketiga dimensi memberikan dampak bagi algoritma pemilihan rekomendasi kandidat lagu untuk memberikan hasil rekomendasi yang lebih variatif, kontekstual, dan berorientasi pada pengalaman pengguna.

Secara keseluruhan, *greedy algorithm* merupakan pendekatan yang layak dan efektif dalam implementasi dasar sistem rekomendasi *Recommended Songs* milik Spotify, tetapi sangat terbuka untuk pengembangan lebih lanjut dengan memperhatikan kebutuhan dan preferensi pengguna yang beragam dan dinamis. Integrasi dimensi tambahan secara tepat dan kontekstual dapat meningkatkan kualitas rekomendasi kandidat lagu hasil sistem rekomendasi, untuk memberikan pengalaman menggunakan aplikasi *platform streaming* musik yang lebih adaptif dan personal.

VIII. Ucapan Terima Kasih

Penulis memanjatkan puji syukur kepada Tuhan yang Maha Esa atas berkat dan rahmat-Nya yang melimpah dalam proses penulisan makalah sehingga makalah dapat diselesaikan dengan baik dan tepat waktu. Penulis juga mengucapkan terima kasih yang sebesar-besarnya kepada Ibu Dr. Nur Ulfa Maulidevi, S.T, M.Sc. selaku dosen pengampu mata kuliah IF2211 Strategi Algoritma kelas K01 atas waktu, bimbingan, dan ilmu pengetahuan yang telah diberikan sebagai bekal dalam pembentukan makalah. Akhir kata, penulis menyampaikan dukungan semangat kepada teman-teman penulis yang sedang berjuang bersama-sama untuk menyelesaikan masa studi semester genap tahun ajaran 2024/2025.

IX. Lampiran

Source code program yang dibentuk sebagai pendukung bagian studi kasus dan analisis dari makalah dapat diakses melalui tautan berikut:

https://github.com/rafenmaxxx/RecommendedSongs.git

References

[1] F. O. Isinkaye, Y. O. Folajimi, dan B. A. Ojokoh, "Recommendation systems: Principles, methods and evaluation," *Egyptian Informatics Journal*, vol. 16, no. 3, pp. 261-273, Nov. 2015.

[2] Munir, Rinaldi. "Homepage Rinaldi Munir". https://informatika.stei.itb.ac.id/~rinaldi.munir. [Diakses pada 24 Juni 2025].

[3] Spotify for Developers, "Get Audio Features," *Spotify Web API Reference*. [Daring]. Tersedia: https://developer.spotify.com/documentation/web-api/reference/get-audio-features. [Diakses: 24 Juni 2025].

[4] C. Maheshwari, "Music recommendation on spotify using deep learning," *arXiv preprint arXiv:2312.10079*, 2023.

[5] Y. Wang, Y. Gao, M. Li, dan J. Zhang, "Result Diversification in Search and Recommendation: A Survey," arXiv preprint arXiv:2212.14464, 2022. [Daring]. Tersedia: https://arxiv.org/abs/2212.14464

[6] A. G. M. Shahrul dan S. Pradana, "Sistem Rekomendasi dengan Time-Aware menggunakan Teknik Pembobotan Naive Bayes Classifier," dalam Proceedings of the 3rd Conference on Computer and Applications (3CA), Atlantis Press, 2013. [Daring]. Tersedia: https://www.atlantis-press.com/proceedings/3ca-13/10182

[7] D. Fan, L. Nie, L. Yang, dan T.-S. Chua, "Learning Heterogeneous Temporal Patterns of User Preference for Timely Recommendation," arXiv preprint arXiv:2109.12839, 2021. [Daring]. Tersedia: https://arxiv.org/abs/2109.12839

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, 24 Juni 2025

Rafen Max Alessandro
13523031