Automatic Playlist Continuation Menggunakan Hybrid Recommender System

Martin A. Linggajaya, Henry N. Palit, Alvin N. Tjondrowiguno
Program Studi Informatika Fakultas Teknologi Industri Universitas Kristen Petra
Jl. Siwalankerto 121 – 131 Surabaya 60236
Telp. (031) – 2983455, Fax. (031) - 8417658
E-mail: martin.alinggajaya@gmail.com, hnpalit@petra.ac.id, alvin.nathaniel@petra.ac.id

ABSTRAK

Salah satu cara terpopuler untuk mendengarkan musik adalah dengan membuat *playlist*. Fitur *playlist* bisa dikembangkan dengan memberikan rekomendasi lagu untuk ditambahkan ke dalam *playlist*. Untuk mengembangkan proses rekomendasi ini, ACM dan Spotify mengadakan RecSys Challenge 2018 untuk *automatic playlist continuation*.

Penelitian ini adalah pengembangan dari [6] yang menempati juara ketiga dalam RecSys Challenge 2018. Metode yang digunakan terdiri dari dua tahap, yaitu seleksi kandidat menggunakan sebuah *hybrid recommender system* bernama LightFM dan *ranking* menggunakan XGBoost. Pengembangan yang dilakukan berfokus kepada salah satu perhitungan yang digunakan dalam *ranking*, yaitu perhitungan fitur *co-occurrence.*

Hasil penelitian ini menunjukkan bahwa *co-occurrence* antara 3 *track* tidak meningkatkan performa model. Dari percobaan yang dilakukan, model oleh [6] mendapatkan nilai 0,5251, 0,5582, dan 1,295 untuk R-precision, NDCG, dan *recommended song clicks*. Hasil model di penelitian ini mendapatkan nilai 0,5241, 0,5579, dan 1,312 untuk R-precision, NDCG, dan *recommended song clicks*.

Kata Kunci: sistem rekomendasi *hybrid*, LightFM, XGBoost, *automatic playlist continuation.*

ABSTRACT

One of the most popular ways to listen to music is using playlists. The playlist feature can be improved by giving track recommendations to be added to certain playlists. To support the development of this recommendation process, ACM and Spotify held the RecSys Challenge 2018 with the task of automatic playlist continuation.

This research is a continuation from [6] that placed 3rd in the RecSys Challenge 2018. The method used consists of 2 phases: candidate selection using a hybrid recommender system called LightFM and ranking using XGBoost. The research gap being developed focuses on one of the calculations for co-occurrence features used in the ranking phase.

The result of this research shows that co-occurrence of 3 tracks does not improve the performance of the model used. The model by [6] achieved scores of 0.5251, 0.5582, and 1.295 for R-precision, NDCG, and recommended song clicks respectively. Meanwhile, the model produced in this research achieved an R-precision of 0.5241, an NDCG of 0.5579, and recommend song clicks of 1.312.

Keywords: *hybrid recommender system, LightFM, XGBoost, automatic playlist contination.*

1. PENDAHULUAN

Streaming musik adalah cara yang populer untuk mendengarkan musik dan menjadi sumber lebih dari 50% pemasukkan di seluruh industri musik [5]. Banyak layanan *streaming* musik seperti Spotify menyediakan fitur di mana pengguna dapat menyeleksi dan menyimpan lagu-lagu mereka ke dalam *playlist*. Fitur *playlist* bisa dikembangkan dengan memberikan rekomendasi lagu atau *track* untuk ditambahkan ke dalam *playlist*. Proses rekomendasi ini disebut sebagai *automatic playlist continuation*. *Automatic playlist continuation* yang baik dapat meningkatkan durasi *streaming* per *session* dan meningkatkan tingkat *engagement* antara pengguna dan layanan *streaming* [1].

Untuk mendukung penelitian ini, ACM Recommender Systems (RecSys) Challenge 2018 memberikan *task* berupa *automatic playlist continuation*. Dataset yang digunakan adalah *Million Playlist Dataset* dari Spotify yang berisi 1 juta *playlist* beserta dengan daftar lagu-lagu di dalamnya [7]. Penelitian ini diinspirasi dari ACM RecSys Challenge 2018, yaitu dengan menggunakan *dataset*, tolak ukur, dan format *submission* yang sama dengan ACM RecSys Challenge 2018.

Metode yang diajukan di penelitian ini terinspirasi dari [6] yang menempati juara ketiga di RecSys Challenge 2018. Metode ini terdiri dari 2 tahap. Tahap pertama menggunakan *hybrid recommender system* untuk menyeleksi kandidat lagu yang akan direkomendasikan. Algoritma yang digunakan adalah LightFM, sebuah metode gabungan dari *collaborative filtering* dan *content-based filtering* yang dikebangkan oleh [4]. LightFM menghasilkan rekomendasi yang lebih relevan dibandingkan dengan *collaborative filtering* atau *content-based filtering* pada umumnya. Tahap kedua adalah *feature engineering* dari model LightFM dan dari perhitungan *co-occurrence*, dan mengurutkan *(ranking)* kandidat lagu menggunakan XGBoost – sebuah variasi algoritma *gradient boosting* yang dikembangkan oleh T. Chen & Guestrin [2]. XGBoost memiliki performa yang sangat baik dan adalah salah satu model *machine learning* yang paling banyak digunakan untuk memenangkan perlombaan *data science.*

Perbedaan yang dilakukan dalam penelitian ini adalah perhitungan fitur *normalized co-occurrence* yang akan digunakan dalam *training* model XGBoost. Dari penelitian yang dilakukan oleh Rubtsov et al., fitur ini adalah fitur yang paling berpengaruh pada hasil rekomendasi final dari model XGBoost [6]. Fitur *normalized co-occurrence* yang dilakukan oleh Rubtsov et al. menggunakan *co-occurrence* antara 2 *track* dibagi dengan frekuensi *track* kandidat. Jika perhitungan ini dilihat sebagai *conditional probability*, maka perhitungan ini berimplikasi bahwa 2 *track* yang dicari *co-occurrence*-nya independen dari *track-track* lain dalam *playlist* tersebut. Maka, *normalized co-occurrence* yang dilakukan di penelitian ini menggunakan *co-occurrence* antara 3 *track* dibagi dengan *co-occurrence* antara 2 *track*. Hal ini bertujuan untuk

mengurangi implikasi *independence* terhadap perhitungan fitur *normalized co-ocurrence*.

2. DASAR TEORI

2.1 LightFM

LightFM merepresentasikan *user* dan *item* sebagai *latent factors* atau *embeddings* sama seperti pada *collaborative filtering*. Namun, pada kasus di mana tidak ada interaksi antara suatu *user* dengan *item* tertentu, LightFM membuat prediksi berdasarkan informasi mengenai *user* dan *item* tersebut seperti pada content-based filtering. Setiap *user* digambarkan sebagai sebuah set fitur-fitur yang dimiliki oleh *user* tersebut. Setiap *item* juga digambarkan sebagai sebuah set fitur-fitur yang dimiliki oleh *item*. Hasilnya, *latent factor* atau *embedding* yang dihasilkan untuk suatu *user* adalah penjumlahan dari *latent factor* fitur-fitur *user* tersebut, begitu juga dengan item. Bias yang dihasilkan untuk sebuah *user* atau *item* juga adalah penjumlahan dari bias yang dihasilkan untuk setiap fitur yang dimiliki *user* atau *item* tersebut.

LightFM menghasilkan skor prediksi antara suatu user dengan suatu item yang dijabarkan dalam rumus berikut:

$$\hat{r}_{ui} = f(q_u \cdot p_i + b_u + b_i) \qquad (1)$$

di mana q_u adalah penjumlahan dari fitur *latent vectors* milik suatu *user*, p_i adalah penjumlahan dari fitur *latent vectors* milik suatu *item*, b_u adalah penjumlahan dari bias fitur suatu *user*, dan b_i adalah penjumlahan dari bias fitur suatu *item*.

2.2 XGBoost

XGBoost adalah implementasi khusus dari algoritma *gradient boosting,* metode *machine learning* yang menggunakan *ensemble boosting* dari sejumlah *decision tree*. Algoritma XGBoost memiliki beberapa kelebihan dari algoritma *gradient boosting* pada umumnya. XGBoost menggunakan L1 dan L2 *regularization* untuk menghindari *overfitting*. XGBoost juga dapat melakukan kalkulasi terhadap data *sparse* dengan baik, karena XGBoost akan secara otomatis melakukan *traversal* terhadap data yang tidak kosong. XGBoost juga bersifat *greedy* karena XGBoost membatasi kompleksitas *decision tree*. *Library* XGBoost juga dapat menggunakan *resource* dengan efisien, contohnya kemampuan XGBoost untuk dijalankan secara paralel dan juga dalam sebuah *cluster*.

3. DATA

Karena keterbatasan *resource* yang tersedia, maka data yang digunakan berjumlah 20% dari data lengkap dalam *Million Playlist Dataset*. Data yang digunakan terdiri dari 200.000 *playlist* dengan total *track* sekitar 13 juta. Informasi mengenai *playlist* termasuk judul, daftar *track* yang terkandung di dalamnya, durasi *playlist*, dan jumlah *track* yang hilang dari *playlist*. Informasi mengenai *track* adalah judul, artis, album, dan durasi *track*. Dataset juga dilengkapi dengan *challenge set* yang berisi 10.000 *playlist* dibagi ke dalam 10 skenario atau kategori. Skenario-skenario tersebut adalah (1) judul *playlist* saja, tanpa lagu, (2) judul *playlist* dan 5 lagu pertama, (3) lima lagu pertama, (4) judul *playlist* dan 5 lagu pertama, (5) sepuluh lagu pertama (6) judul *playlist* dan 25 lagu pertama, (7) judul *playlist* dan 25 lagu *random*, (8) judul *playlist* dan 100 lagu pertama, (9) judul *playlist* dan 100 lagu *random*, (10) judul *playlist* dan lagu pertama. Hasil rekomendasi akhir yang dibuat adalah 500 *track* bagi masing-masing *playlist* dalam *test set* yang diurutkan berdasarkan relevansi kepada *playlist*.

4. DESAIN SISTEM

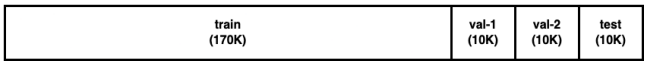
Bagian ini menjelaskan metode yang digunakan.

4.1 Persiapan Data

Data dipersiapkan dengan mengubah bentuk data dari *file* JSON menjadi *dataframe*. Data dimasukkan ke dalam beberapa *dataframe* yaitu *df_tracks* untuk metadata mengenai masing-masing *track*, *df_playlists_metadata* untuk metadata mengenai setiap *playlist*, *df_playlists* untuk data interaksi antara *track* dengan *playlist*, *df_challenge_playlists_metadata* untuk metadata mengenai masing-masing *playlist* dalam set *challenge*, dan *df_challenge_playlists* untuk data interaksi antara *track* dengan *challenge playlist*.

4.2 Strategi Validasi

Playlist yang disediakan dalam *Million Playlist Dataset* dibagi menjadi empat set, yaitu *train, validation-1, validation-2*, dan *test set*. Untuk mendapatkan hasil yang terbaik, semua set validasi dan set test direkayasa untuk menyerupai *challenge set* yang diberikan. Hasilnya, masing-masing set validasi dan set test memiliki 1000 *playlist* untuk setiap skenario *challenge set*. Karena *challenge set* yang disediakan memiliki 10 skenario, maka jumlah total *playlist* yang berada di *validation-1, validation-2*, dan *test* masing-masing adalah 10.000. Pembagian ini dijabarkan dalam Gambar 1.



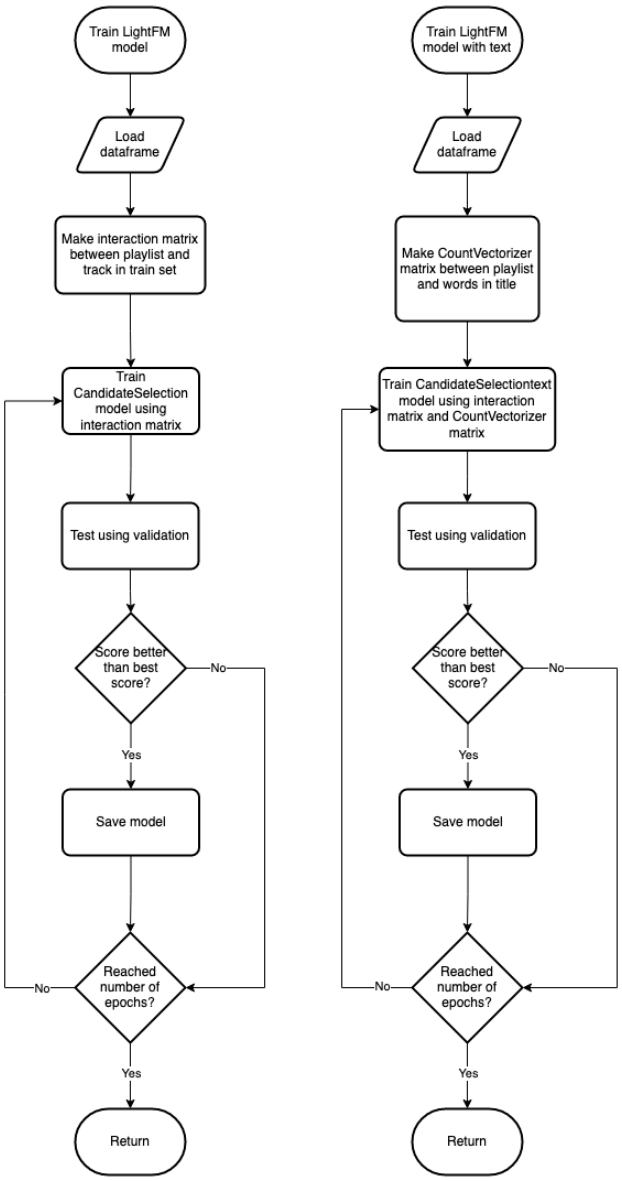
Gambar 1. Pembagian *playlist*

Set validasi digunakan untuk men-*train* model-model yang digunakan. *Validation-1* akan digunakan untuk menguji dan menyeleksi iterasi model LightFM dan XGBoost yang terbaik. *Validation-2* akan digunakan untuk melakukan *hyperparameter tuning* model XGBoost yang sudah di-*train*. *Test* akan digunakan untuk pengujian akhir.

4.3 Training Model LightFM

Proses *training* model mengikuti metode [6] dan digambarkan dalam *flowchart* di Gambar 2. Dua model LightFM akan di-*train* untuk seleksi *track* kandidat. Model pertama, yang dinamakan $CandidateSelection$, akan digunakan untuk menyeleksi kandidat bagi *playlist* yang memiliki *seed track*. Model kedua, yang dinamakan $CandidateSelection_{text}$, akan digunakan untuk menyeleksi kandidat dalam skenario *cold-start*, yaitu bagi *playlist* yang hanya memiliki nama dan tidak mempunyai *track* di dalamnya. Kedua model ini di-*train* menggunakan *train set* dan diuji menggunakan *validation-1*. Proses *training* kedua model akan dijalankan sejumlah kali, dan dalam setiap iterasi, performa model akan dihitung menggunakan *precision at k*. Iterasi model yang memiliki skor *precision at k* terbaik akan digunakan dalam seleksi kandidat.

Model $CandidateSelection_{text}$ juga mengggunakan input berupa fitur *user* – atau dalam kasus ini fitur *playlist* – berupa kata-kata dalam nama *playlist*. Dua ribu kata-kata yang paling sering muncul di nama-nama *playlist* dijadikan sebagai *vocabulary*. Lalu seperti input pertama, input fitur *playlist* direkayasa menggunakan *count vectorizer* menjadi *sparse matrix* antara *playlist* dengan frekuensi munculnya kata-kata di *vocabulary*.



Gambar 2. *Flowchart training* model LightFM

4.4 Seleksi Kandidat

Untuk meningkatkan efisiensi waktu *training* di langkah-langkah selanjutnya, sejumlah kandidat terbaik akan dipilih dari 10.000 kandidat yang sudah ada. Jumlah kandidat yang dipilih untuk setiap *playlist* bersifat proposional dengan jumlah *track* yang disembunyikan (*holdout track*). Jika suatu *playlist* memiliki *k track holdout*, maka jumlah kandidat yang dipilih adalah $top\_max(k \cdot 8, k + 700)$. Angka 8 dan 700 dipilih untuk menjaga jumlah total *track* kandidat yang di-*train* supaya dapat diproses oleh *memory* yang tersedia.

4.5 Rekayasa Fitur

Ada tiga macam utama fitur yang direkayasa untuk XGBoost.

4.5.1 Fitur LightFM

Fitur-fitur model LightFM yang digunakan dalam *ranking* diambil dari $CandidateSelection$ dan $CandidateSelection_{text}$. Ada 5 nilai yang digunakan sebagai fitur dari masing-masing model. Empat dari 5 nilai tersebut didapatkan dari rumus skor LightFM (rumus 1). Nilai-nilai tersebut adalah $score(p, t)$ (skor kandidat), b_p (nilai bias *playlist*), b_t (nilai bias *track*), $\langle q_p, q_t \rangle$ (*dot product* dari *latent factor playlist* dan *track*). Selain itu, fitur lainnya yang direkayasa adalah posisi *rank track* tersebut relatif dengan kandidat-kandidat lainnya untuk suatu *playlist* tertentu.

4.5.2 Fitur Co-occurrence

Nilai *co-occurrence* adalah frekuensi munculnya 2 di *playlist* yang sama. Nilai ini dapat dinyatakan dengan $n_{i,j}$ di mana i dan j menandakan *track* tertentu. Fitur ini dikalkulasi bagi setiap *playlist* p dan *track* kandidat t. *Playlist* p memiliki *seed track* $t_1, t_2, \ldots, t_n$. Fitur-fitur yang dihasilkan adalah nilai mean, minimum, maksimum, dan median dari setiap nilai *co-occurrence* dengan *seed track* dalam p, yaitu dari $n_{t,t_1}, n_{t,t_2}, \ldots, n_{t,t_n}$.

Selain itu, nilai-nilai mean, minimum, maksimum, dan median juga dihitung dari setiap nilai *co-occurrence* yang sudah dinormalisasi., Nilai *co-occurrence* dihitung antara 1 *track* kandidat dan 2 *seed track*, sehingga nilai *co-occurrence* yang dinormalisasi memiliki bentuk: $\frac{n_{t,t_i,t_j}}{n_{t_i,t_j}}$. Nilai ini dikalkulasi untuk setiap pasangan *seed track* t_i dan t_j di dalam p. Seperti yang dijelaskan di Pendahuluan, perhitungan ini dilakukan dengan dua *seed track* untuk melemahkan asumsi bahwa probabilitas suatu *track* ada di dalam *playlist* bersifat independen dari *track* lainnya yang di dalam *playlist*.

4.5.3 Fitur Lainnya

Berikut fitur lainnya yang digunakan untuk *ranking*, sesuai dengan yang dilakukan oleh [6]

- Jumlah artis dan album unik di dalam *playlist*
- Jumlah *track* yang memiliki artis atau album yang sama dengan *track* kandidat
- Frekuensi munculnya *track* kandidat di dalam seluruh dataset (bisa juga disebut sebagai tingkat popularitas *track* kandidat di dalam dataset)
- Frekuensi munculnya artis dan album *track* kandidat di seluruh dataset (tingkat popularitas artis dan album di dataset)
- Jumlah *track* di dalam *seed* dan *holdout playlist*

4.6 Ranking Menggunakan XGBoost

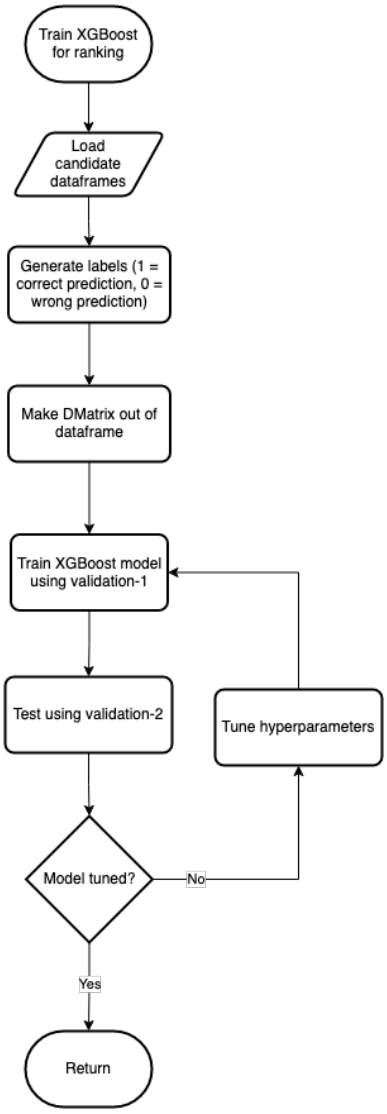
Proses ini dilakukan menggunakan set *validation-1* dan *validation-2*. *Training* awal dilakukan menggunakan *playlist* dan kandidat dari *validation-1* dengan menggunakan *binary logistic loss*. Lalu, *tuning* untuk *hyperparameter* dilakukan menggunakan *validation-2* dan dilakukan secara manual. Proses *training* ini dilakukan untuk memaksimalkan nilai *area under ROC*. Gambar 3 menampilkan *flowchart* dari proses *training* model XGBoost. Setelah *training* dan *tuning* selesai, model XGBoost digunakan untuk mengambil dan mengurutkan 500 *track* kandidat bagi setiap *playlist* di *test set*.

5. PENGUJIAN SISTEM

Bagian ini menjelaskan tolak ukur yang digunakan untuk pengujian dan hasil yang didapat dari pengujian.

5.1 Tolak Ukur

Tolak ukur yang digunakan mengikuti tolak ukur evaluasi RecSys Challenge 2018. Ada 3 tolak ukur yang digunakan yaitu R-precision, NDCG, dan *recommended song clicks*. Tiga ukuran ini dijelaskan di beberapa poin selanjutnya.



Gambar 3. *Flowchart training* model XGBoost

5.1.1 R-precision

R-precision adalah sebuah nilai yang mengukur jumlah *item* rekomendasi yang termasuk di dalam semua *item* yang relevan [1]. Di RecSys Challenge 2018, R-precision tidak hanya diukur dari jumlah *track* rekomendasi yang relevan (S_T) dari seluruh *track* yang relevan (G_T), tetapi juga jumlah artis rekomendasi yang relevan (S_A) dari seluruh artis yang relevan (G_A). Maka, jumlah artis rekomendasi yang relevan tetap akan menghasilkan sebuah skor dengan *weight* 0.25 meski *track* yang direkomendasikan tidak relevan atau salah. Hal ini dijabarkan dalam rumus berikut:

$$R - precision = \frac{|S_T \cap G_T| + 0.25 \cdot |S_A \cap G_A|}{|G_T|} \quad (2)$$

5.1.2 Normalized Discounted Cumulative Gain (NDCG)

Berbeda dengan R-precision, NDCG mengukur seberapa baik urutan *ranking* yang dihasilkan oleh suatu sistem rekomendasi [3]. Nilai NDCG akan lebih tinggi jika *item* yang relevan ditempatkan di posisi yang lebih tinggi pada *ranking* rekomendasi. NDCG didapatkan dari menghitung *discounted cumulative gain* atau DCG dan membaginya dengan *ideal* DCG.

$$DCG = rel_1 + \sum_{i=2}^{|R|} \frac{rel_i}{i} \quad (3)$$

Ideal DCG adalah DCG yang didapatkan dari rekomendasi yang memiliki *ranking* sempurna:

$$IDCG = 1 + \sum_{i=2}^{|G \cap R|} \frac{1}{i} \quad (4)$$

Maka, NDCG yang dihasilkan dijabarkan sebagai berikut:

$$NDCG = \frac{DCG}{IDCG} \quad (5)$$

5.1.3 Recommended song clicks (song clicks)

Recommended song clicks adalah ukuran evaluasi yang didapatkan berdasarkan fitur Spotify bernama *Recommended Songs* [1]. Fitur ini memberikan 10 lagu rekomendasi yang bisa ditambahkan ke dalam suatu *playlist.* Daftar rekomendasi lagu tersebut dapat di-*refresh* untuk menghasilkan 10 lagu rekomendasi baru. *Recommended song clicks* adalah jumlah *refresh* yang dibutuhkan sampai ditemukan satu *track* yang relevan. Nilai *recommended song clicks* didapatkan dengan rumus sebagai berikut:

$$clicks = \left\lfloor \frac{argmin_i\{R_i : R_i \in G\} - 1}{10} \right\rfloor \quad (6)$$

5.2 Hasil Pengujian

Hasil evaluasi dan perbandingan antara metode [6] dengan model-model di penelitian ini dapat dilihat di Tabel 1. Percobaan pertama menggunakan model XGBoost dengan parameter yang sama dengan [6], sedangkan percobaan akhir menggunakan model XGBoost yang sudah di-*tuning*.

Tuning dilakukan dengan menambah parameter *max_depth* dan *min_child_weight* yang akan membuat masing-masing *tree* dalam XGBoost semakin kompleks. Semakin kompleks *tree* yang ada di model XGBoost, model tersebut diharapkan bisa *fit* data dengan lebih baik. Namun seperti yang dapat dilihat di Tabel 1, proses *tuning* hanya membuat model memiliki nilai lebih tinggi di NDCG dan *recommended song clicks*.

Normalized co-occurrence menggunakan 3 *track* tidak memberikan dampak yang signifikan terhadap keseluruhan algoritma. Metode [6] menggunakan *normalized co-occurrence* dengan 2 *track* tetap memberikan performa yang sedikit lebih baik.

Saat f*eature importance* dihitung bagi masing-masing model XGBoost, dapat dilihat bahwa *normalized co-occurrence* dengan 3 *track* tidak lagi menjadi fitur yang berdampak penting bagi model XGBoost. Hal ini dapat diliaht di Tabel 2 dan 3.

Tabel 1. Perbandingan hasil kedua metode

Model	R-prec	NDCG	Song Clicks
Rubtsov et al.	0,52505060	0,5581982	1,29485986
Percobaan pertama	0,52460647	0,5576182	1,34925558
Percobaan akhir	0,52408643	0,5578853	1,31192471

Tabel 2. *Feature gain* dari XGBoost Rubtsov et al.

Feature	Gain
Co-occurrence normalized max	2095
Co-occurrence normalized mean	1832
LightFM rank	396
Tracks holdout	248
LightFM rank text	169
LightFM dot product text	163
Co-occurrence normalized median	156
Co-occurrence median	150
Co-occurrence min	138
LightFM dot product	126

Tabel 3. *Feature gain* dari XGBoost penelitian ini

Feature	Gain
LightFM rank	594
Co-occurrence normalized mean	424
Co-occurrence normalized max	192
Tracks holdout	98
Co-occurrence normalized median	87
LightFM dot product text	82
Co-occurrence mean	63
LightFM rank text	50
Mean artist in playlist	35
Co-occurrence max	28

6. KESIMPULAN DAN SARAN

Dalam kasus *automatic playlist continuation* menggunakan data *playlist* dan *track* seperti Million Playlist Dataset, perhitungan *normalized co-occurrence* menggunakan 2 *track* lebih baik dari menggunakan 3 *track*. Perhitungan *co-occurrence* dapat dilihat sebagai sebuah kasus *association rule mining*. Diduga *normalized co-occurrence* menggunakan 3 *track* memiliki *support* yang lebih rendah dibandingkan dengan *normalized co-occurrence* menggunakan 2 *track*. Hal ini juga didukung dengan penggunaan dataset yang hanya sebesar 20% karena keterbatasan *computing resource*. Maka, *normalized co-occurrence* dengan 3 *track* tidak cukup bisa digunakan untuk menentukan rekomendasi *track* kepada *playlist*.

Saran pengembangan untuk penelitian ini adalah penggunaan dataset Million Playlist Dataset secara penuh. Dataset yang lebih besar dapat meningkatkan *confidence* dan *support* dalam perhitungan *normalized co-occurrence* dengan 3 *track* sehingga diharapkan dapat meningkatkan performa keseluruhan sistem.

7. REFERENSI

[1] Chen, C. W., Schedl, M., Lamere, P., & Zamani, H. 2018. Recsys challenge 2018: Automatic music playlist continuation. In *RecSys 2018 - 12th ACM Conference on Recommender Systems*, 527–528. DOI=https://doi.org/10.1145/3240323.3240342

[2] Chen, T., & Guestrin, C. 2016. XGBoost: A scalable tree boosting system. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, *13-17-August-2016*, 785–794. DOI=https://doi.org/10.1145/2939672.2939785

[3] IFPI. 2020. *Global Music Report: The Industry in 2019*. URI=https://www.ifpi.org/wp-content/uploads/2020/07/Global_Music_Report-the_Industry_in_2019-en.pdf

[4] Järvelin, K., & Kekäläinen, J.(2002. Cumulated Gain-Based Evaluation of IR Techniques. In *ACM Transactions on Information Systems*, *20*(4), 422–446. DOI=https://doi.org/10.1145/582415.582418

[5] Kula, M. 2015. Metadata Embeddings for User and Item Cold-start Recommendations. In *Proceedings of the 2nd Workshop on New Trends on Content-Based Recommender Systems Co-Located with 9th ACM Conference on Recommender Systems*, 1448, 14–21.

[6] Rubtsov, V., Kamenshchikov, M., Valyaev, I., Leksin, V., & Ignatov, D. I. 2018. A hybrid two-stage recommender system for automatic playlist continuation. In *ACM International Conference Proceeding Series*. DOI=https://doi.org/10.1145/3267471.3267488

[7] Zamani, H., Schedl, M., Lamere, P., & Chen, C. W. 2019. An analysis of approaches taken in the ACM Recsys challenge 2018 for automatic music playlist continuation. In *ACM Transactions on Intelligent Systems and Technology*, *10*(5). DOI=https://doi.org/10.1145/3344257