Pengimplementasian Sistem Rekomendasi Musik Dengan Metode *Collaborative Filtering*

Ivander Yoshua[#1], Hendra Bunyamin, S.Si., M.T. [*2]

[#]*Program Studi Teknik Informatika, Universitas Kristen Maranatha*

Jl. Surya Sumantri No.65 Bandung 40164

[1]ivanderyoshua@gmail.com

[2]hendra.bunyamin@it.maranatha.edu

Abstract — **Nowdays, *Recommendation System*s are widely used on various platforms such as e-commerce, online cinemas, youtube, and online music streaming platforms. The *Recommendation System* is a machine learning algorithm that has goal to aim of providing predictions in the form of values or an action on an *item* that is given or given to a number of users, so that users can find new *item*s that match what user likes and preferences of each user. The *Recommendation System* has a variety of methods that can be implemented, one of which is the collaborative screening method. *Collaborative filtering* method works by providing predictions on a number of *item*s to be given to the user based on the interaction value the user gives with these *item*s. However, the *collaborative filtering* method itself has various types of algorithms which can be broadly divided into 2, namely the *KNN* algorithm and *Matrix Factorization*. In this study, a *collaborative filtering* based music *Recommendation System* was implemented. Then conducted a study on the evaluation of the music *Recommendation System* using the *KNN* algorithm with a music *Recommendation System* using a *Matrix Factorization* algorithm.**

Keywords — ***Collaborative filtering*, *KNN*, *Matrix Factorization*, Music, *Recommendation System*.**

I. PENDAHULUAN

Sistem rekomendasi memiliki tujuan untuk memberi rekomendasi pada pengguna sebuah item (dalam kasus ini judul lagu, artis, album) yang sesuai dengan preferensi, kesukaan musik masing-masing setiap pengguna. Namun kebanyakan orang masih belum mengetahui betul prinsip atau cara kerja dari sistem rekomendasi dan jika diperhatikan dengan seksama setiap sistem rekomendasi yang diimplementasikan oleh setiap platform layanan streaming online musik memiliki cara kerja yang berbeda untuk merekomendasikan sebuah lagu pada pengguna. Masalah lainnya yang dihadapi adalah bagaimana cara mengujikan atau melakukan testing pada kinerja atau performa dari sistem rekomendasi yang telah dibuat di samping itu mengingat banyaknya jenis metode sistem rekomendasi memungkinkan peneliti atau orang yang ingin mencoba pertama kali mengimplementasikan sistem rekomendasi terkadang merasa bingung untuk menentukan sistem rekomendasi mana yang memiliki kinerja paling bagus mengingat masing-masing metode memiliki kelebihannya tersendiri.

Untuk itu Penelitian ini dibuat dengan tujuan membuat model sistem rekomendasi musik menggunakan library yang mendukung. Lalu mengimplementasikan model algoritma pada dataset yang sudah disediakan. Kemudian Membuat beberapa model sistem rekomendasi musik dengan pendekatan algoritma yang berbeda. Setelah itu melakukan perbandingan terhadap hasil akhir kinerja dan performa yang dihasilkan oleh masing-masing algoritma menggunakan beberapa module perhitungan akurasi. Terakhir Melakukan analisis terhadap hasil akhir perhitungan akurasi yang didapat pada masing-masing algoritma.

II. KAJIAN TEORI

A. Sistem Rekomendasi

Sistem rekomendasi adalah suatu program atau algoritma yang bertujuan untuk memberikan prediksi atau aksi pada suatu *item* untuk diberikan atau direkomendasikan pada sejumlah pengguna [1]. Sugesti yang diberikan oleh sistem rekomendasi pada sejumlah pengguna dapat berupa berbagai macam aksi yang dilakukan oleh pengguna terhadap *item* tersebut misalnya memberi sugesti pada pengguna akan barang yang ingin dibelinya, musik yang ingin didengarkan oleh pengguna, atau buku yang ingin dibaca oleh pengguna. Sistem rekomendasi sendiri dapat diklasifikasikan menjadi 2 kategori berdasarkan pada pendekatan yang dipakai untuk mengimplementasikan sistem rekomendasi, yaitu memecahkan masalah dengan prediksi dan memecahkan masalah dengan *ranking* [1].

B. Collaborative Filtering Method

Collaborative filtering adalah salah satu metode yang digunakan pada sistem rekomendasi yang bekerja berdasarkan interaksi antara pengguna dengan *item-item* yang direkam atau disimpan yang nantinya digunakan untuk membuat sistem rekomendasi [3]. Interaksi-interaksi tersebut kemudian disimpan pada sebuah matriks yang besar berukuran m kali n notasi m dilambangkan sebagai banyaknya pengguna pada matriks tersebut sedangkan notasi n dilambangkan sebagai banyaknya *item* pada matriks tersebut dan setiap elemen-elemen pada matriks dilambangkan sebagai interaksi antara pengguna dengan *item* tersebut. Algoritma *collaborative filtering* terbagi menjadi 2 bagian atau sub-kategori yang secara umum disebut *memory based* dan *model based.*

1) *Memory Based Collaborative Filtering*: *Memory Based Collaborative Filtering* bekerja dengan cara memanfaatkan interaksi-interaksi yang sudah pernah dilakukan antara pengguna dan *item* yang kemudian interaksi-interaksi tersebut disimpan dalam sebuah matriks [3].
2) *Model Based Collaborative Filtering: Model Based Collaborative filtering* menggunakan interaksi antara *item* dan pengguna untuk implementasinya namun yang membedakannya dengan metode *memory based collaborative filtering*, *model based collaborative filtering* menggunakan sebuah *latent* model untuk menjelaskan interaksi antara pengguna dan *item* tersebut [3].

C. Algoritma K-Nearest Neighbor Model (KNN)

KNN adalah algoritma non-parameteric, serta memiliki metode pembelajaran yang low atau rendah. *KNN* menggunakan sebuah database yang nantinya titik antara data-data dipisahkan menjadi beberapa bagian cluster untuk membuat sampel baru yang berbeda [4]. *KNN* dapat diimplementasikan pada *memory-based collaborative filtering* dengan metode *item-item based* ataupun *user-user based* dengan menggunakan rumus perhitungan nilai *similarity* untuk mencari suatu nilai kesamaan antara pengguna atau *item* dengan tanpa model tambahan [4]. Rumus prediksi dari *KNN* sendiri dapat dijabarkan pada Rumus 1.

$$\hat{r}_{ui} = \frac{\sum_{v \in N_i^k(u)} Sim(u,v) \cdot r_{vi}}{\sum_{v \in N_i^k(u)} Sim(u,v)} \quad \hat{r}_{ui} = \frac{\sum_{j \in N_u^k(i)} Sim(i,j) \cdot r_{uj}}{\sum_{j \in N_u^k(i)} Sim(i,j)}$$

Rumus 1 Rumus Algoritma K-Nearest Neighbor (*KNN*)

Terlihat pada Rumus 1. *KNN* membutuhkan pendekatan berupa perhitungan komputasi *distance* atau *similarity* dari *item-item* atau pengguna-pengguna yang akan dibandingkan. *Similarity* sendiri memiliki beberapa pendekatan algoritma yang berbeda.

1) *Cosine Similarity : Cosine similarity* menghitung sudut cosinus antara 2 *item* atau pengguna yang hendak dibandingkan. *Item-item* atau pengguna-pengguna tersebut direpresentasikan menjadi sebuah bentuk vektor. Rumus dari persamaan *cosine similarity* dapat dijabarkan pada Rumus 2.

$$cosine\_sim(i,j) = \frac{\sum_{u \in U_{ij}} r_{ui} \cdot r_{uj}}{\sqrt{\sum_{u \in U_{ij}} r_{ui}^2} \cdot \sqrt{\sum_{u \in U_{ij}} r_{uj}^2}}$$

atau

$$cosine\_sim(u,v) = \frac{\sum_{i \in I_{uv}} r_{ui} \cdot r_{vi}}{\sqrt{\sum_{i \in I_{uv}} r_{ui}^2} \cdot \sqrt{\sum_{i \in I_{uv}} r_{vi}^2}}$$

Rumus 2 Rumus Perhitungan *Cosine Similarity*

2) *Pearson Corellation* : *Pearson Correlation* melakukan perhitungan pada dua buah vektor sama seperti yang dilakukan dengan perhitungan *cosine similarity* namun *pearson correlation* disebut juga sebagai *cosine similarity mean-centered* karena dapat menghindari terjadinya bias pada data-data dengan cara data-data dapat dinormalisasi dahulu sehingga tidak terjadinya ketimpangan pada data [5]. Rumus *pearson correlation* dapat direpresentasikan ke dalam rumus yang dijabarkan pada Rumus 3.

$$Pearson\_sim(u,v) = \frac{\sum_{i \in I_{uv}}(r_{ui} - \mu_u).(r_{vi} - \mu_v)}{\sqrt{\sum_{i \in I_{uv}}(r_{ui} - \mu_u)^2} \cdot \sqrt{\sum_{i \in I_{uv}}(r_{vi} - \mu_v)^2}}$$

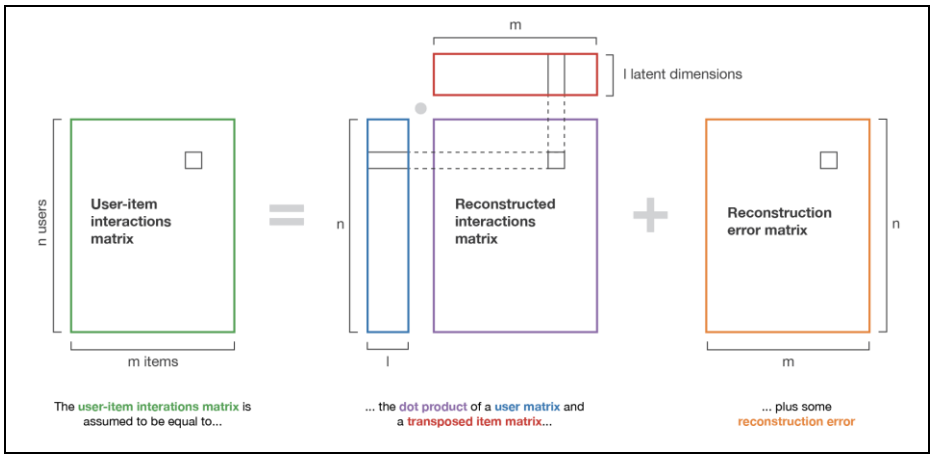
atau

$$Pearson\_sim(i,j) = \frac{\sum_{u \in U_{ij}}(r_{ui} - \mu_i).(r_{uj} - \mu_j)}{\sqrt{\sum_{u \in U_{ij}}(r_{ui} - \mu_i)^2} \cdot \sqrt{\sum_{u \in U_{ij}}(r_{uj} - \mu_j)^2}}$$

Rumus 3 Rumus Perhitungan *Pearson Corellation Similarity*

D. Matrix Factorization

Matrix factorization bekerja dengan cara melakukan dekomposisi sebuah matriks yang berukuran besar ke dalam bentuk matriks yang lebih kecil [8]. Hasil akhir dari matriks tersebut adalah hasil dot produk dari matriks pengguna dan *transpose* dari matriks *item*. Untuk lebih jelasnya dapat melihat Gambar 1.



Gambar 1 Cara Kerja *Matrix Factorization*

Algoritma *matrix factorization* pada penerapannya sendiri memiliki beberapa model pendekatan algoritma yang berbeda-beda sama halnya seperti perhitungan *similarity* pada algoritma *KNN*. Model algoritma *matrix factorization* antara lain yaitu *SVD*, *SVD++*, *NMF*, dan *PMF*.

1) *SVD : SVD* atau yang biasa disebut (*Singular Value Decomposition*) merupakan sebuah algoritma dari *Matrix Factorization* yang dikembangkan oleh Simon Funk pada tahun 2006 [7]. Konsep dekomposisi matriks dari algoritma *matrix factorization* kemudian di applikasikan oleh algoritma *SVD* ke dalam bentuk rumus untuk menghasilkan nilai prediksi dari algoritma *SVD*. Bentuk dari rumus prediksi *SVD* terlihat pada Rumus 4.

$$\hat{r}_{ui} = \mu + b_u + b_i + q_i^T p_u$$

Rumus 4 Rumus Prediksi Algoritma *SVD*

Untuk memperkirakan atau menghitung semua nilai *rating* yang belum diketahui *SVD* dapat mengatasinya dengan cara meminimalisir *regularized square error* dengan rumus yang dijabarkan pada Rumus 5.

$$\sum_{r_{ui} \in R_{train}} (r_{ui} - \hat{r}_{ui})^2 + \lambda \left({b_i}^2 + {b_u}^2 + \|q_i\|^2 + \|p_u\|^2 \right)$$

Rumus 5 Rumus *Cost Function* dari Algoritma *SVD*

Untuk meminimalisir *error* yang dihasilkan oleh model algoritma dalam membuat prediksi dibutuhkan langkah-langkah formal yang dilakukan oleh *gradient descent* dengan rumus-rumus yang dijabarkan sesuai pada Rumus 6.

$$b_u \leftarrow b_u + \gamma(e_{ui} - \lambda b_u)$$
$$b_i \leftarrow b_i + \gamma(e_{ui} - \lambda b_i)$$
$$p_u \leftarrow p_u + \gamma(e_{ui} \cdot q_i - \lambda p_u)$$

$$q_i \leftarrow q_i + \gamma(e_{ui} \cdot p_u - \lambda q_i)$$

Rumus 6 *Gradient Descent* dari *Algoritma SVD*

2) *PMF : PMF* (*Probabilistic Matrix Factorzation)* merukapan algoritma turunan dari *SVD*. Algoritma *PMF* menghilangkan parameter bias dari pengguna dan *item* untuk menghitung prediksi dari suatu interaksi, sehingga rumus *PMF* dapat dijbarkan sesuai pada Rumus 7.

$$\hat{r}_{ui} = q_i^T p_u$$

Rumus 7 Prediksi dari Algoritma *PMF*

3) *SVD++* : *SVD++* merupakan bentuk ekstensi dari *SVD* yang dikembangkan oleh Yehuda Koren. Berbeda dengan *SVD* yang dikembangkan oleh Simon Funk, *SVD++* dikembangkan untuk melibatkan data-data yang bersifat implisit yang berguna dalam pengembangan teknik dari sistem rekomendasi [11]. Implisit data sendiri adalah sebuah interaksi, *feedback* yang diberikan pada pengguna terhadap *item* yang secara tidak langsung mencerminkan atau menggambarkan preferensi pengguna tersebut [12]. Rumus dari *SVD++* dapat dijabarkan sesuai dengan ketentuan yang terlihat pada Rumus 8.

$$\hat{r}_{ui} = \mu + b_u + b_i + q_i^T \left(p_u + |I_u|^{-\frac{1}{2}} \sum_{j \in I_u} y_j \right)$$

Rumus 8 Nilai prediksi *SVD++*

4) *NMF* : Algoritma *NMF* (*Non-Negative Matrix factorization*) memiliki cara kerja yang mirip dengan *SVD* yaitu dengan cara melakukan dekomposisi pada matriks yang berukuran besar menjadi 2 buah matriks berukuran yang lebih kecil. Namun yang membedakan algoritma *NMF* dengan *SVD* adalah hasil *matrix factorization* dari pengguna maupun *item* pada algoritma *NMF* harus selalu bernilai positif [13]. Rumus prediksi *NMF* memiliki rumus yang sama percis dengan algoritma *SVD*. Namun terjadi perubahan rumus pada saat proses update nilai pada algoritma *NMF*. Rumus perubahan nilai yang digunakan pada algoritma *NMF* dijabarkan pada Rumus 9.

$$p_{uf} \leftarrow p_{uf} \cdot \frac{\sum_{i \in Iu} q_{if} \cdot r_{ui}}{\sum_{i \in Iu} qif \cdot \widehat{r_{ui}} + \lambda_u |I_u| p_{uf}}$$

$$q_{if} \leftarrow q_{if} \cdot \frac{\sum_{u \in Ui} P_{uf} \cdot r_{ui}}{\sum_{u \in Ui} puf \cdot \widehat{r_{ui}} + \lambda_i |U_i| q_{if}}$$

Rumus 9 Update nilai algoritma *NMF* pada *item* dan pengguna

III. PERANCANGAN SISTEM

A. Dataset Yang Digunakan

Pada penelitian ini dataset yang digunakan berupa 2 buah metadata yang digabungkan menjadi sebuah metadata. Metadadata yangdigunakan terdiri dari metadata pengguna dan metadata berisi informasi-informasi mengenai lagu yang dijabarkan sesuai pada Tabel 1. dan Tabel 2.

TABEL I

METADATA PENGGUNA

	User_id	Song_id	Listen_count
1	b80344d063b5ccb3212f76538f3d9e43d87dca9e	SOAKIMP12A8C130995	1.0
2	b80344d063b5ccb3212f76538f3d9e43d87dca9e	SOBBMDR12A8C13253B	2.0
3	b80344d063b5ccb3212f76538f3d9e43d87dca9e	SOBXHDL12A81C204C0	1.0
4	b80344d063b5ccb3212f76538f3d9e43d87dca9e	SOBYHAJ12A6701BF1D	1.0

5	b80344d063b5ccb3212f76538f3d9e43d87dca9e	SODACBL12A8C13C273	1.0

TABEL II

METADATA LAGU

	Song_id	Title	Release	Artist_name	Year
1	SOQMMHC12AB0180CB8	Silent Night	Mother Ballads X-mas	Faster PussyCat	2003
2	SOVFVAK12A8C1350D9	Taansi Vaan	Karkuteilla	Karkkiautomaatti	1995
3	SOGTUKN12AB017F4F1	No One Could Ever	Butter	Hudson Mohawke	2006
4	SOBNYVR12A8C13558C	Si Vos Queres	De Culo	Yerba Brava	2003
5	SOHSBXH12A8C13B0DF	Tangle Of Aspens	Rene Ablaze Presents Winter Sessions	Der Mystic	0

Tabel 1. menunjukan isi dari dataset metadata pengguna. Tabel 2. menunjukan isi dari dataset metada lagu. Pada penelitian ini dataset yang digunakan berjumlah sebanyak 75123 baris yang terdiri dari 2934 pengguna unik dan 9749 lagu berbeda.

B. Validation Test

Fitur yang dijadikan sebagai acuan untuk melatih model-model algoritma collaborative filtering dan memberikan rekomendasinya adalah kolom listen_count. Dengan ketentuan jika nilai dari listen_count bernilai tinggi maka nilai *rating* yang diberikan semakin baik dan berlaku untuk hal sebaliknya. Lebih lanjut setelah model-model algoritma tersebut dilatih, kinerja dari masing-masing model algoritma akan dievaluasi. Dalam melakukan evaluasi hasil kinerja algoritma pada machine learning, dataset yang digunakan pada penelitian ini akan dipecah menjadi 2 bagian. Lebih lanjut dataset dipecah ke dalam trainset dan testset. Persentase untuk data di dalam trainset sebesar 75% sedangkan persentase untuk data di dalam testset sebesar 25%..

Namun metode split data ke dalam *trainset* dan *testset* kurang menjamin untuk menghitung tingkat keakuratan akhir dari algoritma karena data-data yang terdapat pada *testset* akan selalu bernilai tetap sehingga performa algoritma tidak teruji pada *testset* dengan nilai yang berbeda-beda. Untuk mengatasi masalah tersebut metode *cross validation* menyediakan solusinya.

Cross validation bekerja dengan cara memecah dataset ke beberapa bagian. Selanjutnya salah satu bagian dipilih untuk menjadi *testset* sedangkan bagian lainnya yang tidak dipilih digunakan sebagai *trainset*. Proses *cross validation* akan beriterasi sebanyak jumlah bagian yang ditentukan. Setiap algoritma hendak dievaluasi kinerjanya dengan menggunakan metode *cross validation*. Algoritma yang hendak dievaluasi kinerjanya beserta dengan parameternya dapat dijabarkan pada scenario yang terdapat Tabel 3.

TABEL III

ALGORITMA YANG DIBANDINGKAN BESERTA DENGAN PARAMETER

Algoritma	Parameter
KNN user-user, cosine similarity	*KNN*Basic(*sim_options* = *cosine*, *user_based* = *False*)
KNN user-user, Pearson correlation	*KNN*Basic(*sim_options* = Pearson, *user_based* = *False*)
KNN item-item, cosine similarity	*KNN*Basic(*sim_options* = *cosine*, *user_based* = *False*)
KNN item-item, Pearson correlation	*KNN*Basic(*sim_options* = Pearson, *user_based* =
Matrix factorization SVD	*SVD*(*lr_all* = 0.005, *reg_all* = 0,02, biased = *False*, *n_factors* = 100, *n_epochs* = 20)
Matrix factorization SVDpp	*SVD*pp(*lr_all* = 0.005, *reg_all* = 0,02, biased = *False*, *n_factors* = 20, *n_epochs* = 20)

Matrix factorization PMF	*SVD* (*lr_all* = 0.005, reg_all = 0,02, biased = *False*, *n_factors* = 100, *n_epochs* = 20)
Matrix factorization NMF	*NMF*(*reg_pu* = 0.06, *reg_qi* = 0.06, *reg_bu* = 0.02, *reg_bi* = 0,02, *lr_bu* = 0.005, *lr_bi* = 0.005, *biased* = *False*, *n_factors* = 15, *n_epochs* = 50)

Tabel 3. Menunjukan scenario algoritma beserta parameter yang hendak dibandingkan performanya serta dievaluasi kinerjanya yang dilakukan pada penelitian ini.

IV. IMPLEMENTASI

A. Simple Exploratory Data Analysis (EDA)

Sebelum masuk pada tahap pengimplementasian algoritma sistem rekomendasi. Tahap pertama yang biasa dilakukan oleh data scientist atau praktisi machine learning adalah *Simple Exploratory Data Analysis* (*EDA*). *EDA* bertujuan agar data scientist atau praktisi machine learning dapat mengenal lebih dalam lagi perihal karakteristik dataset yang digunakan untuk uji coba pada pembuatan model machine learning. *EDA* sendiri terbagi lagi menjadi beberapa sub proses.

1) *Load Data* : Langkah *EDA* yang pertama dilakukan oleh data scientist adalah melakukan import modules. Modules yang umumnya digunakan yakni pandas dataframe. Langkah selanjutnya yang dilakukan adalah load data atau memuat data. Seperti yang dijelaskan pada bab 3, dataset yang digunakan pada penelitian ini terdiri atas 2 buah metadata yang terpisah yaitu metadata yang berisikan atribut dari data pengguna serta metadata yang berisikan atribut dari data lagu-lagu. Namun dikarenakan 2 dataset tersebut dimuat secara terpisah perlu digunakan sebuah metode untuk menggabungkan kedua dataset tersebut menjadi sebuah dataset yang utuh, Kemudian Untuk memudahkan dalam membaca serta mengolah dataset dapat menggunakan sebuah fungsi untuk menghapus kolom-kolom yang tidak diperlukan dalam penelitian, sehingga hasil akhir tabel dataset terlihat seperti pada Tabel 4.

TABEL IV

TAMPILAN DATA SETELAH DROP KOLOM

	User_id	**Song_id**	**Listen_count**	**Song**
0	b80344d063b5ccb3212f76538f3d9e43d87dca9e	SOAKIMP12A8C130995	1.0	The Cove - Jack Johnson
1	b80344d063b5ccb3212f76538f3d9e43d87dca9e	SOBBMDR12A8C13253B	2.0	Entre Dos Aguas - Paco De Lucia
2	b80344d063b5ccb3212f76538f3d9e43d87dca9e	SOBXHDL12A81C204C0	1.0	Stronger - Kanye West
3	b80344d063b5ccb3212f76538f3d9e43d87dca9e	SOBYHAJ12A6701BF1D	1.0	Constellations - Jack Johnson
4	b80344d063b5ccb3212f76538f3d9e43d87dca9e	SODACBL12A8C13C273	1.0	Learn To Fly - Foo Fighters

2) *Normalisasi* Data

Proses normalisasi yang dilakukan dengan cara menambah sebuah kolom baru yang diberi nama *Rating*. Kolom ini yang nantinya digunakan sebagai acuan untuk sistem rekomendasi bekerja. Bentuk formula dari rumus normalisasi yang digunakan dapat dilihat pada Rumus 10.

$$Z_i = \frac{X_i - \bar{X}}{S}$$

Rumus 10 Normalisasi Data

TABEL V

HASIL NORMALISASI DATA

	User_id	**Song_id**	***Rating***

31179	bb85bb79612e5373ac714fcd4469cabeb5ed94e1	SOZQSV312A8C13C271	13.743.582
72747	9b6440915ca3250a333012566f5b5bc98c912268	SOAPBTAF729BFB8	8.344.204
31649	b8bcd5537ffe1392d48894b88e038802eb685da6	SOMAKIT12A58A7E292	6.444.423
55156	a4b540b4e7d83b97022ca7fdf7c31a9bf0db05a3	SOLGIWB12A58A77A05	4.961.260
69298	c0b6f8728119cb1e3ef96148d3ac0180364ae15d	SOUFTBI12AB0183F65	14.660.143

Sesuai pada Tabel 5 range *rating* dari hasil normalisasi di atas tidak efektif karena memiliki range data yang sangat bervariasi. Range data yang dimiliki berkisar di antara angka -0,37 sampai 38,174701. Untuk itu perlu adanya langkah tambahan untuk melakukan normalisasi data pada nilai *rating* yang ada. Langkah normalisasi selanjutnya dilakukan dengan tujuan agar nilai range pada *rating* berkisar di nilai antara 1 sampai 5 dan hasilnya seperti pada Tabel 6.

TABEL VI

HASIL AKHIR DARI NORMALISASI RATING

	User_id	**Song_id**	***Rating***
31179	bb85bb79612e5373ac714fcd4469cabeb5ed94e1	SOZQSVB12A8C13C271	3.0
72747	9b6440915ca3250a333012566f5b5bc98c912268	SOAPBTS12AF729BFB8	2.0
31649	b8bcd5537ffe1392d48894b88e03880 2eb685da6	SOMAKIT12A58A7E292	2.0
55156	a4b540b4e7d83b97022ca7fdf7c31a9bf0db05a3	SOLGIWB12A58A77A05	2.0
69298	c0b6f8728119cb1e3ef96148d3ac0180364ae15d	SOUFTBI12AB0183F65	3.0

B. Implementasi Model Algoritma Sistem Rekomendasi

Seperti yang dibahas pada bab 2 penelitian ini menggunakan algoritma sistem rekomendasi dengan 2 pendekatan metode yang berbeda dari metode *collaborative filtering*, yaitu *neighborhood method (KNN)* dan *matrix factorization (SVD, SVD++, PMF ,NMF).* Seperti yang dibahas pada bab 3 ada sebanyak 8 algoritma yang hendak dibandingkan kinerja dan performanya pada penelitian ini. Masing-masing algoritma dibuat menggunakan library *surprise* yang kemudian modela algoritma dilatih meggunakan metode *cross-validation, train-test split.* Kemudian algoritma menghasilkan prediksinya lalu diukur tingkat keakurasian prediksi pada masing-masing algoritma dengan menggunakan perhitungan secara *RMSE* dan *MAE*, sehingga hasil *RMSE* dan *MAE* pada masing – masing algoritma tampak seperti Tabel 7.

TABEL VII

HASIL *RMSE* DAN *MAE* MASING-MASING ALGORITMA

Algoritma	***RMSE***	***MAE***
KNN User-User Cosine similarity	0.1297	0.0523
KNN User-User Pearson Similarity	0.1282	0.0509
KNN Item-Item Cosine similarity	0.1384	0.0521

KNN Item-Item Pearson Similarity	0.1260	0.0503
SVD	0.1474	0.1057
PMF	0.4078	0.1650
SVD++	0.1036	0.0726
NMF	0.0816	0.0329

Namun hasil tingkat *error* prediksi yang dihasilkan oleh masing-masing algoritma masih dapat diminimalisir. Untuk meminimalisir tingkat *error* dari prediksi yang dihasilkan dapat menggunakan beberapa metode dan salah satu metode yang dapat digunakan adalah *gridsearchcv. Gridsearchcv* merupakan sebuah fungsi yang memungkinkan untuk menemukan kombinasi parameter terbaik dari setiap algoritma agar tingkat *error* yang dihasilkan oleh hasil prediksi algoritma dapat diminimalisir.

Cara kerja dari *gridsearchcv* adalah diberikan sebuah *dictionary* berupa kumpulan-kumpulan parameter yang di set. Kemudian *gridsearchcv* melakukan perhitungan *RMSE* serta *MAE* terhadap parameter-parameter yang di set dengan menggunakan metode *cross validation*. Kemudian gridsearchcv mengeluarkan output berupa nilai *MAE dan RMSE* Setelah diberlakukanya metode *gridsearchcv* pada masing-masing algoritma, hasil perolehan nilai *RMSE* dan *MAE* berubah menjadi seperti pada hasil Tabel 8.

TABEL VIII

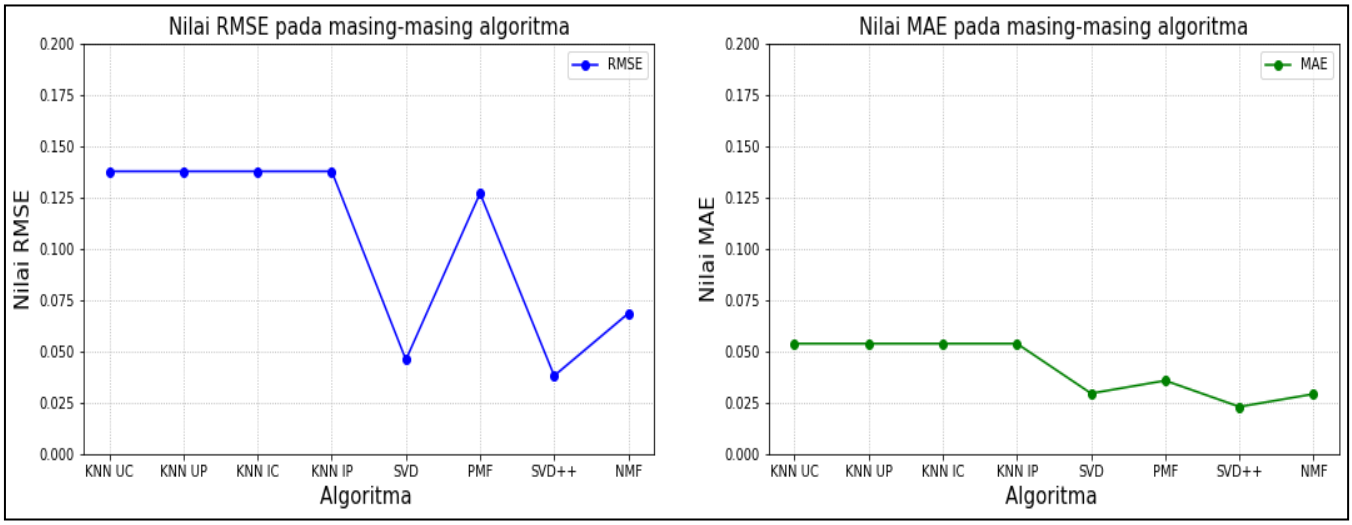
HASIL *RMSE* DAN *MAE* SETELAH DIIMPLEMENTASIKAN *GRIDSEARCHCV*

Algoritma	***RMSE***	***MAE***
KNN User-User Cosine similarity.	0.1295	0.0526
KNN User-User Pearson Similarity.	0.1295	0.0526
KNN Item-item Cosine similarity	0.1295	0.0526
KNN Item-item Pearson Similarity	0.1295	0.0526
SVD	0.0457	0.0291
PMF	0.1090	0.0325
SVD++	0.0386	0.0228
NMF	0.0677	0.0289

V. PENGUJIAN

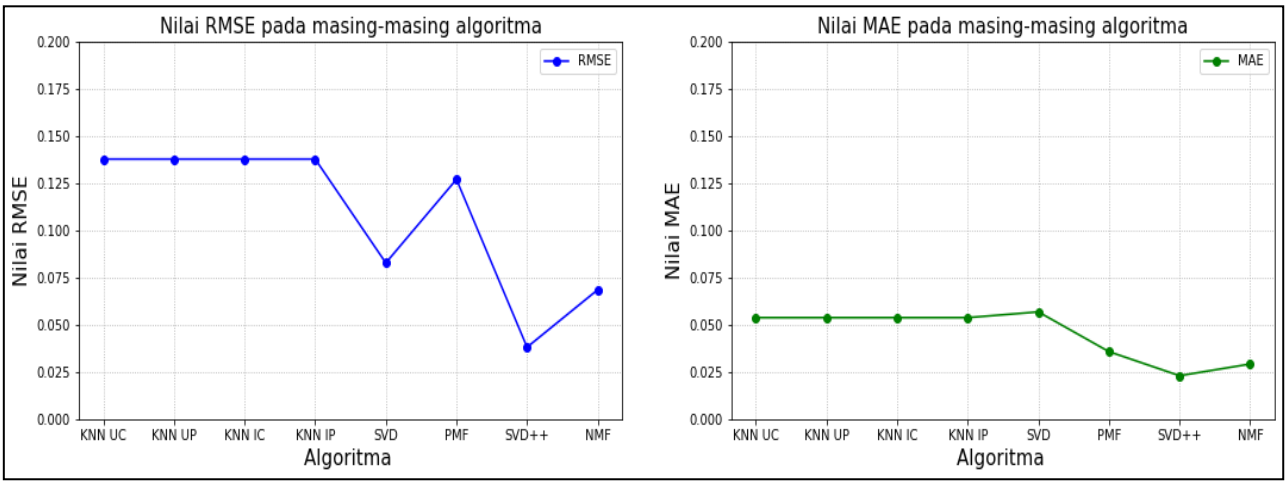
A. Analisis RMSE dan MAE

Pada bab ini hendak dijelaskan lebih dalam lagi perihal faktor-faktor yang berpotensi dapat mempengaruhi perubahan nilai dari *RMSE* dan *MAE*. Namun perlu diingatkan kembali bahwa semakin kecil nilai *RMSE* dan *MAE* yang dihasilkan oleh algoritma, menandakan bahwa hasil prediksi yang dibuat oleh algoritma terkait memiliki kinerja yang baik dan terjadi pada hal sebaliknya. Perolehan nilai *RMSE* dan *MAE* dapat divisualkan ke dalam bentuk histogram untuk memudahkan melakukan analisis perbandingan nilai *RMSE* dan *MAE* pada setiap algoritma. Untuk lebih jelasnya histogram dari perolehan nilai *RMSE* dan *MAE* pada masing-masing algoritma dapat dilihat pada Gambar 2.



Gambar 2 Nilai *RMSE* dan *MAE* Algoritma

Sesuai dengan Gambar 2. kinerja dari seluruh algoritma *KNN* ternyata tidak lebih baik dari seluruh algoritma *Matrix Factorization*. Hal ini jelas terlihat ketika keempat algoritma *KNN* memperoleh nilai *RMSE* dan *MAE* lebih tinggi jika dibandingkan dengan algoritma *matrix factorization*. Alasan utama tersebut terjadi dikarenakan algoritma *matrix factorization* memiliki lebih banyak *hyperparameter* yang dapat dikonfigurasi dan sangat berpengaruh untuk menghasilkan prediksi yang lebih akurat. Sebagai contoh kasus ketika nilai *learning rate* pada parameter algoritma *SVD* diturunkan yang semula 0.05 menjadi 0.007 yang terjadi adalah sesuai dengan hasil pada Gambar 3.



Gambar 3 Histogram Nilai *RMSE* dan *MAE* Perubahan *Learning Rate* pada Algoritma *SVD*

Pada Gambar 3. Nilai *RMSE* dan *MAE* pada algoritma *SVD* mengalami kenaikan secara drastis. Jika dikonversikan ke dalam bentuk angka nilai *RMSE* yang semula 0.0457 naik menjadi 0.0828. Sedangkan nilai *MAE* yang semula 0.0291 naik menjadi 0.0564. pada kasus penelitian ini nampaknya perubahan parameter yang digunakan oleh algoritma *KNN* tidak berpengaruh. Hal ini terlihat pada Gambar 2. dan Gambar 3. bahwa meskipun menggunakan jenis perhitungan similarity yang berbeda serta menggunakan metode *KNN* yang berbeda yaitu metode *user-user* , dan *item-item* namun nilai *RMSE dan MAE* yang dihasilkan tidak berubah sama sekali..

B. Precision and Recall Analysis

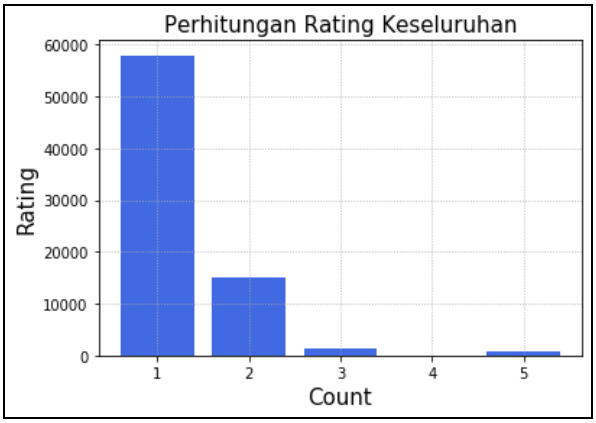
Module perhitungan akurasi yang selanjutnya digunakan adalah modul *precision* and *recall*. Untuk lebih jelasnya nilai *precision and recall* yang diperoleh masing-masing algoritma dapat dilihat pada Tabel 9.

TABEL IX

NILAI *PRECISION* AND *RECALL MASING-MASING ALGORITMA*

Algoritma	***Precision***	***Recall***	**K**	***Threshold***
KNN User-User Cosine Similarity	0.0652	0.0652	10	3.5
KNN User-User Pearson Similarity	0.0652	0.0652	10	3.5
KNN Item-Item Cosine Similarity	0.0652	0.0652	10	3.5
KNN Item-Item Pearson Similarity	0.0652	0.0652	10	3.5
SVD	0.0652	0.0652	10	3.5
PMF	0.0641	0.0641	10	3.5
SVD++	0.0652	0.0652	10	3.5
NMF	0.0652	0.0652	10	3.5

Seperti yang terlihat pada Tabel 9. modul perhitungan *precision and recall* dilakukan terhadap 10 rekomendasi terbaik pada masing-masing algoritma dengan batas nilai *threshold* = 3.5. Namun nilai *precision and recall* yang dihasilkan pada masing-masing algoritma tidak melebihi dari 10%.. Hal tersebut terjadi tentunya bukan tanpa sebab, faktor yang sangat krusial dibalik rendahnya nilai *precision and recall* yang diperoleh karena terjadi kasus *imbalance* data pada penelitian ini. Untuk lebih jelasnya dapat melihat Gambar 4.



Gambar 4 Kasus *Imbalance* Data

Dari histogram Gambar 4. dapat dilihat bahwa nilai *rating* sebenarnya yang memiliki nilai di atas 3.5 sangat sedikit. Namun untuk menghasilkan nilai *precision and recall* yang tinggi perlu nilai *rating* dengan kondisi harus lebih besar atau sama dengan batas nilai *threshold* agar *item* yang direkomendasikan bisa dikatakan relevan dan r*ecommended*. Untuk itu percobaan kedua dilakukan dengan cara menurunkan batas *threshold* yang semula di set pada 3.5 menjadi 2 hasilnya dapat dilihat pada Tabel 10.

TABEL X

NILAI *PRECISION* AND *RECALL* MASING-MASING ALGORITMA KETIKA *THRESHOLD DITURUNKAN*

Algoritma	***Precision***	***Recall***	**K**	***Threshold***
KNN User-User Cosine Similarity	0.5707	0.5589	10	2.0
KNN User-User Pearson Similarity	0.5707	0.5589	10	2.0
KNN Item-Item Cosine Similarity	0.5707	0.5589	10	2.0
KNN Item-Item Pearson Similarity	0.5707	0.5589	10	2.0
SVD	0.2105	0.1296	10	2.0
PMF	0.1955	0.1088	10	2.0
SVD++	0.1973	0.1193	10	2.0
NMF	0.2468	0.1855	10	2.0

Seperti yang terlihat pada Tabel 10. ketika nilai *threshold* di set menjadi 2 nilai *precision and recall* pada masing-masing algoritma meningkat secara drastis.

C. Analaisis Perbandingan Hasil Prediksi Algoritma

Hal lainnya yang dapat dilakukan untuk membandingkan kinerja algoritma adalah dengan melihat langsung hasil prediksi yang dihasilkan oleh masing-masing algoritma. Algoritma yang hendak dibandingkan yaitu.

- *KNN user-user cosine similarity*
- *SVD++*

Tampilan 5 baris pertama dari masing-masing tabel prediksi algoritma dapat dilihat pada Tabel 11. dan Tabel 12.

TABEL XI

5 BARIS PERTAMA PREDIKSI SVD++

	Uid	**Iid**	**Rui**	**Est**	***Details***	**Err**
0	344af62cf08ea5c4ea1eb554366d221c1431f4d3	SOYNVEE12AF72A49EE	1.0	1.032	{'was_impossible': False}	0.032
1	4e11f45d732f4861772b2906f81a7d384552ad12	SOKUQTM12A81C218C4	1.0	1.024	{'was_impossible': False}	0.024
2	08d31ac4452516e702815fef13b2059aa8210034	SOUYPYF12A58A76897	1.0	1.021	{'was_impossible': False}	0.021
3	dc28a894cfec1eba07b35b143d6c999fc7cf8e0f	SONHWUN12AC468C014	2.0	1.965	{'was_impossible': False}	0.034
4	2f05297f683c61f92ba4dd2810f4e950f467a2a0	SOVMIHC12A8C13703A	1.0	1.000	{'was_impossible': False}	0.000

TABEL XII

5 PREDIKSI *KNN USER-USER COSINE SIMILARITY*

	Uid	Iid	Rui	Est	*Details*	Err
0	344af62cf08ea5c4ea1eb554366d221c1431f4d3	SOYNVEE12AF72A49EE	1.0	1.000	{'actual_k': 2, 'was_impossible': False}	0.000
1	4e11f45d732f4861772b2906f81a7d384552ad12	SOKUQTM12A81C218C4	1.0	1.000	{'actual_k': 2, 'was_impossible': False}	0.000
2	08d31ac4452516e702815fef13b2059aa8210034	SOUYPYF12A58A76897	1.0	1.276	{'was_impossible': True , 'reason': 'Not enough.}	0.276
3	dc28a894cfec1eba07b35b143d6c999fc7cf8e0f	SONHWUN12AC468C014	2.0	2.000	{'actual_k': 26, 'was_impossible': False}	0.000
4	2f05297f683c61f92ba4dd2810f4e950f467a2a0	SOVMIHC12A8C13703A	1.0	1.276	{'was_impossible': True , 'reason': 'Not enough.}	0.276

Seperti yang terlihat pada Tabel 11 dan Tabel 12 masing-masing tabel prediksi memiliki 6 kolom yaitu:

- *uid* : kolom berisi id pengguna.
- *iid* : kolom berisi id *item* yang direkomendasikan pada pengguna.
- *rui* : kolom berisi nilai *rating* sebenarnya yang diberikan pada pengguna terhadap *item*.
- *est* : kolom berisi nilai *rating* hasil prediksi dari masing-masing algoritma
- *details* : kolom berisi penjelasan lebih detail mengenai sebab terjadinya *error* pada masing-masing prediksi.
- *err* : kolom berisi nilai *error* yang dihasilkan oleh masing-masing prediksi. Nilai *error* diperoleh dari selisih nilai *rating* sebenarnya dengan nilai *rating* hasil prediksi.

Selanjutnya membandingkan kedua algoritma tersebut dengan cara melihat 10 prediksi terburuk yang dihasilkan dari masing-masing algoritma sesuai yang terlihat pada Tabel 13 dan Tabel 14.

TABEL XIII

10 PREDIKSI TERBURUK ALGORITMA *KNN*

	Uid	Iid	Rui	Est	*Details*	Err
14749	a643df7fba29007b1e3a1206a6fa4c344dd7a20f	SOSNJIT12A8159E8DB	2.0	1.276	{'was_impossible': True , 'reason': 'Not enough *neighbors*.}	0.723
17062	6a3dcc268684dc7df0a3bd6c61f4a963aeac33fa	SOVTLQW12AB0186641	2.0	1.276	{'was_impossible': True , 'reason': 'Not enough *neighbors*.}	0.723
15657	53cf9e65b8418b64669351ab24324e1a31e02b08	SOHFJAQ12AB017E4AF	2.0	1.276	{'was_impossible': True , 'reason': 'Not enough *neighbors*.}	0.723
1924	e21477efb83bd323205ce6f5bd662f3df9d477e5	SOZFYTZ12AB018565D	2.0	1.276	{'was_impossible': True , 'reason': 'Not enough *neighbors*.}	0.723
7536	0fc22d4c05443a9e6b73b3cc9315ab48a9d65f08	SOCNAXF12A6D4F9B34	2.0	1.276	{'was_impossible': True , 'reason': 'Not enough *neighbors*.}	0.723

	Uid	Iid	Rui	Est	*Details*	Err
11975	470c65144c49b122c4be2ea008cba342b7af2a9d	SORHWSI12A6310E1FF	2.0	1.276	{'was_impossible': True , 'reason': 'Not enough *neighbors*.}	0.723
9143	a633999a259275ac6b9bbadce2f5223be4b84bc3	SOAZFQH12A8C13D101	2.0	1.276	{'was_impossible': True , 'reason': 'Not enough *neighbors*.}	0.723
13000	b40869597b8a672f1efedb7b50defb833ec3518b	SOIZAZL12A6701C53B	2.0	1.276	{'was_impossible': True , 'reason': 'Not enough *neighbors*.}	0.723
3378	ed7d4c476013b1c3dd91982b61494bf7436083ba	SOFRQTD12A81C233C0	3.0	1.276	{'was_impossible': True , 'reason': 'Not enough *neighbors*.}	1.723
17184	bb85bb79612e5373ac714fcd4469cabeb5ed94e1	SOZQSVB12A8C13C271	3.0	1.276	{'was_impossible': True , 'reason': 'Not enough *neighbors*.}	1.723

TABEL XIV

PREDIKSI TERBURUK SVD++

	Uid	Iid	Rui	Est	*Details*	Err
8818	9b417dda603eee8e66d3c9e84c1330459ac7d034	SOBONKR12A58A7A7E0	5.0	4.796	{'was_impossible': False}	0.203
9158	22264df0cf834cf3066af44104659d90fa87931f	SODXNYI12A8C13E031	1.0	1.204	{'was_impossible': False}	0.204
6736	35d354f4afef6a6b464862d132a60c5ae970ee21	SOYEJQY12AB018696B	1.0	1.204	{'was_impossible': False}	0.204
16965	36751d44579bd38d49749d1eb45f275393e60c7e	SOPREFD12AB0187C3A	1.0	1.204	{'was_impossible': False}	0.204
3038	8ed04296b2d7d886c16645dc647c125f73379f9f	SOFROOV12A67020901	1.0	1.210	{'was_impossible': False}	0.210
18346	d3fd2adbac66ca7e7527b563fde09d0b3a5546d9	SOJDNPX12A6310E10F	1.0	1.212.	{'was_impossible': False}	0.212
15617	260c7adab9c3a441b922d9466872d1b1f6710eff	SOAZFQH12A8C13D101	2.0	1.782	{'was_impossible': False}	0.217
9143	a633999a259275ac6b9bbadce2f5223be4b84bc3	SOAZFQH12A8C13D101	2.0	1.760	{'was_impossible': False}	0.239
17777	b1269307f2ae8c17062c6aea2502b099aad517b6	SOFZWTX12A8C138B1D	2.0	1.643	{'was_impossible': False}	0.356

Terlihat pada Tabel 13 dan Tabel 14 prediksi dari algoritma *SVD++* memiliki tingkat akurasi yang lebih baik jika dibandingkan dengan algoritma *KNN user-user cosine similarity*. Hal ini terbukti pada *error* yang dihasilkan oleh algoritma *SVD++* terhadap 10 prediksi terburuknya lebih kecil dari *error* yang dihasilkan oleh algoritma *KNN user-user cosine similarity.* Langkah selanjutnya adalah melakukan perbandingan dengan cara melihat prediksi algoritma yang dihasilkan

ketika algoritma lawan memiliki *error* dengan tingkat yang tinggi. Untuk lebih jelasnya dapat melihat Tabel 15 dan Tabel 16.

TABEL XV

TABEL PREDIKSI ALGORITMA *KNN USER-USER COSINE SIMILARITY* TERHADAP 10 PREDIKSI SVD++

	Uid	**Iid**	**Rui**	**Est**	***Details***	**Err**
8818	9b417dda603eee8e66d3c9e84c1330459ac7d034	SOBONKR12A58A7A7E0	5.0	5.000	{'actual_k': 21, 'was_impossible': False}	0.000
9158	22264df0cf834cf3066af44104659d90fa87931f	SODXNYI12A8C13E031	1.0	1.276	{'was_impossible': True, 'reason': 'User and/or *item* is unkown.'}	0.276
6736	35d354f4afef6a6b464862d132a60c5ae970ee21	SOYEJQY12AB018696B	1.0	1.276	{'was_impossible': True, 'reason': 'User and/or *item* is unkown.'}	0.276
16965	36751d44579bd38d49749d1eb45f275393e60c7e	SOPREFD12AB0187C3A	1.0	1.276	{'was_impossible': True, 'reason': 'User and/or *item* is unkown.'}	0.276
3038	8ed04296b2d7d886c16645dc647c125f73379f9f	SOFROOV12A67020901	1.0	1.276	{'was_impossible': True, 'reason': 'Not enough *neighbors*.'}	0.276
18346	d3fd2adbac66ca7e7527b563fde09d0b3a5546d9	SOJDNPX12A6310E10F	1.0	1.276	{'was_impossible': True, 'reason': 'User and/or *item* is unkown.'}	0.276
15617	260c7adab9c3a441b922d9466872d1b1f6710eff	SOAZFQH12A8C13D101	2.0	1.276	{'was_impossible': True, 'reason': 'Not enough *neighbors*.'}	0.723
9143	a633999a259275ac6b9bbadce2f5223be4b84bc3	SOAZFQH12A8C13D101	2.0	1.276	{'was_impossible': True, 'reason': 'Not enough *neighbors*.'}	0.723
17777	b1269307f2ae8c17062c6aea2502b099aad517b6	SOFZWTX12A8C138B1D	2.0	1.276	{'was_impossible': True, 'reason': 'Not enough *neighbors*.'}	0.723
18606	7ea016e249d4851792aa36d98db1b58c9674e592	SOFZWTX12A8C138B1D	2.0	1.276	{'was_impossible': True, 'reason': 'Not enough *neighbors*.'}	0.723

TABEL XVI

TABEL PREDIKSI ALGORITMA SVD++ TERHADAP 10 PREDIKSI TERBURUK *KNN USER-USER COSINE SIMILARITY*

	Uid	**Iid**	**Rui**	**Est**	***Details***	**Err**
14749	a643df7fba29007b1e3a1206a6fa4c344dd7a20f	SOSNJIT12A8159E8DB	2.0	1.966	{'was_impossible': False}	0.033
17062	6a3dcc268684dc7df0a3bd6c61f4a963aeac33fa	SOVTLQW12AB0186641	2.0	1.973	{'was_impossible': False}	0.026

	Uid	Iid	Rui	Est	*Details*	Err
15657	53cf9e65b8418b64669351ab24324e1a31e02b08	SOHFJAQ12AB017E4AF	2.0	1.947	{'was_impossible': False}	0.052
1924	e21477efb83bd323205ce6f5bd662f3df9d477e5	SOZFYTZ12AB018565D	2.0	1.946	{'was_impossible': False}	0.053
7536	0fc22d4c05443a9e6b73b3cc9315ab48a9d65f08	SOCNAXF12A6D4F9B34	2.0	2.001	{'was_impossible': False}	0.001
11975	470c65144c49b122c4be2ea008cba342b7af2a9d	SORHWSI12A6310E1FF	2.0	1.973	{'was_impossible': False}	0.062
9143	a633999a259275ac6b9bbadce2f5223be4b84bc3	SOAZFQH12A8C13D101	2.0	1.760	{'was_impossible': False}	0.239
13000	b40869597b8a672f1efedb7b50defb833ec3518b	SOIZAZL12A6701C53B	2.0	1.988	{'was_impossible': False}	0.011
3378	ed7d4c476013b1c3dd91982b61494bf7436083ba	SOFRQTD12A81C233C0	3.0	2.937	{'was_impossible': False}	0.062
17184	bb85bb79612e5373ac714fcd4469cabeb5ed94e1	SOZQSVB12A8C13C271	3.0	2.843	{'was_impossible': False}	0.156

Dapat dilihat pada Tabel 15 dan Tabel 16 bahwa algoritma *SVD++* lebih baik dari algoritma *KNN user-user cosine similarity*. Hal ini terlihat bahwa algoritma *SVD++* dapat menghasilkan prediksi dengan tingkat *error* yang lebih rendah terhadap 10 prediksi terburuk dari *KNN user-user cosine similarity*.

VI. KESIMPULAN

A. Simpulan

Dalam penelitian ini dilakukan uji coba algoritma sistem rekomendasi musik dengan menggunakan metode collaborative filtering. Beberapa poin yang dapat disimpulkan dari hasil penelitian ini adalah.

1. *Library surprise* dapat digunakan untuk melakukan implementasi sistem rekomendasi. Tidak hanya sekedar sebagai *library* penyedia model-model algoritma sistem rekomendasi berbasis *collaborative filtering* yang siap pakai, namun juga dapat melakukan implementasi sistem rekomendasi dengan alur yang lengkap dan utuh.
2. Dari sisi menghasilkan prediksi yang akurat algoritma *SVD++* memiliki kinerja yang paling baik daripada metode algoritma *collaborative filtering* lainnya. Karena perolehan nilai *RMSE* dan *MAE SVD++* paling kecil daripada algoritma *collaborative filtering* lainnya. Namun dari sisi memberikan *item* yang relevan dan *recommended*. Algoritma *KNN* memiliki kinerja lebih baik daripada metode *matrix factorization*. Karena nilai *precision and recall* yang dihasilkan *KNN* lebih tinggi bila dibandingkan dengan *matrix factorization*.
3. Banyaknya parameter yang terlibat dan dapat dikonfigurasi pada algoritma sistem rekomendasi sangat berpengaruh dalam menghasilkan prediksi yang lebih akurat. Faktor dataset yang digunakan juga dapat menjadi salah satu faktor utama dalam menentukan performa kinerja algoritma.

B. Saran

Saran untuk perbaikan penelitian ini kedepanya adalah :

1. Melakukan perbandingan kinerja algoritma ketika algoritma sistem rekomendasi diterapkan pada kehidupan sehari-hari untuk mendapatkan feedback dari pengguna secara langsung.
2. Faktor pemilihan dataset perlu juga diperhatikan, sehingga algoritma sistem rekomendasi dapat memberikan performa dan kinerja terbaiknya.

DAFTAR PUSTAKA

[1] Rounak Banik, *Hands-On Recommendation Systems with Python*, 1st ed., Pravin Dhadre, Ed. Brimingham, England: Packt Publishing, 2018.J. S.
[2] Karolina Grubinska & Médéric Thomas. (2019, July) Correcting for Self-selection in Product *Rating*: Causal Recommender Systems. [Tersedia]. https://humboldt-wi.github.io/blog/research/applied_predictive_modeling_19/causalrecommendersystem/
[3] Bapiste Rocca. (2019, Juny) Introduction To Recommender System. [Tersedia]. https://towardsdatascience.com/introduction-to-recommender-systems-6c66cf15ada
[4] Prince Gover. (2017, Desember) Various Implementations of *Collaborative filtering*. [Tersedia]. https://towardsdatascience.com/various-implementations-of-collaborative-filtering-100385c6dfe0 [Diakses 6 *MAE*et 2020].
[5] Rabin Poudyal. (2018, Juni) Nearest neighbour based method for *collaborative filtering*. [Tersedia]. https://medium.com/@rabinpoudyal1995/nearest-neighbour-based-method-for-collaborative-filtering-16961c962dd
[6] Oktay Baheci. (2017, September) Recommendation Deconstructed: *Collaborative filtering*. [Tersedia]. https://medium.com/@oktaybahceci/recommendation-deconstructed-collaborative-filtering-cc02357a6106
[7] Surprise Python *Library*. [Tersedia]. https://surprise.readthedocs.io/en/stable/
[8] Salma Elsahahawy. (2017, January) Understanding *Matrix factorization* for recommender systems. [Tersedia]. https://towardsdatascience.com/understanding-matrix-factorization-for-recommender-systems-4d3c5e67f2c9
[9] Sachin Prabhu Thandapani. (2019, *MAE*et) *Recommendation System*s: *Collaborative filtering* using *Matrix factorization* — Simplified. [Tersedia]. https://medium.com/sfu-cspmp/recommendation-systems-collaborative-filtering-using-matrix-factorization-simplified-2118f4ef2cd3
[10] Paritosh Pantola. (2018, Juni) Recommendation Using *Matrix factorization*. [Tersedia]. https://medium.com/@paritosh_30025/recommendation-using-matrix-factorization-5223a8ee1f4
[11] Yehuda Koren. (2008, Agustus) Factorization Meets the Neighborhood: a Multifaceted. [Tersedia]. http://www.cs.rochester.edu/twiki/pub/Main/HarpSeminar/Factorization_Meets_the_Neighborhood-_a_Multifaceted_Collaborative_Filtering_Model.pdf.
[12] D.W. Oard and J. Kim, *Implicit Feedback for Recommender System*. America: Proc. 5th DELOS Workshop on, 1998.
[13] Piotr Gabrys. (2018, November) Non-negative *matrix factorization* for *Recommendation System*s. [Tersedia]. https://medium.com/logicai/non-negative-matrix-factorization-for-recommendation-systems-985ca8d5c16c.
[14] Kirill Bondarenko. (2019, Februari) *Precision* and *recall* in recommender systems. And some metrics stuff. [Tersedia]. https://medium.com/@bond.kirill.alexandrovich/*precision*-and-*recall*-in-recommender-systems-and-some-metrics-stuff-ca2ad385c5f8
[15] JJ. (2016, *MAE*et) *MAE* and *RMSE* — Which Metric is Better? [Tersedia]. https://medium.com/human-in-a-machine-world/*MAE*-and-*RMSE*-which-metric-is-better-e60ac3bde13d
[16] Maher Maleb. (2017, Agustus) *Recall* and *Precision* at k for Recommender Systems. [Tersedia]. https://medium.com/@m_n_malaeb/*recall*-and-*precision*-at-k-for-recommender-systems-618483226c54
[17] Krishni. (2018, Desember) K-*Fold* Cross Validation. [Tersedia]. https://medium.com/datadriveninvestor/k-*fold*-cross-validation-6b8518070833
[18] Brian Srebrenik. (2018, December) Introduction to Music Recommendation and Machine Learning. [Tersedia]. https://medium.com/@briansrebrenik/introduction-to-music-recommendation-and-machine-learning-310c4841b01d