TUGAS AKHIR – IF184802

PEMETAAN EMOSI PADA LAGU BERDASARKAN RUSSEL'S TWO-DIMENSIONAL MODEL MENGGUNAKAN GAUSSIAN PROCESS

TEGAR SATRIO UTOMO
NRP 05111540000178

Dosen Pembimbing I
Dr. Agus Zainal Arifin, S.Kom., M.Kom.

Dosen Pembimbing II
Dr.Eng Nanik Suciati, S.Kom., M.Kom.

Departemen Informatika
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember
Surabaya 2019

TUGAS AKHIR – IF184802

PEMETAAN EMOSI PADA LAGU BERDASARKAN RUSSEL'S TWO-DIMENSIONAL MODEL MENGGUNAKAN GAUSSIAN PROCESS

TEGAR SATRIO UTOMO
NRP 05111540000178

Dosen Pembimbing I
Dr. Agus Zainal Arifin, S.Kom., M.Kom.

Dosen Pembimbing II
Dr.Eng Nanik Suciati, S.Kom., M.Kom.

Departemen Informatika
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember
Surabaya 2019

(Halaman ini sengaja dikosongkan)

UNDERGRADUATE THESIS – IF184802

EMOTION MAPPING IN SONG BASED ON RUSSEL TWO-DIMENSION MODEL USING GAUSSIAN PROCESS

TEGAR SATRIO UTOMO
NRP 05111540000178

First Advisor
Dr. Agus Zainal Arifin, S.Kom., M.Kom.

Second Advisor
Dr.Eng Nanik Suciati, S.Kom., M.Kom.

Department of Informatics
Faculty of Information and Communication Technology
Institut Teknologi Sepuluh Nopember
Surabaya 2019

(Halaman ini sengaja dikosongkan)

LEMBAR PENGESAHAN

PEMETAAN EMOSI PADA LAGU BERDASARKAN RUSSEL'S TWO-DIMENSIONAL MODEL MENGGUNAKAN GAUSSIAN PROCESS

TUGAS AKHIR

Diajukan Untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Bidang Studi Komputasi Cerdas dan Visi
Program Studi S-1 Departemen Informatika
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember

Oleh:
TEGAR SATRIO UTOMO
NRP: 05111540000178

Disetujui oleh Pembimbing Tugas Akhir:

1. Dr. Agus Zainal Arifin, S.Kom., M.Kom. .....
(NIP. 19720809 199512 1 001) (Pembimbing 1)

2. Dr.Eng Nanik Suciati, S.Kom., M.Kom .....
(NIP. 19710428 199412 2 001) (Pembimbing 2)

SURABAYA
Juni, 2019

(Halaman ini sengaja dikosongkan)

PEMETAAN EMOSI PADA LAGU BERDASARKAN RUSSEL'S TWO-DIMENSIONAL MODEL MENGGUNAKAN GAUSSIAN PROCESS

Nama Mahasiswa	**: Tegar Satrio Utomo**
NRP	**: 05111540000178**
Jurusan	**: Informatika, FTIK-ITS**
Dosen Pembimbing 1	**: Dr. Agus Zainal Arifin, S.Kom., M.Kom.**
Dosen Pembimbing 2	**: Dr.Eng Nanik Suciati, S.Kom., M.Kom.**

ABSTRAK

Dalam Music information retriaval, terdapat beberapa hal yang lazim dilalukan seperti artist identification, instrument recognition dan emotion recognition. Salah satu pengaplikasian Music information retriaval adalah emotion recognition. Emotion recognition pada musik dapat dilakukan salah satunya dengan memanfaatkan model Russell, yaitu sebuah bidang dalam sumbu x (valence) dan y (arousal) yang membagi emosi ke dalam model dimensional representation yang memetakan emosi kedalam cluster - cluster yang merepresentasikan kelompok emosi yang memiliki sifat sama. Pada tugas akhir ini akan dicari nilai valence - arousal untuk model Russell yang merepresentasikan emosi dari suatu lagu dengan gaussian process regression. Data pelatihan dan uji coba diambil dari dataset MediaEval'2013 yang merupakan data yang berisi lagu - lagu bebas hak cipta dan memiliki anotasi valence (sumbu x) dan arousal (sumbu y) yang nantinya menjadi tujuan pemetaan lagu pada bidang emosi. Akan dibuat dua model yang digunakan untuk memprediksi nilai untuk valence (sumbu x) dan arousal (sumbu y). Setiap model akan melalui tahap Preprocessing, antara lain penyeragaman sampling rate, serta perubahan format lagu menjadi wav, yang kemudian akan diambil fitur fitur seperti mel frequency cepstral

coefficients, timbre features, spectral crest factor dan spectral flatness measure, chromagram. Akan digunakan metode Gaussian process regressor dengan fungsi kernel Rational Quadratic Hasil uji coba terakhir didapatkan nilai R2 0.68 untuk model arousal dan 0.38 untuk model valence, serta akurasi pemetaan sebesar 63%.
Kata kunci: ***Dataset MediaEval'2013, Emotion Recognition, Gaussian Process Regressio,. Music Information Retriaval, Model Russell*.**

EMOTION MAPPING IN SONG BASED ON RUSSEL TWO-DIMENSION MODEL USING GAUSSIAN PROCCESS

Student's Name : Tegar Satrio Utomo
Student's ID : 05111540000178
Department : Informatics, Faculty of ICT-ITS
First Advisor : Dr. Agus Zainal Arifin, S.Kom., M.Kom.
Second Advisor : Dr.Eng Nanik Suciati, S.Kom., M.Kom.

ABSTRACT

In Music information retriaval, there are some things that are commonly practiced such as artist identification, instrument recognition and emotion recognition. One application of Music information retriaval is emotion recognition. Emotion recognition in music can be done by using the Russell model, which is a field in the valence (x axis) and arousal (y axis) that divides emotions into dimensional representation models that map emotions into clusters - clusters that represent groups of emotions that have the same nature . In this final project, the value of valence - arousal for the Russell model that represents the emotion of a song with gaussian process regression will be sought. Training data and trials were taken from the MediaEval '2013 dataset which is data that contains copyright-free song songs and has an annotation of the x-axis and y-axis which later becomes the goal of song mapping in the emotional field. Two models will be used to predict values for the valence (x axis) and arousal (y axis). Each model will go through the Preprocessing stage, including sampling rate uniformity, and changes to the song format to wav, which will then take features such as mel frequency cepstral coefficients, timbre features, spectral crest factor and spectral flatness measure, chromagram. Gaussian process method will be used regressor with Rational Quadratic kernel function The results of the last trial obtained R2 0.68 for the arousal model and 0.38 for the valence model, and 63% for mapping accuracy

Keywords: ***Dataset MediaEval'2013, Emotion Recognition, Gaussian Process Regressio,. Music Information Retriaval, Model Russell.***

KATA PENGANTAR

Puji syukur saya sampaikan kepada Tuhan yang Maha Esa karena berkat rahmat-Nya saya dapat melaksanakan Tugas Akhir yang berjudul:

"**PEMETAAN EMOSI PADA LAGU BERDASARKAN RUSSEL'S TWO-DIMENSIONAL MODEL MENGGUNAKAN GAUSSIAN PROCESS**"

Terselesaikannya Tugas Akhir ini tidak terlepas dari bantuan dan dukungan banyak pihak, oleh karena itu melalui lembar ini penulis ingin mengucapkan terima kasih dan penghormatan kepada:

1. Allah SWT, karena limpahan rahmat dan karunia-Nya penulis dapat menyelesaikan Tugas Akhir dan juga perkuliahan di Informatika ITS.
2. Kedua orangtua penulis, dan anggota keluarga lainnya yang telah memberikan dukungan doa, moral, dan material kepada penulis sehingga penulis dapat menyelesaikan Tugas Akhir ini.
3. Dr. Agus Zainal Arifin, S.Kom., M.Kom. dan Dr.Eng Nanik Suciati, S.Kom., M.Kom. selaku pembimbing I dan II yang telah membimbing dan memberikan motivasi, nasihat dan bimbingan dalam menyelesaikan Tugas Akhir ini.
4. Dr. Eng. Radityo Anggoro, S.Kom., M.Sc. selaku Kepala Program Studi Informatika ITS dan seluruh dosen dan karyawan Departemen Informatika ITS yang telah memberikan ilmu dan pengalaman kepada penulis selama menjalani masa kuliah di Informatika ITS.
5. Keluarga Admin Hima X Warkop Wardug yang memberi semangat, dan menjadi tempat bercakap ketika nongkrong.

6. Keluarga Dagri Optimasi HMTC yang telah menemani penulis belajar bersama menempa diri.
7. Seluruh mahasiswa Informatika ITS angkatan 2015 yang telah menjadi teman penulis selama menjalani masa kuliah di Informatika ITS.
8. Semua musisi music metal dalam negeri maupun luar negeri.
9. Untuk Pentil, Aqil, Fajar, Rezky, Nopal, Djohan, Nahda, Huda. Selamat dan Sukses.
10. Serta semua pihak yang yang telah turut membantu penulis dalam menyelesaikan Tugas Akhir ini.

Penulis menyadari bahwa laporan Tugas Akhir ini masih memiliki banyak kekurangan. Oleh karena itu dengan segala kerendahan hati penulis mengharapkan kritik dan saran dari pembaca untuk perbaikan penulis kedepannya. Selain itu, penulis berharap laporan Tugas Akhir ini dapat berguna bagi pembaca secara umum.

Surabaya, Mei 2019

DAFTAR ISI

DAFTAR TABEL

DAFTAR KODE SUMBER

DAFTAR GAMBAR

BAB I
PENDAHULUAN

1.1 Latar Belakang

Dalam *music information retriaval*, terdapat beberapa hal yang lazim dilalukan seperti *artist identification*, *instrument recognition* dan *emotion recognition*. Kunci utama dari Music Information Retriaval adalah *feature extraction*. Kami fokus pada *music information retriaval* berdasarkan sinyal audio. Dari perspektif pemahaman musik, kita dapat membagi fitur audio menjadi tiga level,*low-level feature* dan *mid-level feature* dan *top level feature* [1].

Salah satu pengaplikasian *music information retriaval* adalah *emotion recognition*. *Emotion recognition* adalah pengenalan emosi dari sebuah lagu [2]. Manusia dapat menikmati musik karena musik dapat menimbulkan suatu perubahan emosi pada pendengar nya [3] *Emotion recognition* pada musik dapat dilakukan salah satunya dengan memanfaatkan model Russell *circumplex model of affect* [4], yaitu sebuah bidang dalam sumbu x (valence) dan y (arousal) yang membagi emosi ke dalam model representasi dimensional yang memetakan emosi kedalam *cluster-cluster* yang merepresentasikan kelompok emosi yang memiliki sifat sama.

Gaussian process regressor memiliki persamaan dengan *support vector machine* yang merupakan *state-of-the-art* dalam bidang *music information retrieval* karena keduanya menggunakan fungsi *kernel* [5]. Pada tugas akhir ini akan dicari nilai *Valence-Arousal* pada model Russell [4] yang merepresentasikan emosi dari suatu lagu dengan *Gaussian process regression*.. Hasil prediksi akan membentuk titik koordinat sumbu x (valence) dan sumbu y (arousal) [2] sebagai representasi *cluster* emosi dalam model Russell [4]. Data pelatihan dan uji coba diambil dari dataset MediaEval'2013, dataset ini merupakan data yang berisi lagu - lagu bebas hak cipta, memiliki anotasi sumbu x dan sumbu y yang dilabeli

manual, dan nantinya menjadi tujuan *emotion recognition* yang dibuat.

1.2 Rumusan Masalah

Rumusan masalah yang diangkat dalam Tugas Akhir ini adalah sebagai berikut:

1. Bagaimana fitur ekstraksi pada dataset berupa audio?
2. Bagaimana mencari nilai *Valence-Arousal* yang paling mendekati nilai yang merepresentasikan emosi?
3. Bagaimana Gaussian Processes untuk pemetaan emosi pada lagu?

1.3 Batasan Permasalahan

Permasalahan yang dibahas pada Tugas Akhir ini memiliki beberapa batasan, yaitu sebagai berikut:

1. Dataset yang digunakan berasal dari dataset MediaEval'2013 [6]
2. Emosi yang dikenali adalah emosi yang berada dalam *Russell's Two-Dimensional Model* [4]
3. Implementasi program menggunakan bahasa pemrograman *Python 3*
4. Fitur yang digunakan adalah fitur audio dari lagu.

1.4 Tujuan

Tujuan dari pembuatan tugas akhir ini adalah untuk mencari emosi sebuah lagu berdasarkan *Russell's two-dimensional model* menggunakan *Gaussian Processes*.

1.5 Manfaat

Tugas akhir ini diharapkan meningkatkan hubungan manusia dengan sebuah lagu, dengan rekomendasi sesuai emosi, penghayatan sebuah lagu sesuai emosi, serta kebutuhan-kebutuhan lainnya.

1.6 Metodologi

Pembuatan Tugas Akhir ini dilakukan dengan menggunakan metodologi sebagai berikut:

1.6.1 Penyusunan Proposal Tugas Akhir

Tahapan awal dari Tugas Akhir ini adalah penyusunan proposal Tugas Akhir yang berisi pendahuluan, deskripsi dan gagasan metode-metode yang dibuat dalam Tugas Akhir ini. Pendahuluan ini terdiri dari latar belakang diajukannya Tugas Akhir, rumusan masalah dan batasan masalah yang ditetapkan, serta manfaat dari hasil pembuatan Tugas Akhir ini. Selain itu, dijabarkan pula tinjauan pustaka yang digunakan sebagai referensi pendukung pembuatan Tugas Akhir. Terdapat pula sub bab jadwal kegiatan yang menjelaskan jadwal pengerjaan Tugas Akhir.

1.6.2 Studi Literatur

Pada tahap ini dilakukan pencarian literatur berupa jurnal yang digunakan sebagai referensi untuk pengerjaan tugas akhir ini. Literatur yang dipelajari pada pengerjaan tugas akhir ini berasal dari jurnal ilmiah yang diambil dari berbagai sumber di internet, beserta berbagai literatur online tambahan terkait *Music information retreval*, model Russell, *Gaussian process regression*.

1.6.3 Implementasi Perangkat Lunak

Pada tahap ini akan dilaksanakan implementasi metode dan algoritma yang telah direncanakan. Implementasi sistem menggunakan *Python 3* sebagai bahasa pemrograman, serta *library* pendukung lainya.

1.6.4 Pengujian dan Evaluasi

Tahap pengujian dan evaluasi dilakukan menggunakan dataset MediaEval’2013 untuk mengetahui hasil dan performa arsitektur yang telah dibangun. Evaluasi dilakukan dengan

metode pengukuran R^2 Untuk model, dan Akan dicari Nilai Akurasi, *precision, recall* untuk pemetaan emosi.

1.6.5 Penyusunan Buku

Pada tahap ini dilakukan penyusunan buku yang menjelaskan seluruh konsep, teori dasar dari metode yang digunakan, implementasi, serta hasil yang telah dikerjakan sebagai dokumentasi dari pelaksanaan Tugas Akhir.

1.7 Sistematika Penulisan Laporan

Sistematika penulisan laporan Tugas Akhir adalah sebagai berikut:

Bab I Pendahuluan

Bab ini berisikan penjelasan mengenai latar belakang, rumusan masalah, batasan masalah, tujuan, manfaat, metodologi, dan sistematika penulisan dari pembuatan Tugas Akhir.

Bab II Tinjauan Pustaka

Bab ini berisi kajian teori dari metode dan algoritma yang digunakan dalam penyusunan Tugas Akhir ini. Secara garis besar, bab ini berisi tentang *Music information retreval*, model Russell, *Gaussian process regression*.

Bab III Perancangan Sistem

Bab ini berisi pembahasan mengenai perancangan dari *feature extraction* dan metode *Gaussian process regression* yang digunakan untuk *emotion recognition* pada lagu.

Bab IV Implementasi

Bab ini membahas implementasi dari perancangan yang telah dibuat pada bab sebelumnya. Penjelasan berupa kode yang digunakan untuk proses implementasi.

Bab V Uji Coba Dan Evaluasi

Bab ini membahas tahapan uji coba, kemudian hasil uji coba dievaluasi terhadap kinerja dari sistem yang dibangun.

Bab VI Kesimpulan dan Saran

Bab ini merupakan bab yang menyampaikan kesimpulan dari hasil uji coba yang dilakukan, masalah-masalah yang dialami pada proses dan tertulis saat pengerjaan Tugas Akhir, dan saran untuk pengembangan solusi ke depannya.

(Halaman ini sengaja dikosongkan)

BAB II
TINJAUAN PUSTAKA

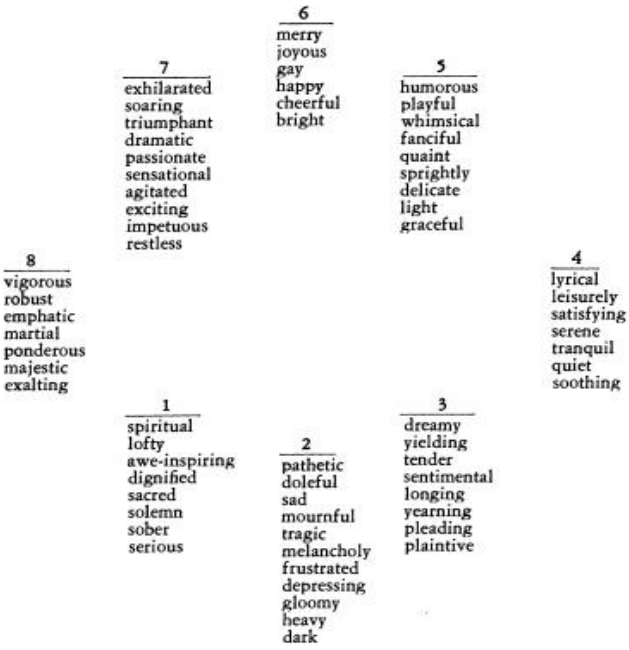
Bab ini membahas mengenai teori-teori dasar yang digunakan dalam Tugas Akhir. Penjelasan ini bertujuan untuk memberikan gambaran umum dan diharapkan dapat mendukung sistem yang dibangun.

2.1 Music Information Retrieval

Music Information Retrieval adalah pengambilan informasi dari sebuah musik mencakup beberapa bidang, seperti *artist recognition, genre classification,* dan *emotion recognition*, Informasi tersebut berasal dari fitur fitur yang diekstraksi sehingga dapat diolah menjadi informasi. Fitur fitur yang biasa digunakan salah satunya fitur audio. Setidaknya terdapat 3 level fitur audio, yaitu *low level, mid level* serta *top level*, dimana *low level* merupakan fitur yang didapatkan langsung dari teknik pemrosesan sinyal. *Mid level* adalah fitur yang berisikan tiga kelas fitur, yaitu ritme, nada dan harmony. Sedangkan *Top Level* adalah fitur yang diberikan manusia sebagai pemahaman mereka tentang musik [1].

2.2 Music Emotion Recognition

Manusia dapat menikmati musik karena musik dapat menimbulkan suatu perubahan emosi pada pendengar nya [3].
Merupakan salah satu bidang tugas dalam *music information retrieval* yang mengelompokkan musik kedalam bidang emosi tertentu. Fitur yang digunakan dapat berupa teks (lirik lagu), audio (lagu), video (*clip music*) ataupun gabungan dari ketiga nya [3]. Representasi dari emosi dapat dibedakan menjadi dua, *categorical representation* dan *dimensional representation*. *Categorical representation* membagi emosi kedalam kategori, dimana setiap emosi diberi label dengan beberapa kata sifat [3].



Gambar 2.1 Kata Sifat dan *Cluster* Oleh Hevners [3]

Gambar 2.1 menunjukkan pengelompokkan emosi berdasarkan kemiripan kata sifat. Sementara *dimensional representation* emosi dibagi berdasarkan sumbu, salah satunya model Russell [4].

2.3 Audio File Format

Audio file format adalah format file untuk menyimpan data audio digital pada sistem komputer. Data dapat berupa *bitstream* mentah dalam *audio coding format*, tetapi biasanya tertanam dalam format wadah atau format data audio dengan *layer* penyimpanan yang ditentukan. Setidaknya terdapat tiga grup besar dari tipe format dari file audio, *uncompressed audio format, lossless compressed audio format, lossy compressed audio*

format. Uncompressed audio format adalah format yang biasa diterima perangkat keras konversi digital-analog. *Uncompressed audio format* menulis suara dan "kesunyian" dalam jumlah bit yang sama per unit waktu. *Lossless compressed audio format* menyimpan data dalam ruang lebih sedikit tanpa kehilangan informasi. Pada *lossless compressef audio* musik akan berukuran lebih kecil dan "kesunyian" hamper tidak memakai tempat. *Lossy compressed audio* memungkinkan pengurangan ukuran file yang lebih besar dengan menghapus beberapa informasi audio dan menyederhanakan data. Ini, tentu saja, menghasilkan pengurangan dalam kualitas audio. MP3 adalah contoh paling popular dari grup ini.

2.3.1 WAV

Termasuk dalam grup tipe format *uncompressed audio format,* adalah audio file format standart dari Microsoft dan IBM untuk menyimpan audio *bitstream.* Di desain untuk menampung berbagai macam format audio, lossless dan lossy. WAV akan berisikan tambahan header kecil berisi metadata.

2.4 Short-Time Fourier Transform

Analisis *spectrum* biasanya dilakukan pada sebuah segment pendek dari sinyal audio (biasa disebut frames) agar dapat menangkap variasi pada konten frekuensi sepanjang waktu. Hal ini dapat dilakukan dengan mengalikan sinyal diskrit (Discrete Fourier Transform) dengan sebuah fungsi window, yang biasanya berbentuk lonceng dan memiliki nilai nol di luar interval yang dimaksud [1]. Penggunaan STFT sangat berguna untuk mengekstrak informasi pada rentang waktu pendek pada sebuah sinyal sepanjang waktu karena sinyal audio memiliki *spectrum* yang berubah sepanjang waktu. Komputasi Fourier Transform dari sebuah sinyal audio juga dapat membutuhkan waktu yang sangat lama dan sangat jarang sekali dapat dimasukkan ke dalam memori komputer sekaligus [7].

2.5 Low-Level Audio Features

Dalam perspektif pemahaman musik, *low-level features* adalah fitur fitur yang dihasilkan langsung dari proses proses pemrosesan sinyal. Digunakan dalam tugas–tugas pengklasifikasian karena performa yang baik [1]. *Low-level features* tidak dapat langsung dimengerti manusia, tidak seperti level lainnya yaitu *mid-level* ataupun *top-level feature* yang lebih dekat dengan manusia [1].

2.5.1 Mel Frequency Cepstral Coefficient

Fitur yang biasanya digunakan untuk *speech recognition* [5]. Digunakan untuk mewakili audio dengan cara yang meniru sifat fisiologis sistem pendengaran manusia [1].

Pembentukan fitur *mel frequency cepstral coefficient* dapat melalui

1. Menghitung transformasi *fourier* dari setiap *window*
2. *Triangular overlapping windows* digunakan untuk memetakan *spectrum* yang diperoleh diatas *mel-scale*
3. Mengambil log dari *mel-frequency.*
4. Ambil *Discrete Cosine Transform* dari *mel-log*
5. *MFCC* adalah amplitudo dari *spectrum* yang dihasilkan

Atau melalui Persamaan **(2.1)**,

$$v^3 MFCC = \sum_{k'=1}^{K'} \log(|X'(k',n)|) \cdot \cos\left(j.\left(k' - \frac{1}{2}\right)\frac{\pi}{K'}\right) \qquad \textbf{(2.1)}$$

Dimana $|X'(k',n)|$ adalah *mel-warped magnitude spectrum* dari blok sinyal, K adalah ukuran blok sinyal.

2.5.2 Fitur Timbre

Fitur Timbre menangkap kualitas tonal suara yang terkait dengan instrumentasi yang berbeda [1]. Merupakan representasi

dari satu set fitur skalar yang terdiri dari subfitur seperti *spectral centroid, spectral flux, spectral rolloff, and zero crossing rate* [5].

2.5.2.1 Spectral Centroid

Ukuran yang digunakan untuk mengkarakterisasasi suatu *spectrum*, Itu menunjukkan dimana *centre of mass* dari *spectrum*, dihitung dari *mean* dari frekuensi sinyal yang ditentukan oleh transformasi *fourier* dengan *magnitude* sebagai bobotnya, Nilai centroid yang lebih tinggi menunjukkan frekuensi yang lebih tinggi. [1]. Dapat ditulis dalam Persamaan **(2.2)**,

$$v_{SC}(n) = \frac{\sum_{k=0}^{\frac{k}{2}-1} k|X(k,n)|^2}{\sum_{k=0}^{\frac{k}{2}-1} |X(k,n)|^2} \quad \textbf{(2.2)}$$

Pada Persamaan **(2.2)**, $X(k)$ adalah amplitude yang terkait dengan *bin k* dalam *spectrum* DFT. Dalam praktiknya, *Spectral Centroid* menemukan frekuensi untuk frame yang diberikan, dan kemudian menemukan *bin* terdekat untuk frekuensi tersebut.

2.5.2.2 Spectral Flux

Spectral flux adalah ukuran seberapa cepat *power spectrum* suatu sinyal berubah, dihitung dengan membandingkan *power spectrum* dari *frame* satu dengan *frame* sebelumnya, atau biasanya dihitung dengan *euclidean distance* [8]. Dapat dirumuskan seperti persamaan **(2.3)**,

$$v_{SF}(n) = \sqrt{\sum_{k=0}^{\frac{k}{2}-1} (|X(k,n)| - |X(k,n-1)|)^2} \quad \textbf{(2.3)}$$

Hasil *spectral flux* adalah nilai dalam kisaran $0 \leq v_{SF}(n) \leq$ A, dengan A representasi dari kemungkinan maksimal dari *spectral amplitude* [8].

2.5.2.3 Spectral Rolloff

Spectral rolloff adalah pengukuran *bandwidth* dari *frame* audio. Didefinisikan penyimpanan frekuensi yang diakumulasi dari *magnitude* dari STFT [1]. Dapat dinotasikan dalam Persamaan **(2.4)**,

$$vSR = i \left| \sum_{k=0}^{i} |X(k,n)| = k. \sum_{k=0}^{\frac{K}{2}-1} |X(k,n)| \right. \quad \textbf{(2.4)}$$

Dengan nilai standart k adalah 0.85 atau 0.95[1].

2.5.2.4 Zero Crossing Rate

Low-Level Feature yang telah digunakan selama beberapa dekade dalam analisis bicara dan audio, menghitung jumlah perubahan dari *frame* audio [8]. Dapat dihitung melalui persamaan **(2.5)**,

$$Z_i = \frac{1}{2} \sum_{n}^{N} |sign(x[n]) - sign(x[n-1])| \quad \textbf{(2.5)}$$

$$\text{Dengan } sign(x[k]) = \begin{cases} 1, & \text{If } x(i) > 0 \\ 0, & \text{If } x(i) = 0 \\ -1, & \text{If } x(i) < 0 \end{cases} \quad \textbf{(2.6)}$$

Zero Crossing dihitung dengan menghitung berapa kali sinyal domain waktu memotong nol dalam *frame* yang diberikan [1]., nilai $sign$ disyaratkan seperti pada Persamaan **(2.6)**.

2.5.3 Spectral Crest Factor

Pengukuran nada sederhana yang membandingkan *magnitude spectrum* tertinggi dengan jumlah *magnitude spectrum*. Dapat dinotasi kan dalam Persamaan **(2.7)**,

$$v_{Tsc}(n) = \frac{\max_k |X(k,n)|}{\sum_{k=0}^{\frac{k}{2}-1} |X(k,n)|} \quad \textbf{(2.7)}$$

Hasil rendah menunjukkan *spectrum magnitude* datar dan hasil tinggi menunjukkan sinusoidal [7].

2.5.4 Spectral Flatness Measure

Spectral Flatness adalah rasio dari *geometric mean* dan *arithmetic mean* dari *magnitude spectrum.* yang selanjutnya digunakan *arithmetic mean* dari log *magnitude spectrum* dalam pembilang untuk menghindari masalah akurasi komputasi [7], didefinisikan dengan Persamaan **(2.8)**,

$$vTf(n) = \frac{\exp(\frac{2}{k} . \sum_{k=0}^{\frac{k}{2}-1} \log(|X(k,n)|))}{\frac{2}{k} . \sum_{k=0}^{\frac{k}{2}-1} |X(k,n)|} \quad \textbf{(2.8)}$$

2.5.5 Chromagram

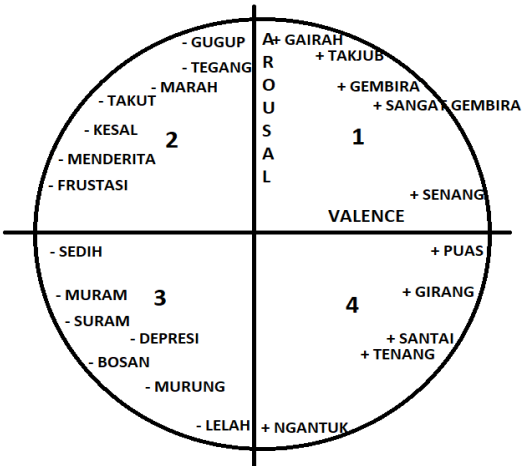
Chromagram adalah histogram dengan 12 dimensi yang setiap dimensinya merepresentasikan satu *pitch class* seperti C,C#,dll [7]. Fitur ini mewakili distribusi *spectrum* dari *semitone* yang berbeda dan menyediakan informasi tentang kunci lagu [5].

. Cara paling sederhana untuk mengekstraksi *Chromagram* dengan menjumlahkan besar STFT disetiap *semitone band* dengan batas indeks k_1, k_u, dan hasil dalam setiap oktaf *o* ditambahkan ke dalam entri yang sesuai dengan indeks *pitch class* [7], dapat ditulis ke dalam Persamaan **(2.9)**,

$$v(j,n) = \sum_{o=o_i}^{o_u} \left(\frac{1}{k_u(o,j) - k_1(o,j) + 1} \sum_{k=k_1(o,j)}^{k_u(o,j)} |X(k,n)| \right) \quad \textbf{(2.9)}$$

2.6 Model Russell

Suatu permodelan dalam pengelompokan emosi yang mendistribusikan emosi yang berasal dari kata sifat yang merepresentasikan emosi tersebut kedalam ruang dua dimensi sirkular. Bidang dipisahkan oleh nilai *Arousal* (sumbu y) dan *Valence* (sumbu x) [4]. Seperti pada Gambar 2.2. Emosi yang diwakilkan dengan kata sifat dikelompokkan ke dalam bidang emosi yang sama.



Gambar 2.2 Permodelan Emosi Menurut Model Russell

2.7 Akurasi, Precision & Recall

Ketika membangun sebuah model klasifikasi, pertanyaan yang muncul adalah bagaimana mengetahui seberapa baik model tersebut. Mengevaluasi model klasifikasi dilakukan dengan mencari tahu seberapa baik hasil prediksi dari model tersebut. *Recall* di Persamaan (2.10**)** *precision* di Persamaan (2.11) dan

akurasi di Persamaan (2.12) adalah metode pengukuran yang biasa digunakan dalam mengevaluasi model, penjelasan variabel ada pada *confusion matrix* pada Tabel 2.1,

Tabel 2.1 *Confusion matrix*

		Kelas Prediksi	
		Benar	Salah
Kelas sebenarnya	Benar	*True Positive* (TP)	*False Negative* (FN)
	Salah	*False Positive* (FP)	*True Negative* (TN)

$$Recall = TP / (TP + FN) \quad (2.10)$$

$$Precision = TP / (TP + FP) \quad (2.11)$$

$$Akurasi = (TP + TN) / (TP + FP + TN + FN) \quad (2.12)$$

Keterangan :

1. *True Positive* (TP) Kelas aktual bernilai benar, dan prediksi berhasil menebak benar
2. *True Negative* (TN) Kelas aktual bernilai salah, dan prediksi berhasil menebak salah.
3. *False Positive* (FP) Kelas aktual bernilai benar, namun prediksi bernilai salah.
4. *False Negative* (FN) Kelas aktual bernilai salah, namun prediksi benilai benar.

2.8 R^2

Suatu metode untuk mengukur kinerja model, biasanya digunakan untuk menggambarkan kesesuaian model statistik yang didefinisikan dengan Persamaan **(2.13)**,

$$R^2 = 1 - \frac{\sum_i (y_i - f_i)^2}{\sum_i (y_i - y)^2} \quad \textbf{(2.13)}$$

Dimana y_i adalah nilai referensi, y adalah *mean* dan f_i adalah estimasi. R^2 mengambil nilai dalam rentang [0;1] dimana nilai 1 berarti data sesuai dengan sempurna. [5]

2.9 Z-Score Normalization

Disebut juga *standart score*, adalah salah satu metode normalisasi terhadap data pengamatan, dirumuskan dengan Persamaan **(2.14)**,

$$z = \frac{x - \mu}{\sigma} \tag{2.14}$$

Dimana z didapatkan dari data pengamatan yang dibagi oleh standart deviasi dan dikurangi nilai *mean* dari populasi data. Menghitung z menggunakan rumus ini membutuhkan *mean* dari populasi dan standar deviasi dari populasi, bukan *mean* sampel atau sampel deviasi. Tetapi mengetahui mean sebenarnya dan standar deviasi suatu populasi seringkali tidak realistis kecuali dalam kasus-kasus seperti pengujian standar, di mana seluruh populasi diukur [9].

2.10 MinMax Normalization

Adalah metode paling mengubah skala dalam rentang dalam [0, 1] atau [−1, 1]. Memilih rentang target tergantung pada sifat data. Rumus umum untuk min-max [0, 1] diberikan seperti pada Persamaan **(2.15)**,

$$x' = \frac{x - min(x)}{max(x) - min(x)} \tag{2.15}$$

Dimana *x* adalah nilai original dan *x'* adalah nilai hasil nomalisasi. Untuk normalisasi dalam suate *range a,b* dapat ditulis seperti Persamaan **(2.16)**,

$$x' = a + \frac{\big(x - min(x)\big)(b - a)}{max(x) - min(x)} \quad \textbf{(2.16)}$$

Dimana *a*,*b* adalah nilai minimal dan maksimal [9].

2.11 Gaussian Process

Berbagai macam metode telah diusulkan untuk menangani masalah *supervised learning*, setidaknya ada dua pendekatan umum. Yang pertama adalah membatasi kelas fungsi yang kita pertimbangkan, misalnya dengan hanya mempertimbangkan fungsi linear. Pendekatan kedua adalah untuk memberikan prior probability untuk setiap fungsi yang memungkinkan, di mana probabilitas yang lebih tinggi diberikan kepada fungsi yang dianggap lebih memungkinkan. Pendekatan pertama memiliki masalah yang jelas karena kita harus memutuskan kelas fungsi yang dipertimbangkan. Pendekatan kedua tampaknya memiliki masalah serius, karena pasti ada serangkaian fungsi yang tak terbatas yang tak terhingga jumlahnya. *Gaussian process* adalah generalisasi dari distribusi probabilitas *Gaussian*. Sedangkan distribusi probabilitas menggambarkan variabel acak yang merupakan skalar atau vektor. Fungsi sebagai vektor yang sangat panjang, setiap entri dalam vektor menentukan nilai fungsi f (x) pada input x tertentu [10].

Gaussian process dapat didefinisikan sebagai kumpulan variabel acak, angka berhingga dalam distribusi Gaussian. *Gaussian Process* ditentukan sepenuhnya dari fungsi mean dan fungsi covariances [6]. Untuk proses sesungguhnya $f(x)$, fungsi *mean* $m(x)$ dapat dituliskan menjadi Persamaan **(2.17)** dan fungsi *covariance* $k(x, x')$ persamaan nya dapat ditulis menjadi Persamaan **(2.18)**,

$$m(x) = \in [\, f(x)] \quad \textbf{(2.17)}$$

$$k(x, x') = \in [(f(x) - m(x))(f(x') - m(x')) \quad \textbf{(2.18)}$$

Sehingga *Gaussian Process* dapat ditulis menjadi Persamaan **(2.19)**,

$$f(x) \sim GP(m(x), k(x, x')) \quad \textbf{(2.19)}$$

Gaussian Process prior pada fungsi $f(x)$ mengartikan untuk sejumlah input X=$\{x_i\} \in R^d$,i=1,..,*n*, *vector* dari fungsi $f = [f(x_i), \dots, f(x_n)]^T = [f_{1,\dots,f_n}]^T$ memiliki distribusi *Gaussian* multivarian dalam Persamaan **(2.20)**,

$$f \sim N(m, K) \quad \textbf{(2.20)}$$

Dimana *mean* m untuk Persamaan **(2.20)** diasumsikan 0, dan *covariance matrix K* memiliki bentuk seperti dalam Persamaan **(2.21)**,

$$K = \begin{bmatrix} k(x_1, x_1) & \dots & k(x_1, x_n) \\ k(x_2, x_1) & \dots & k(x_2, x_n) \\ \vdots & & \vdots \\ k(x_n, x_1) & \cdots & k(x_n, x_n) \end{bmatrix} \quad \textbf{(2.21)}$$

Dan mencirikan korelasi antara titik-titik yang berbeda dalam proses. Untuk $k(x, x')$, setiap fungsi kernel yang menghasilkan *covariance matrix* simetrik dan *semi-definite* dapat digunakan [5].

2.11.1 Gaussian Porcess Regression

Dalam regresi sederhana, jika diberikan input data berupa vector **X** = $\{x_i\}$, dimana i = 1, … ,n dan target berupa *y* = $\{y_i\}$, *x* dan *y* saling berhubungan seperti pada Persamaan **(2.22)**,

$$y = f(x) + \varepsilon \quad \textbf{(2.22)}$$

Dimana $f(x)$ tidak diketahui dan ε sering diasumsikan sebagai *zero mean Gaussian noise*, yaitu $\varepsilon \sim N(0, \sigma_n^2)$. Menempatkan *Gaussian Process prior* kedalam $f(x)$

memungkinkan memarginalkan mereka.yang artinya kita tidak perlu menentukan parameter dan bentuknya [10].

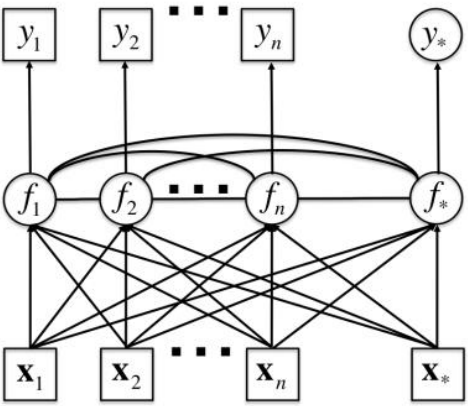
Dalam praktiknya target y_i di asumsikan *independent* bersyarat dengan f_i sehingga *likehood* dapat difaktorkan menjadi persamaan **(2.23)**,

$$p(y|\boldsymbol{f}) = \prod_{1}^{n} p(y_i|f_i) \tag{2.23}$$

Dimana $p(y_i|f_i) = N(y_i|f_i, \sigma_n^2)$, berdasarkan Persamaan **(2.20)**, karena $\boldsymbol{f}$ memiliki distribusi normal, yaitu $\boldsymbol{f}|\boldsymbol{X} \sim N(\boldsymbol{0}, \boldsymbol{K})$, maka $\boldsymbol{y}$ juga *gaussian random vector*. Sehingga menjadi Persamaan **(2.24)**,

$$p(y|\boldsymbol{X}) = N(y|O, \boldsymbol{K} + \sigma_{\boldsymbol{n}}^{\boldsymbol{2}}\boldsymbol{I}) \tag{2.24}$$

Jika diberikan input baru (test) x_* sekarang kita dapat mengestimasi target y_* dan yang terpenting, terdistribusi [10].



Gambar 2.3 Hubungan antar Variabel pada Gaussian Process Regression [5]

Secara visual, hubungan antara semua variabel yang terlibat dapat direpresentasikan pada Gambar 2.3.

Untuk mencari y_*, pertama kita mendapatkan probabilitas gabungan dari target *training* $\boldsymbol{y}$ dan $f_* = f(x_*)$, yang merupakan *Gaussian.* Seperti pada Persamaan **(2.25)**,

$$p(y, f_*|x_*, \boldsymbol{X}) = N\left(0, \begin{bmatrix} \boldsymbol{K} + \sigma_n^2 \boldsymbol{I} & \boldsymbol{k}_* \\ \boldsymbol{k}_*^{\boldsymbol{T}} & k(\boldsymbol{x}_*, \boldsymbol{x}_*) \end{bmatrix}\right) \quad \textbf{(2.25)}$$

Dimana $\boldsymbol{k}_*^{\boldsymbol{T}} = [\boldsymbol{k}(\boldsymbol{x_1}, \boldsymbol{x}_*), \ldots, \boldsymbol{k}(\boldsymbol{x_n}, \boldsymbol{x}_*)]$**.** Lalu, dari distribusi ini, mudah mendapatkan kondisi $p(f_*[y, x_*, \boldsymbol{X}])$ yang juga *Gaussian.* Seperti pada Persamaan **(2.26)**,

$$p(f_*[y, x_*, \boldsymbol{X}]) = N(f_*|\mu_{f*}, \sigma_{f_*}^2) \quad \textbf{(2.26)}$$

Dengan *mean* pada Persamaan **(2.27)** dan *variance* pada Persaman **(2.28)**

$$\mu f_* = \boldsymbol{k}_*^T (\boldsymbol{K} + \sigma_{\boldsymbol{n}}^{\boldsymbol{2}} \boldsymbol{I})^{-1} y \quad \textbf{(2.27)}$$

$$\sigma_{f_*}^2 = k(\boldsymbol{x}_*, \boldsymbol{x}_*) - \boldsymbol{k}_*^{\boldsymbol{T}} (\boldsymbol{K} + \sigma_{\boldsymbol{n}}^{\boldsymbol{2}} \boldsymbol{I})^{-1} \boldsymbol{k}_* \quad \textbf{(2.28)}$$

Perlu dicatat bahwa *mean* μ_{f_*} adalah kombinasi linear dari target yang diamati (*y*). Disatu sisi, *variance* $\sigma_{\boldsymbol{n}}^{\boldsymbol{2}}$ tergantung pada input **X** [5].

2.11.2 Rational Quadratic Covariance Function

Covariance function adalah bagian penting dalam *Gaussian process* [10] *Covariance function* akan menentukan *covariance* antara pasangan variabel acak. *Covariance* antara *output* sebagai fungsi dari *input* Dapat ditulis notasi dalam Persamaan **(2.29)**. Dimana x adalah *input* dan x' adalah *output.*

$$k(x, x') \quad \textbf{(2.29)}$$

Salah satu *covariance function* yang umum digunakan adalah *Squared Exponential Covariance Function*, yang dapat dituliskan dalam Persamaan **(2.30)**,

$$k(x, x') = \sigma_k^2 \exp(-\frac{1}{2l^2}(x - x')^T(x - x')) \quad \textbf{(2.30)}$$

Dimana σ_k^2 seperti pada Persamaan **(2.28)** [5] Skala gabungan (penjumlahan tak terbatas) dari fungsi *covariance squared exponential* (SE) dengan karakteristik panjang yang berbeda, Dengan $\alpha, l > 0$, dapat diartikan sebagai *covariance function Rational Quadratic* [5]. Dapat ditulis pada Persamaan **(2.31)**:

$$k(x, x') = \sigma_k^2 \left(1 + \frac{1}{2\alpha l^2}(x - x')^T(x - x')^{-\alpha}\right) \quad \textbf{(2.31)}$$

2.12 Ffmpeg

Ffmpeg adalah proyek *open-source* gratis yang terdiri dari serangkaian *library* untuk menangani video, audio, dan file multimedia lain. Beberapa *library* lain memanfaatkan nya karena sifatnya yang berupa command-line tool sebagai tool untuk menjalankan proses yang berkaitan dengan file multimedia [11].

2.13 Librosa

Librosa adalah *open source* audio dan musik analisis untuk bahasa pemograman Python. Librosa menyediakan fitur seperti *feature extraction* pada audio baik *low-level* maupun *mid-level feature* [12].

2.14 Scikit-learn

Scikit-learn adalah *open source machine learning library* untuk bahasa pemograman Python. Scikit-learn menyediakan fitur seperti *classification*, *regression*, *clustering*, termasuk juga didalamnya algoritma *support vector machines*, *random forest*, *gradient boosting*, dan lain-lain [13].

2.15 Numpy

Numpy adalah *library Python* yang mendukung pengolahan data pada *array* dan *matrix* multidimensi yang besar. Numpy menyediakan kumpulan fungsi matematika, seperti aljabar linear, transformasi Fourier, pembuatan angka acak, dan lain-lain. *Numpy* bersifat *open source* sehingga banyak dimanfaatkan dalam pengolahan data peneilitian [14].

2.16 Matplotlib

Matplotlib adalah *library Python* yang mendukung pembuatan grafik dua dimensi dalam berbagai format dan dari berbagai jenis data. Matplotlib bersifat *open source* dan banyak digunakan untuk pengolahan data dalam penelitian. Matplotlib dapat membuat plot, histogram, spektrum daya, diagram batang, diagram kesalahan, plot pencar, dan lain-lain [15].

BAB III
PERANCANGAN SISTEM

Bab ini menjelaskan mengenai perancangan data dan sistem pemetaan emosi ke dalam model Russell menggunakan *Gaussian process regression*. Bab ini juga akan menjelaskan gambaran umum sistem dalam bentuk diagram alir.

3.1 Perancangan Data

Data yang digunakan sebagai masukan awal dari sistem pemetaan emosi ke dalam model Russell menggunakan *Gaussian process regression* adalah dataset MediaEval'13. MediaEval adalah badan yang didedikasikan untuk mengevaluasi algoritma baru untuk akses dan pengambilan multimedia. Dalam hal ini kata 'multi' dalam multimedia dan berfokus pada aspek manusia dan sosial dari tugas multimedia. pada tahun 2013 MediaEval mengadakan workshop yang bertempat di Barcelona, Catalunya, Spain, pada Jumat- sabtu, 18-19 Oktober 2013 [16]. Beberapa tugas yang diberikan dalam workshop tersebut adalah emosi dalam musik, yang menggunakan dataset 1000 song dari *Free Music Archive* [4]. Dataset MediaEval'13 adalah dataset yang berisikan 1000 lagu bebas hak cipta yang kemudian ditemukan kemiripan data sehingga dikurangi menjadi 744 lagu. Dataset terdiri dari lagu dengan durasi penuh, lagu dengan durasi 45 detik, serta anotasi sumbu x dan y mereka.

Spesifikasi lengkap dataset dapat dilihat pada Tabel 3.1.

Tabel 3.1 Spesifikasi Awal Dataset

Keterangan	Spesifikasi
Jumlah lagu (awal)	1000
Jumlah lagu (unik)	744
Ekstensi	Mp3
Sampling rate frequency	44100Hz
Durasi	45 detik
Tipe	Monophonic

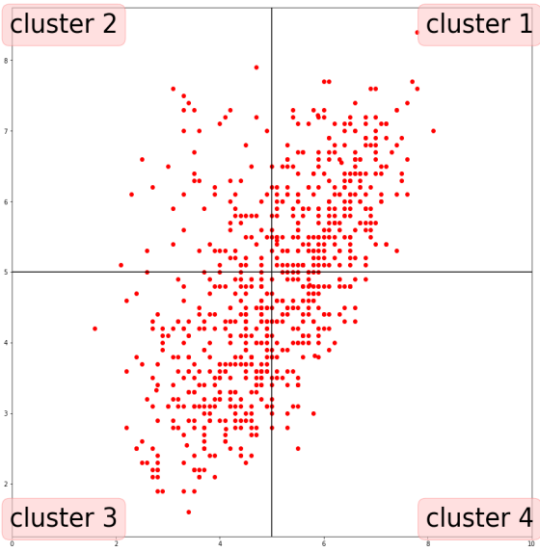
Informasi pada lagu dapat dilihat pada Tabel 3.2

Tabel 3.2 Informasi lagu

No	Song id	Mean arousal	Std arousal	Mean valence	Std valence
1	2	3.1	0.99	3.0	0.66
2	3	3.5	1.84	3.3	1.70
3	4	5.7	1.49	5.5	1.71
4	5	4.4	2.11	5.3	1,.94
5	7	5.8	1.54	6.4	1.77

Tabel 3.2 menggambarkan data yang akan digunakan sebagai dataset (lagu unik pada table 3.1). Pada Tabel 3.2 ditunjukkan lima data dari dataset sebagai representasi, dataset ini dianotasi secara manual, oleh *coworker* Amazon MTurk workers, kolom *song_id* adalah *id* dari setiap lagu. Kolom *mean_arousal* adalah nilai titik untuk sumbu Y pada model Russell [4]. Kolom *mean_valence* adalah nilai titik untuk sumbu X pada model Russell [4]. Kolom *std* menghitung standart deviasi dari hasil anotasi oleh *coworker*. Anotasi *valence-arousal* memiliki kisaran antara 0 hingga 10, yang artinya semakin tinggi nilai *valence-arousal* maka akan semakin positif emosi yang dihasilkan.

Gambar 3.1 menunjukkan persebaran lagu terhadap model Russell.

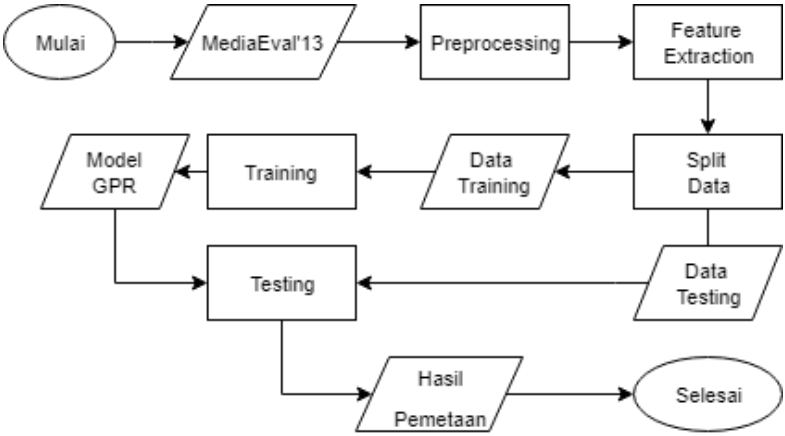


Gambar 3.1 Persebaran Lagu Pada Model Russell

Titik merah mewakili lagu, sedangkan sumbu *x* adalah *valence* dan sumbu *y* adalah *arousal*. Bidang terbagi empat mewakili *cluster* pada model Russell. Nilai *valence* dan *arousal*. Tabel 3.3 menunjukkan jumlah lagu pada setiap *cluster*.

Tabel 3.3 Jumlah Lagu Setiap *Cluster*

Cluster	**Jumlah Lagu**
Cluster 1	259
Cluster 2	73
Cluster 3	292
Cluster 4	120



Gambar 3.2 Diagram Alir Sistem yang Dibangun

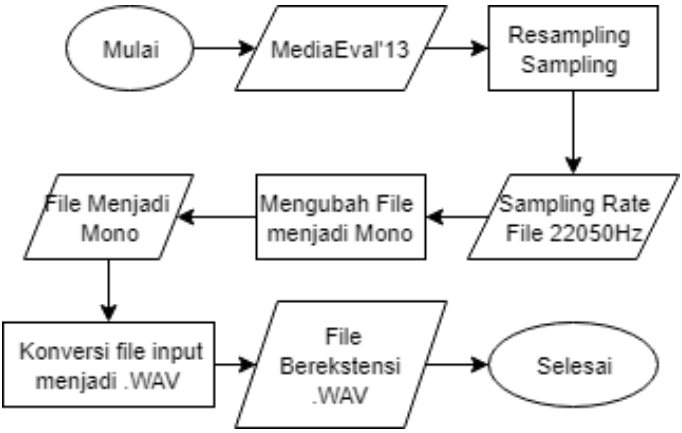
3.2 Desain Umum Sistem

Sistem pemetaan emosi ke dalam model Russell yang dibangun memiliki proses utama diantaranya *preprocesing* data, *feature exctraction*, *training*, *testing* dan pemetaan. Diagram alir dari sistem ditunjukan pada Gambar 3.2.

Akan dilakukan *preprocessing* data terlebih dahulu sebelum data digunakan sebagai data di ekstraksi fiturnya. *Feature extraction* pada tugas ini memanfaatkan *library python librosa.* Setiap lagu akan diekstraksi fitur-fitur seperti *mel frequency cepstral coefficients*, *timbre features*, *spectral crest factor*, *spectral flatness measure* dan *chromagram.*

Lalu data dibagi menjadi data *training* dan data *testing*, data *training* akan masuk ke proses *training* menggukanan metode *Gaussian process regression* menggunakan kernel *Rational Quadratic.* Pada tahap ini parameter *kernel* akan diatur secara otomatis.

Tahap *testing* pada tugas ini dilakukan dengan memasukkan data *testing* kedalam model hasil pembentukan tahap *training.* Dan performa nya akan di ukur dengan metode R^2.



3.2.1 Tahap Preprocessing Data

Pada tahap ini akan dilakukan proses penyeragaman data yaitu pada format file menjadi berekstensi *WAV*. Langkah pertama dengan penyeragaman *Sampling rate* menjadi 22050Hz. Selain itu data yang dipakai akan diubah menjadi *mono.* Langkah ini bertujuan agar bentuk data menjadi seragam, alasan perubahan menjadi format *WAV* seperti yang dijelaskan pada bagian 2.3 Diagram alir dari tahap *preprocessing* ditunjukkan pada Gambar 3.3.

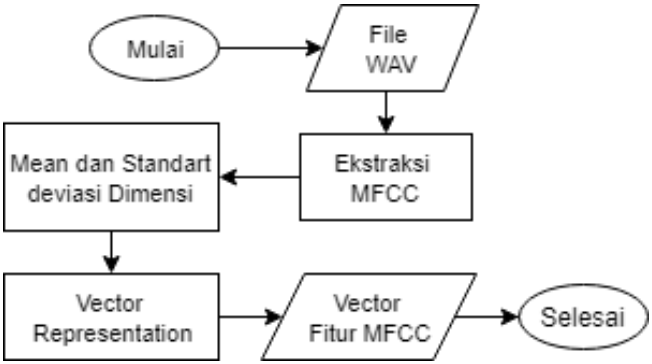
Setiap file dalam dataset akan diseragamkan *sampling rate* nya, hal ini berpengaruh terhadap jumlah *frame* yang akan di

Se

Gambar 3.3 Diagram Alir Tahap *Preprocessing*

3.2.2 Tahap Feature Extraction

Pada tahap ini setiap lagu akan diekstraksi fitur – fitur nya secara terpisah. Seperti yang dijelaskan pada bagian 2.5, ekstraksi fitur menggunakan *library python Librosa.* Hasil ekstraksi awal adalah *matrix* berdimensi *(n,m)* dimana *n* adalah dimensi fitur dan *m* adalah *window* dari lagu. *Matrix* tersebut nantinya akan dicari *mean* dan standart deviasi nya berdasarkan sumbu *window* yang akan membentuk vector representatif berisikan *mean* dan standart deviasi untuk membentuk *vector feature* yang merepresentasikan fitur dari lagu.



Gambar 3.4 Diagram Alir Tahap Ekstraksi Fitur *MFCC*

3.2.2.1 Tahap Ekstraksi Fitur *MFCC*

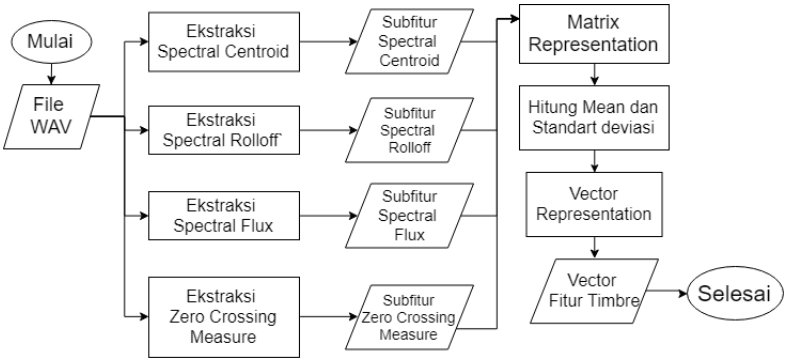
Pada tahap ini akan diekstraksi sebuah fitur yaitu *MFCC*, hasil tahap ini berupa *vector* berisikan *mean* dan standart deviasi. Diagram alir tahap ini dapat dilihat pada Gambar 3.4.

3.2.2.2 Tahap Ekstraksi Fitur *Timbre*

Pada tahap ini akan diekstaksi empat buah subfitur yaitu *spectral centroid, spectral rolloff, spectral flux,* dan *zero-crossing rate*. Setiap subfitur akan menghasilkan vector berisikan

mean dan standart deviasi yang akan ditumpuk menjadi satu *matrix* yang berisikan *mean* dan standart deviasi setiap subfitur yang akan yang menjadi representasi fitur timbre. Hasil *matrix* akan diubah menjadi *vector representation* sebagai representasi fitur *timbre* dari sebuah lagu.

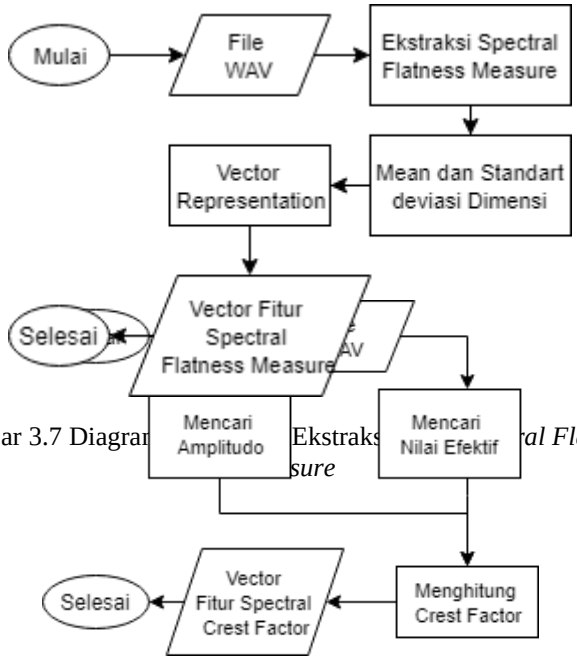
Diagram alir tahap ini dapat dilihat pada Gambar 3.5.



Gambar 3.5 Diagram Alir Tahap Ekstraksi Fitur *Timbre*

3.2.2.3 Tahap Ekstraksi Fitur *Spectral Crest Factor*

Pada tahap ini akan diekstraksi sebuah fitur yaitu *spectral crest factor*, hasil dari tahap ini berupa *vector* berisi *mean* dan standart deviasi. Diagram alir untuk tahap ini adalah pada Gambar 3.6.



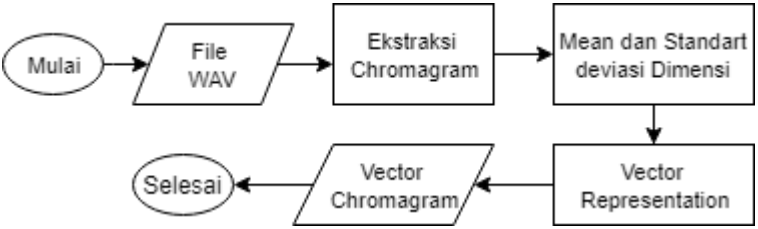
Gambar 3.7 Diagram Alir Ekstraksi *ral Flatness sure*

Gambar 3.6 Diagram Alir Tahap Ekstraksi Fitur *Spectral Crest Feature*

3.2.2.4 Tahap Ekstraksi Fitur *Spectral Flatness Measure*

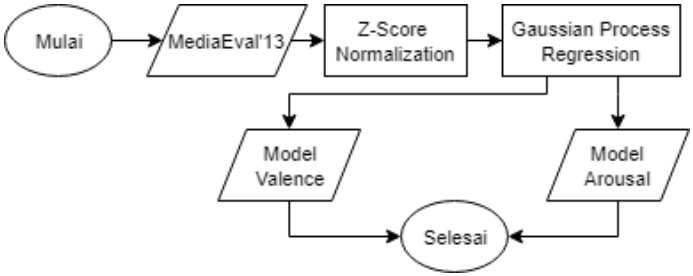
Pada tahap ini akan diekstraksi sebuah fitur yaitu *spectral flatness measure*, hasil dari tahap ini berupa *vector* berisi *mean* dan standart deviasi. Diagram alir untuk tahap ini adalah pada Gambar 3.7.

3.2.2.5 Tahap Ekstraksi Fitur *Chromagram*



Gambar 3.8 Diagram Alir Tahap Ekstraksi Fitur *Chromagram*

Pada tahap ini akan diekstraksi sebuah fitur yaitu *chromagram*, hasil dari tahap ini berupa *vector* berisi *mean* dan standart deviasi. Diagram alir untuk tahap ini adalah pada Gambar 3.8.



3.2.3 Tahap Training

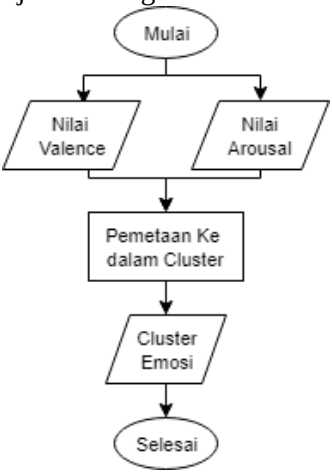
Pada tahap ini, fitur hasil ekstraksi akan dinormalisasi dengan metode *Z-Score Normalization*, selanjutnya data yang telah dinormalisasi akan dibagi dalam data *training* dan *test*. Data *training* akan dibentuk menjadi dua model dengan *Gaussian Process Regression*, hasil *training* berupa model untuk *valence* (sumbu *x*) dan *arousal* (sumbu *y*). Diagram alir untuk proses ini sepe

Gambar 3.9 Diagram Alir Tahap *Training*

Tahap *Gaussian Process Regressor* melibatkan *optimizer* yang akan digunakan untuk mengatur *hyperparameter* milik fungsi *kernel* dengan meminimalkan nilai *marginal likehood* . Pengaturan *hyperparameter* akan dilakukan otomatis didalam proses, sehingga *hyperparameter* dari *kernel* milik model hasil pelatihan akan diatur sesuai data *training* untuk model.

3.2.4 Tahap Pemetaan

Pada tahap ini nilai prediksi untuk *valence* (sumbu *x*) dan *arousal* (sumbu *y*) akan menjadi titik koordinat pada model Russel yang akan menentukan *cluster* emosi dari sebuah lagu. Gambar 3.10 menunjukkan diagram alir untuk tahap pemetaan



Gambar 3.10 Diagram Alir Tahap Pemetaan

3.2.5 Tahap Testing

Pada tahap ini akan dievaluasi performa dari model *valence* (sumbu *x*) *arousal* (sumbu *y*). Performa model akan diukur menggunakan metode R^2. Lalu hasil pemetaan akan diukur nilai akurasi, *precision*, dan *recall* nya.

BAB IV
IMPLEMENTASI

Bab ini menjelaskan mengenai implementasi perangkat lunak dari rancangan sistem yang telah dibahas pada Bab 3 meliputi kode program dalam perangkat lunak. Selain itu, implementasi dari tiap proses, parameter masukan, keluaran, dan beberapa keterangan yang berhubungan dengan program juga dijelaskan.

4.1 Lingkungan Implementasi

Dalam mengimplementasikan aplikasi pengenalan ekspresi manusia diperlukan beberapa perangkat pendukung sebagai berikut.

4.1.1 Perangkat Keras

Implementasi tugas akhir ini menggunakan desktop *personal computer* (PC) LENOVO-10132. Sistem operasi yang digunakan adalah Windows 10 64-bit. PC yang digunakan memiliki spesifikasi Intel Core i3-4150 dengan kecepatan 3,5 GHz, *Random Access Memory* (RAM) sebesar 8 GB.

4.1.2 Perangkat Lunak

Personal computer dari sisi perangkat lunak memiliki spesifikasi antara lain menggunakan bahasa pemograman Python 3.6, dilengkapi dengan ffmpeg dan *library* antara lain Pandas, Librosa, Math, Numpy, Matplotlib dan Scikit-learn.

4.2 Implementasi Preprocessing Data

Pada bagian ini akan dijabarkan implementasi pada tahap *preprocessing* data, yaitu, penyeragaman *sampling rate*, perubahan tipe file menjadi *mono.* serta perubahan format file menjadi .*WAV*

4.2.1 Implementasi *Sampling Rate* dan *Mono*

Proses penyeragaman *sampling rate* dan mengubah file menjadi tipe *mono* memanfaatkan fungsi *load* dari Librosa, dimana parameter *src* adalah *path* menuju file yang akan dibuka, *mono* untuk memilih tipe yang file, berisikan *Boolean* jika *True* makan file akan dibuka sebagai *mono* dan *sr* adalah besaran *sampling rate* dalam satuan Hz. Seperti pada Kode Sumber 4.1.

```
1.  y, sr = librosa.load(src,mono=True,sr=22050)
```

Kode Sumber 4.1 Funsi Penyeragaman *Sampling Rate* dan *Mono*

Fungsi ini akan mengembalikan dua nilai yang akan dipakai pada proses selanjutnya, *y* adalah representasi gelombang file dalam *numpy array*, *sr* adalah besaran *sampling rate.*

4.2.2 Implementasi Convert WAV

Proses konversi file menjadi *.WAV* memanfaatkan fungsi *write_wav* dari Librosa. Dimana parameter yang digunakan adalah *dst* sebagai *path* destinasi dari file hasil konversi,*y* dan *sr* adalah output dari proses sebelumnya, yaitu representasi gelombang dalam *numpy array* untuk *y* dan besaran *sampling rate* untuk *sr*. Seperti pada Kode Sumber 4.2.

```
1.  librosa.output.write_wav(dst, y, sr)
```

Kode Sumber 4.2 Fungsi Konversi File menjadi *WAV*

Hasil dari proses ini akan disimpan dalam *path* di dalam variabel *dst* berupa file berekstensi *.WAV.*

4.3 Implementasi Feature Extraction

Pada bagian ini akan dijabarkan implementasi pada tahap *feature extraction* data, yaitu ekstraksi pada fitur *mfcc,* fitur *timbre,* fitur *spectral crest factor,* fitur *spectral flatness measure,* fitur *chromagram.*

4.3.1 Ekstraksi Fitur MFCC

Bagian ini akan menjelaskan proses ekstraksi fitur *MFCC*, memanfaatkan fungsi *mfcc* dari Librosa, seperti pada Kode Sumber 4.3.

```
1.  mfcc = librosa.feature.mfcc(y=y, sr=sr, n_mfcc=13)
```

Kode Sumber 4.3 Fungi Ekstraksi Fitur *MFCC*

Parameter yang digunakan dalam fitur ini adalah *y* dan *sr* yang merupakan representasi gelompang untuk *y* dan *sampling rate* untuk *sr* dari lagu, serta *n_mfcc* adalah dimensi fitur yang akan digunakan. Hasilnya berupa *matrix* berdimensi n,m dimana n adalah dimensi fitur dan m adalah *window* lagu. Untuk memudahkan pengolahan data, hasil dari ekstraksi akan disimpan dalam *pandas dataframe* seperti pada Kode Sumber 4.4.

```
1.  data = pd.DataFrame(mfcc)
2.  mean = data.mean(axis = 1).values #Get Mean
3.  std = data.std(axis = 1).values #Get std
4.  vector.append(np.insert(mean,13,std))
```

Kode Sumber 4.4 Fungsi *Vector Representation* Fitur *MFCC*

Pada Kode Sumber 4.4 akan dicari *mean* dan standart deviasi dari fitur *MFCC* yang nantinya akan membentuk *vector representation* untuk fitur *MFCC.* Mencari *mean* memakai *method mean* dari *dataframe* dan standart deviasi memakai *method std* dengan parameter *axis*=1 yang berarti patokan untuk pengerjaannya adalah berdasarkan kolom *dataframe.* Lalu hasil perhitungan *mean* yang disimpan pada variabel *mean* akan digabung dengan data perhitungan standart deviasi dalam variabel *std*, kemudian akan disimpan dalam *list* bernama *vector* yang akan menjadi representasi untuk fitur *MFCC.*

4.3.2 Ekstraksi Fitur Timbre

Pada tahap ini akan dilakukan ekstrasi fitur *Timbre* yang berisi empat subfitur yaitu *spectral centroid, spectral rolloff, spectral flux*, *zero crossing rate*. Menggunakan Librosa. Kode Sumber 4.5 digunakan untuk mengekstraksi fitur *spectral centroid*, memanfaatkan fungsi *spectral_centroid.*

```
1.  cent = librosa.feature.spectral_centroid(y=y,sr=sr)
```

Kode Sumber 4.5 Fungi Ekstraksi Fitur *Spectral Centroid*

Input untuk fungsi ini adalah *y* dan *sr* yang merupakan representasi gelompang untuk *y* dan *sampling rate* untuk *sr*. Untuk fitur *spectral flux* akan diekstraksi dengan fungsi *onset_strength* dari Librosa, seperti pada Kode Input 4.6.

```
1.  flux = librosa.onset.onset_strength(y=y, sr=sr)
```

Kode Sumber 4.6 Fungi Ekstraksi Fitur *Spectral Flux*

Input untuk fungsi pada Kode Gambar 4.6 adalah *y* dan *sr* yang merupakan representasi gelompang untuk *y* dan *sampling rate* untuk *sr*. Untuk fitur *spectral rolloff* akan diekstraksi dengan fungsi *spectral_rolloff* dari Librosa. Seperti pada Kode Sumber 4.7

```
1.  rolloff=librosa.feature.spectral_rolloff(y=y,sr=sr)
```

Kode Sumber 4.7 Fungi Ekstraksi Fitur *Spectral Rolloff*

Input untuk fungsi ini adalah *y* dan *sr* yang merupakan representasi gelompang untuk *y* dan *sampling rate* untuk *sr*. Fitur *zero crossing rate* diekstraksi dengan fungsi *zero_crossing_rate* - seperti pada Kode Sumber 4.8

```
1.  zc = librosa.feature.zero_crossing_rate(y)
```

Kode Sumber 4.8 Fungi Ekstraksi Fitur *Zero Crossing*

Input pada fungsi ini hanya memakai variabel *y* yang merupakan representasi gelompang. Setiap fitur nantinya akan menghasilkan *vector* dengan panjang sama dengan jumlah *window* lagu. Langkah selanjutnya adalah membentuk *matrix representation* dari fitur *timbre*, yang dijawab pada Kode Sumber 4.9.

```
1.  a = pd.DataFrame(cent)
2.  b = pd.DataFrame(flux).T
3.  c = pd.DataFrame(rolloff)
4.  d = pd.DataFrame(zc)
5.  frame = [a,b,c,d]
6.  data = pd.concat(frame)
```

Kode Sumber 4.9 Fungsi *Matrix Representation* Fitur *Timbre*

Hasil disimpan dalam *pandas dataframe* untuk memudahkan perhitungan *mean* dan standart deviasi yang akan dihitung pada Kode Sumber 4.10 dan hasilnya akan menjadi representasi untuk fitur *timbre*.

```
1.  mean = data.mean(axis = 1).values #Get Mean
2.  std = data.std(axis = 1).values #Get std
3.  vector.append(np.insert(mean,4,std))
```

Kode Sumber 4.10 Fungsi *Vector Representation* Fitur *Timbre*

4.3.3 Ekstraksi Fitur Spectral Crest Factor

Ekstraksi pada fitur *spectral crest factor* dilakukan dalam beberapa tahapan. Pertama akan dicari *aplitudo* dari gelombang, menggukanan method *max* terhadap gelombang *y*, dapat dilihat pada Kode Sumber 4.11.

```
1.  peak = y.max()
```

Kode Sumber 4.11 Fungsi Pengambil *Amplitudo*

Langkah kedua adalah menghitung nilai efektif dari gelombang seperti pada Kode Sumber 4.12.

```
1.  rms = librosa.feature.rmse(y=y)
2.  n = rms.size
3.  square = rms**2
4.  result = math.sqrt((1/n)*(square.sum()))
```

Kode Sumber 4.12 Fungsi Pencarian Nilai Efektif

Pada Kode Sumber 4.12 dengan memanfaatkan fungsi *rmse* dari Librosa, akan dicari nilai rms dari gelombang. Hasilnya adalah *numpy array* nilai *rms* dari setiap *window*. Lalu akan dicari jumlah *window* dari *rms* gelombang dengan *method size* dari Librosa. Nilai efektif dari gelombang akan disimpan dalam variabel *result* setelah melalui perhitungan.

Kode Sumber 4.13 menunjukkan langkah terakhir yaitu pembagian antara *amplitude* dengan nilai efektif

```
1.  crest = [peak/result]
```

Kode Sumber 4.13 Fungsi Pencarian *Spectral Crest Factor*

Hasil dari proses ini adalah sebuah nilai yang akan merepresentasikan fitur *spectral crest factor.*

4.3.4 Ekstraksi Fitur Spectral Flatness Measure

Ekstraksi untuk fitur *spectral flatness measure* dapat dicari memafaatkan fungsi *spectral_flatness* dari Librosa. Seperti pada Kode Sumber 4.14.

```
1.  flatness = librosa.feature.spectral_flatness(y=y)
```

Kode Sumber 4.14 Fungsi Pencarian *Spectral Flatness Measure*

Selanjutnya hasil akan dihitung *mean* dan standart deviasi memanfatkan *method mean* dan *std* dari *pandas dataframe.* Seperti pada Kode Sumber 4.15.

```
1.  data = pd.DataFrame(flatness)
2.  mean = data.mean(axis = 1).values #Get Mean
3.  std = data.std(axis = 1).values #Get std
4.  vector.append(np.insert(mean,1,std))
```

Kode Sumber 4.15 Fungsi *Vector Representation* Fitur *Timbre*

Variabel *vector* adalah sebuah *list* yang nantinya berisikan *mean* dan standart deviasi, yang nantinya akan digabungkan dengan *method append.* Seperti pada Kode Sumber 4.15. Hasil dari proses ini adalah *vector representation* dari fitur *spectral flatness measure.*

4.3.5 Ekstraksi Fitur Chromagram

Proses ekstraksi fitur *chromagram* dilakukan dengan fungsi *chroma_stft* dari Librosa, dengan input *y* dan *sr* yang merupakan *numpy array* berisikan representasi gelombang serta *sampling rate*, serta *n_chroma* diset 12 menandakan dimensi *chromagram* yang akan dibuat. Seperti pada Kode Sumber 4.16.

```
1.  chroma = librosa.feature.chroma_stft(y=y, sr=sr,
    n_chroma=12)
```

Kode Sumber 4.16 Fungsi Ekstraksi Fitur *Chromagram*

Hasil dari proses ini adalah *matrix representation* dari fitur *chromagram*, berdimensi n,m dimana n adalah dimensi fitur dan m adalah *window*. Kemudian *matrix representation* fitur *chromagram* akan disimpan dalam *pandas dataframe* untuk memudahkan proses selanjutnya, yaitu pencarian *mean* dan standart deviasi. Dengan memanfaatkan *method mean* dan *std* seperti pada Kode Sumber 4.17.

```
1.  data = pd.DataFrame(chroma)
2.  mean = data.mean(axis = 1).values #Get Mean
3.  std = data.std(axis = 1).values #Get std
4.  vector.append(np.insert(mean,12,std))
```

Kode Sumber 4.17 Fungsi *Vector Representation* Fitur *Chromagram*

Hasil dari Fungsi ini akan disimpan dalam variabel *vector* yang berupa *list*, dan akan menjadi *vector representation* untuk fitur *chromagram.*

4.4 Implementasi Training

Pada tahap ini akan dibentuk dua model untuk memprediksi nilai untuk *valence* (sumbu *x*) dan *arousal* (sumbu *y*).

4.4.1 MinMax Normalization

Normalisasi *MinMax* dilakukan memanfaatkan fungsi *minmax_scale* dari *Sklearn*. Dapat dilihat pada Kode Sumber 4.18.

```
1.  minmax_scale(X=arousal_mean.values,feature_range=(-
    1,1))
```

Kode Sumber 4.18 Fungsi Normalisasi *MinMax* untuk Nilai Anotasi

Parameter *X* adalah data target yang akan dinormalisasi, pada kode Sumber 4.18 input yang dimasukkan adalah *arousal_mean*, yang artinya data yang akan dinormalisasi adalah anotasi nilai *arousal* (sumbu *y*), input dapat diganti *valence* (sumbu *x*) untuk nilai *valence.* Input *arousal_mean* memakai *method values* agar menjadi *numpy array* karena tipe awal dari *arousal_mean* adalah *python dataframe*. Parameter *feature_range* memiliki input berupa *tuple* dengan format (min,max).

4.4.2 Z-Score Normalization

Normalisasi *Z-Score* dilakukan dengan fungsi pada Kode Sumber 4.19.

```
1.  scaler = StandardScaler()
2.
3.  def norm(data):
4.      scaler.fit(data)
5.      normalized = scaler.transform(data)
6.
7.      return normalized
```

Kode Sumber 4.19 Fungsi Untuk Normalisasi *Z-Score*

Kode Sumber 4.19 digunakan untuk menormalisasi fitur, variabel *data* akan berisikan fitur dari hasil ekstraksi. Seperti pada Persamaan **(2.14)** maka untuk mencari nilai *Z-Score* perlu mencari nilai *mean* dan standart deviasi, dengan memanfaatkan fungsi *StandardScaler* dari *Sklearn*, dengan input *data* akan dicari *mean* dan standart deviasi nya ketika masuk pada *method* fit. Lalu akan distandarisasi dengan *method* transform.

4.4.3 Split Data

Split data akan dilakukan memanfaatkan fungsi *train_test_split* dari *Sklearn*, seperti pada Kode Sumber 4.20.

```
1.  def split(data,value):
2.      train_x, test_x, train_y, test_y =
    train_test_split(data,value,test_size=0.3,
    random_state=1000)
3.
4.      return train_x, test_x, train_y, test_y
```

Kode Sumber 4.20 Fungsi Untuk *Split* Data

Kode pada Kode Sumber 4.20 memiliki parameter *data* dan *value*. Kedua parameter ini akan menjadi input parameter untuk fungsi *train_test_split* dari *Sklearn*. Fungsi *train_test_split*

memiliki empat parameter, yaitu *data*, *value*, *test_size*, dan *random_state*. Parameter *data* berarti data fitur yang akan diproses, *value* adalah data target yang akan diproses, *test_size* di isi 0.3 yang artinya dari data input, 30% dari data akan menjadi data *testing*. *Random_state* diset 1000 agar *split* setiap uji coba memiliki hasil yang sama.

Hasil *split* dari fungsi ini disimpan dalam *train_x* yang berisikan data fitur untuk tahap *training*, *test_x* yang berisikan data fitur untuk tahap *testing*, *train_y* yang berisikan data target untuk tahap *training*, *test_y* yang berisikan data target untuk tahap.*testing*.

4.4.4 Gaussian Process Regression

Pada tahap ini akan dibuat model pemetaan lagu terhadap model Russell, akan dibangun dua buah model regresi yang digunakan untuk mencari nilai *valence* (sumbu x) dan nilai *arousal* sumbu y.

Kode Sumber 4.21 menunjukkan inisiasi model untuk metode ini.

```
1.  gpr = GPR(kernel=kernels, alpha=1e-10,
    normalize_y=True, optimizer='fmin_l_bfgs_b',
    n_restarts_optimizer=10, random_state=1000)
```

Kode Sumber 4.21 Inisiasi Model

GPR adalah model dari *Sklearn* yang di inisiasi untuk fungsi *GaussianProcessRegressor* dari *Sklearn* yang memiliki beberapa parameter untuk input, seperti *kernel*, *alpha*, *optmazer*, *n_restarts_optimizer*, *normalize_y*, *copy_X_train*, *random_state*.

Parameter *kernel* adalah jenis fungsi kernel *Gaussian process regression* yang akan diisi dengan fungsi pada Kode Sumber 4.22. Parameter *alpha* adalah nilai tambah yang dimasukkan fungsi *kernel*, mengacu pada Persamaan **(2.31)**. Input parameter *alpha* diisi 1e-10. Parameter *n_restart_optimizer* adalah parameter yang digunakan sebagai iniasi jumlah berapa

kali kernel di *restart* untuk mencari parameter *kernel* yang memaksimalkan *log marginal likehood*. Parameter *optimizer* adalah *optimizer* yang digunakan untuk memaksimalkan parameter *kernel*. *Fmin_l_bfgs_b* adalah fungsi dari *spicy* yang merupakan *value default*. Parameter *normalize_y* adalah parameter untuk menandakan apakah data target dinormalisasi atau tidak, Parameter *copy_X_train* adalah parameter yang menandakan apakah data fitur akan disimpan dalam *object* atau tidak. *Random_state* adalah parameter yang digunakan sebagai inisiasi, diberi input seragam untuk kedua model agar memiliki titik awal sama.

Kode Sumber 4.22 adalah fungsi dari kernel yang digunakan

```
1.  kernels = 1.0 * RationalQuadratic(length_scale=1.0,
    alpha=1.0, length_scale_bounds=(1e-6, 1e6),
    alpha_bounds=(1e-6, 1e6))
```

Kode Sumber 4.22 Kernel RationalQuadratic

Kernel yang digunakan adalah *RationalQuadratic*, dengan memanfaatkan fungsi *RationalQuadratic* dari *Sklearn*. Fungsi ini memiliki beberapa input seperti *length_scale, alpha, length_scale_bounds, alpha_bounds.*

length_scale dan *alpha* adalah parameter yang berisikan inisiasi nilai, sementara *length_scale_bounds* dan *alpha_bounds* adalah batas dengan tipe input *tuple* dengan format (batas bawah,batas atas).

Dari inisiasi model dan *kernel* akan dilakukan *training* terhadap dua model, yaitu model *valence* (sumbu *x*) dan *arousal* (sumbu *y*).

```
1.  def training(data,values,clf):
2.      trained = clf.fit(data,values)
3.
4.  return trained
```

Kode Sumber 4.23 Fungsi *Training*

Seperti pada Kode Sumber 4.23, setiap model, baik untuk nilai *valence* maupun *arousal* akan menggukanan fungsi ini, dengan *method fit* data akan di *train* sehingga membentuk model yang akan ditampung pada variabel *trained*, dan setelah data di *training* dan terbentuk model, akan diprediksi titik-titik untuk *valence* (sumbu *x*) dan *arousal* (sumbu *y*) menggunakan Kode Sumber 4.24.

```
1.  def predict(test,true,clf):
2.      pred = clf.predict(test)
3.      df = pd.DataFrame(data=[pred,true])
4.
5.  return pred,df
```

Kode Sumber 4.24 Fungsi *Predict*

Pada fungsi ini, prediksi akan ditampung pada variabel *pred.* Menggunakan *method pred* dari *Sklearn object* dengan input berupa data testing, akan mengembalikan nilai hasil prediksi. Variabel *df* adalah *python dataframe* yang digunakan untuk visualisasi hasil.

Parameter dalam kernel berubah selama *training* data, sehingga nilai parameter *length_scale* dan *alpha* dapat berbeda dengan inisiasi pada Kode Sumber 4.22. Untuk itu Kode Sumber 4.23 digunakan untuk mengatur parameter kernel sesuai hasil *training.*

```
1.  trained.set_params(kernel=trained.kernel_)
```

Kode Sumber 4.25 Fungsi Mengatur Parameter

Pada Kode Sumber 4.25 *trained* adalah *classifier* yang telah melalui proses *training* sehingga dengan *method set_params* parameter kernel diatur dengan *method kernel_..Method kernel_* akan mengembalikan parameter kernel setelah proses *training.*

4.5 Implementasi Pemetaan

Tahap pemetaan adalah tahap dimana hasil prediksi dari model untuk *valence* (sumbu *x*) dan *arousal* (sumbu *y*) dijadikan koordinat dalam *cluster* emosi.

Kode Sumber 4.26 menunjukkan proses pemetaan sebuah lagu kedalam *cluster* emosi.

```
1.  def emotion(a,v):
2.      if ((v > 0) & (a >= 0)):
3.          cluster = 'cluster 1'
4.      if ((v <= 0) & (a > 0)):
5.          cluster = 'cluster 2'
6.      if ((v < 0) & (a <= 0)):
7.          cluster = 'cluster 3'
8.      if ((v >= 0) & (a < 0)):
9.          cluster = 'cluster 4'
10.
11.     return cluster
```

Kode Sumber 4.26 Fungsi Pemetaan ke Dalam *Cluster* Emosi

Kode Sumber 4.26 adalah sebuah fungsi yang memiliki parameter *v* dan *a*, *v* berisikan hasil prediksi untuk *valence* (sumbu *x*) sedangkan *a* adalah hasil prediksi untuk *arousal* (sumbu *y*). Fungsi ini akan mengembalikan posisi *cluster* seperti yang pada Gambar 2.2.

4.6 Implementasi Testing

Pada tahap ini data *testing* akan diukur performa dari model *valence* (sumbu *x*) dan arousal (sumbu *y*) serta performa pemetaan.

4.6.1 Pengukuran Performa Model Valence dan Arousal

Tahap ini akan diukur performa model *valence* dan *arousal*, seperti yang dapat dilihat pada Kode Sumber 4.27

```
1.  def score(pred,true):
2.      r2 = r2_score(true, pred)
3.
4.  return r2
```

Kode Sumber 4.27 Fungsi Pengukuran Performa Model

Dengan memanfaatkan fungsi *r2_score* dari *Sklearn*, fungsi ini memiliki parameter input berupa nilai *true* dan *pred.* Parameter *true* akan berisikan data target yang digunakan untuk *testing* sementara *pred* akan berisikan dara target yang digunakan.

4.6.2 Validasi Model Regresi

Tahap ini akan dilakukan validasi model regresi dengan metode *K-Fold Cross Validation.* Dengan memanfaatkan fungsi *KFold* dari *Sklearn*, seperti pada Kode Sumber 4.28, *kf* adalah variabel sebagai iniasi *KFold. N_split* adalah jumlah yang akan dijalankan, dan *random_state* adalah angka yang diset agar setiap uji coba memiliki pembagian yang sama.

```
1.  kf = KFold(n_splits=10, random_state=1000)
```

Kode Sumber 4.28 Inisiasi KFold

Kode Sumber 4.29 menunjukkan fungsi *cross validation* yang akan dilakukan. Dengan input *clf, data, values* fungsi ini akan memanfaatkan fungsi *cross_val_score* dari *Sklearn. Input clf* akan diisikan model dari Kode Sumber 4.21, *data* akan berisikan dataset, *values* akan berisikan fitur.

```
1.  def crosval(clf,data,values):
2.       scores = cross_val_score(clf, data, values,
    scoring='r2', cv=kf)
3.
4.       return scores
```

Kode Sumber 4.29 Fungsi *Cross Validation*

4.6.3 Pengukuran Performa Pemetatn

Tahap ini akan diukur performa Pemetaan, seperti pada Kode Sumber 4.30.

```
5.  def evaluate(true,pred):
6.      acc = accuracy_score(true, pred)
7.      rec = recall_score(true, pred, average='micro')
8.      prec = precision_score(true, pred,
    average='micro')
9.      mat = confusion_matrix(true, pred)
10.     print("\nAccuracy\t:",acc.round(2) * 100)
11.     print("\nRecall\t:",rec.round(2) * 100)
```

Kode Sumber 4.30 Fungsi Evaluasi Pemetaan

Pada Kode Sumber 4.30 Fungsi *evaluate* memiliki dua parameter input, *true* dan *pred. True* akan berisikan nilai *cluster* sesungguhnya, dan *pred* akan berisikan nilai prediksi. Dengan menggunakan fungsi *accuracy_score* untuk mencari nilai akurasi, *recall_score* untuk mencari nilai *recall*,dan *precission_score* untuk mencari nilai *precision.* Ketiga fungsi itu adalah fungsi yang disediakan oleh *Sklearn*.

(Halaman ini sengaja dikosongkan)

BAB V
UJI COBA DAN EVALUASI

Bab ini akan membahas mengenai hasil uji coba sistem yang telah dirancang dan dibuat. Uji coba dilakukan untuk mengetahui kinerja sistem dengan lingkungan uji coba yang telah ditentukan.

5.1 Lingkungan Uji Coba

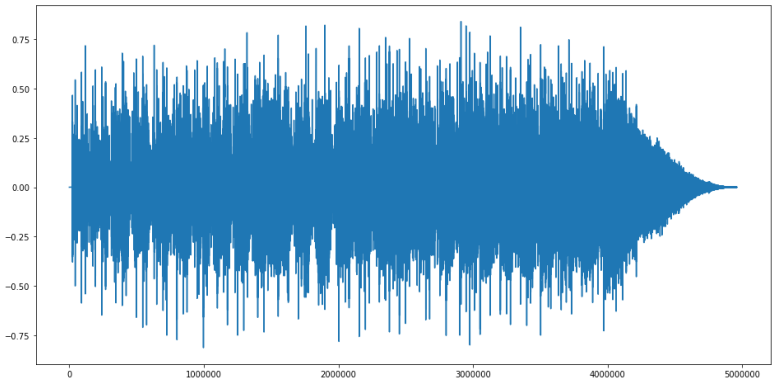
Lingkungan uji coba pada tugas akhir ini desktop *personal computer* (PC) LENOVO-10132. Sistem operasi yang digunakan adalah Windows 10 64-bit. PC yang digunakan memiliki spesifikasi Intel Core i3-4150 dengan kecepatan 3,5 GHz, *Random Access Memory* (RAM) sebesar 8 GB. PC dari sisi perangkat lunak memiliki spesifikasi antara lain menggunakan bahasa pemograman Python 3.6, dilengkapi dengan ffmpeg dan *library* antara lain Pandas, Librosa, Math, Numpy, Matplotlib dan Scikit-learn.

5.2 Dataset

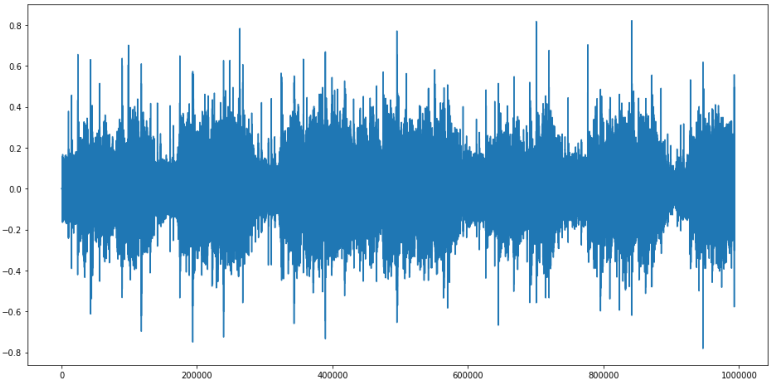
Pada tugas akhir ini, data yang digunakan adalah MediaEval'13. Dataset MediaEval'13 berisikan 1000 lagu bebas hak cipta, yang memiliki anotasi nilai untuk *valence* (sumbu *x*) dan *arousal* (sumbu *y*). Digunakan 744 lagu karena ditemukan redudansi pada beberapa lagu. Setiap anotasi lagu memiliki *range* antara 0 sampai 9 yang selanjutnya dilakukan *MinMax Normalization* dengan range minimal -1 dan maksimal 1. Dari 744 lagu dalam dataset akan dijadikan data *training* sebesar 520 lagu dan *test* 224 menjadi data *testing*.

5.3 Hasil Preprocessing

Sebelum ekstraksi fitur, akan dilakukan preprocessing seperti penyeragaman *sampling rate*, merubah tipe jadi *mono*, dan perubahan format menjadi *WAV*. Gambar 5.1 menunjukkan *plotting Waveform* sebelum preprocessing.

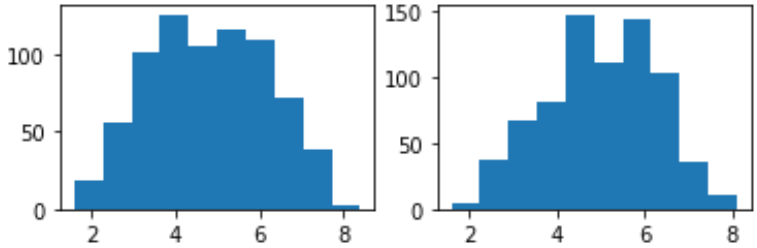


Gambar 5.1 *Waveform* File Sebelum *Prerocessing*

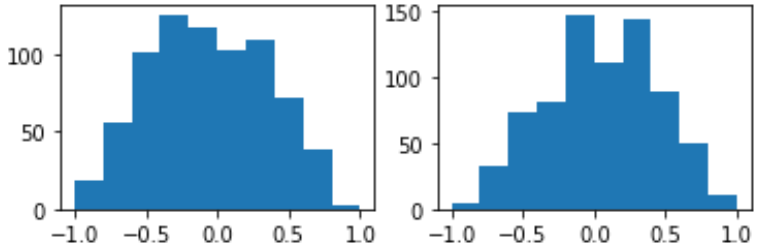


Gambar 5.2 *Waveform* File Setelah *Prerocessing*

Pada Gambar 5.1 file masih memiliki *sampling rate* sebesar 44100Hz, berformat *.mp3* dan bertipe *mon*, sedangkan Gambar 5.2 file sudah memiliki *sampling rate* sebesar 22050Hz, berformat *.WAV*, dan bertipe mono.



Gambar 5.4 anotasi *valence* dan *arousal* sebelum normalisasi



Gambar 5.3 anotasi *valence* dan *arousal* setelah normalisasi

5.4 Hasil MinMax Normalization

Proses ini dilakukan agar nilai dari anotasi *valence* (sumbu *x*) dan *arousal* (sumbu *y*) memiliki titik tengah pada angka 0. Gambar 5.3 menunjukkan anotasi sebelum normalisasi. Pada Gambar 5.3 terdapat dua buah gambar, gambar posisi kiri menunjukkan anotasi untuk *arousal* (sumbu y) dan posis kanan adalah anotasi untuk *valence* (sumbu *x*). *Range* anotasi diantara 0 sampai 9 dan titik tengah berada pada angka 5.

Pada Gambar 5.4 terdapat dua buah gambar,posisi kiri menunjukkan anotasi untuk *arousal* (sumbu *y*) dan gambar posisi kanan *valence* (sumbu *x*). Setelah melalui proses normalisasi, *range* akan diantara -1 hingga 1 dan titik tengah berada pada angka 0.

5.5 Hasil Z-Score Normalization

Proses ini dilakukan kepada data fitur agar setiap fitur dari lagu memiliki standart deviasi 1 dan *mean* 0. Normalisasi dilakukan pada representasi fitur, bukan hanya representasi *window* saja. Tabel 5.1 menunjukkan hasil normalisasi pada fitur *MFCC.* Tabel 5.2 menunjukkan hasil normalisasi pada fitur *timbre*. Tabel 5.3 menunjukkan hasil normalisasi pada fitur *Spectral Crest Factor* dan *Spectral Flatness Measure.* Tabel 5.4 menunjukkan hasil normalisasi pada fitur *Chromagram*

Tabel 5.1 Perbandingan Fitur *MFCC* Sebelum dan Sesudah Normalisasi

	Sebelum	**Sesudah**
Mean	7.10	0
Standart Deviasi	12.81	1

Tabel 5.2 Perbandingan Fitur *Timbre* Sebelum dan Sesudah Normalisasi

	Sebelum	**Sesudah**
Mean	876.42	0
Standart Deviasi	389.76	1

Tabel 5.3 Perbandingan Fitur *SCF* & *SFM* Sebelum dan Sesudah Normalisasi

	Sebelum	**Sesudah**
Mean	1.98	0
Standart Deviasi	0.76	1

Tabel 5.4 Perbandingan Fitur *Chromagram* Sebelum dan Sesudah Normalisasi

	Sebelum	**Sesudah**
Mean	0.30	0
Standart Deviasi	0.01	1

5.6 Hasil Feature Extraction

Hasil ekstraksi setiap fitur akan menjadi matrix yang setiap barisnya merepresentasikan lagu, dan kolom nya merepresentasikan *window.*

Tabel 5.5 Hasil Ekstraksi dan Representasi Fitur

Nama Fitur	Dimensi
MFCC	26
Timbre	8
Spectral Crest Factor	1
Specral Flatness Measure	2
Chromagram	12

5.7 Skenario Uji Coba

Proses uji coba berguna untuk menemukan fitur yang menghasilkan performa model yang paling optimal. fitur yang tepat akan memberikan hasil yang lebih baik pada saat proses uji coba. Ada 4 macam skenario uji coba dan semuanya akan dicoba pada *Gaussian process regression*. Skenario uji coba yang akan dilakukan yaitu:

1. Uji coba 1 (*Baseline* fitur.) (i)
2. Uji coba 2 ((i) + *Timbre*.) (ii)
3. Uji coba 3 ((ii) + *SCF & SFM*.) (iii)
4. Uji coba 4 ((iii) + *Chromagram*.) (iv)

Setiap uji coba akan menggunakan model dan fungsi kernel seperti pada Kode Sumber 4.21 dan 4.22. fitur dengan performa terbaik pada uji coba pertama akan menjadi *baseline* fitur untuk uji coba selanjutnya. Akan dihitung nilai R^2 untuk model *valence* (sumbu *x*) dan *arousal* (sumbu y). dan akan dihitung nilai akurasi untuk mengetahui performa pemetaan. Akan dilakukan validasi terhadap setiap uji coba menggunakan 10-*fold cross validation.*

Akan disajikan grafik dan *confusion matrix* untuk menunjukkan performa setiap fitur, garis biru menunjukkan nilai prediksi *valence-arousal* sedangkan garis merah adalah nilai

aktual dari *valence-arousal,* Sumbu *x* pada grafik berarti *range* anotasi, sementara sumbu *y* berarti lagu dari data *testing.* Untuk *confusion matrix,* semakin gelap warna maka semakin banyak kemunculan data tersebut, sedangkan semakin terang menandakan semakin sedikit kemunculannya. Label 0 untuk *cluster 1,* Label 1 untuk *cluster 2,* Label 2 untuk *cluster 3,* Label 3 untuk *cluster 4.*

5.7.1 Uji Coba 1

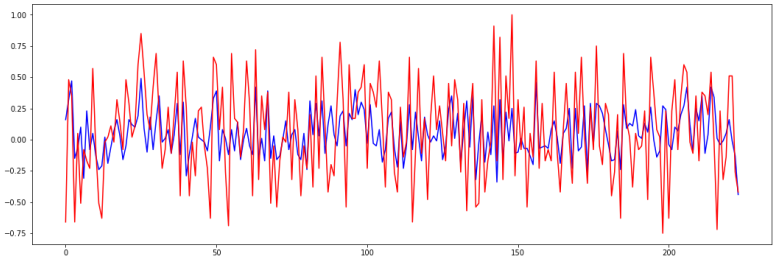
Pada uji coba ini, setiap fitur akan digunakan secara individu untuk menjadi data fitur, yaitu fitur *MFCC, Timbre, Spectral Crest Factor* & *Spectral Flatness Measure* dan *Chromagram.* Uji coba dilakukan untuk mengetahui *baseline* fitur.

Gambar 5.5 menunjukkan prediksi untuk model *valence* (sumbu *x*) dan Gambar 5.6 menunjukkan prediksi untuk model *arousal* (sumbu *y*) untuk fitur *MFCC.* Gambar 5.7 menunjukkan *confusion matrix* dari hasil pemetaan dengan fitur *MFCC.*

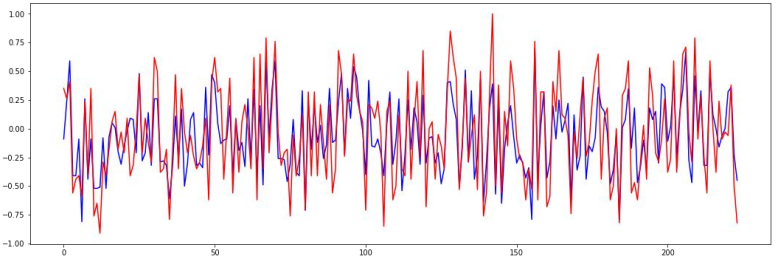
Gambar 5.8 menunjukkan prediksi untuk model *valence* (sumbu *x*) dan Gambar 5.9 menunjukkan prediksi untuk model *arousal* (sumbu *y*) untuk fitur *Timbre*. Gambar 5.10 menunjukkan *confusion matrix* dari uji coba dengan fitur *Timbre.*

Gambar 5.11 menunjukkan prediksi untuk model *valence* (sumbu *x*) dan Gambar 5.12 menunjukkan prediksi untuk model *arousal* (sumbu *y*) untuk fitur *SCF* & *SFM.* Gambar 5.13 menunjukkan *confusion matrix* dari uji coba dengan fitur *SCF* & *SFM.*

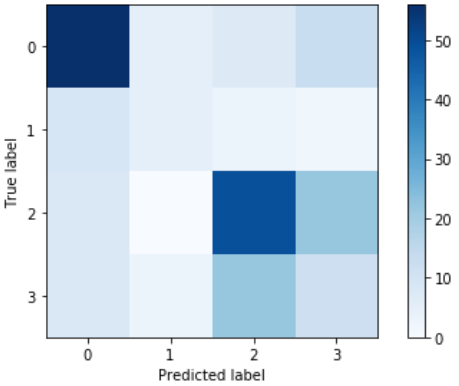
Gambar 5.14 menunjukkan prediksi untuk model *valence* (sumbu *x*) dan Gambar 5.15 menunjukkan prediksi untuk model *arousal* (sumbu *y*) untuk fitur *Chromagram.* Gambar 5.16 menunjukkan *confusion matrix* dari uji coba dengan fitur *Chromagram.*



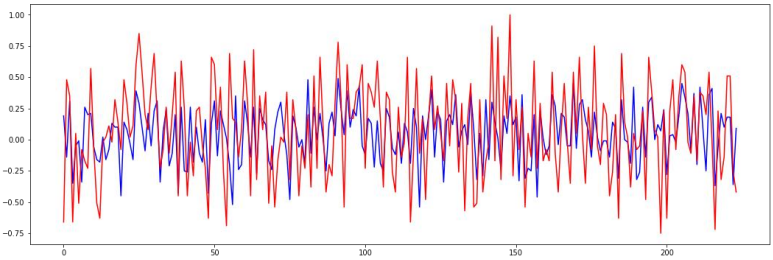
Gambar 5.8 Prediksi *Valence* (sumbu *x*) dengan Fitur *MFCC*



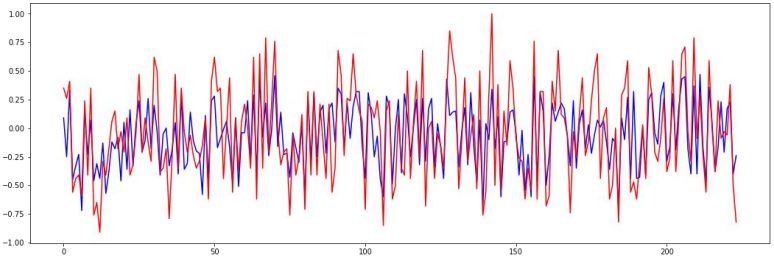
Gambar 5.7 Prediksi *Arousal* (sumbu *y*) dengan Fitur *MFCC*



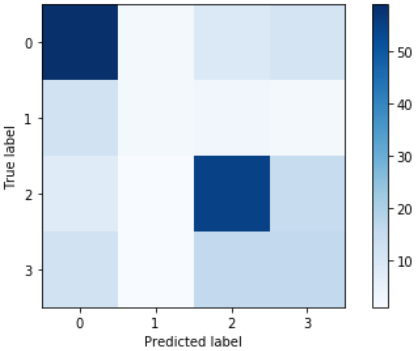
Gambar 5.5 Gambar 5.6 *Confusion Matrix* Uji Coba dengan Fitur *MFCC*



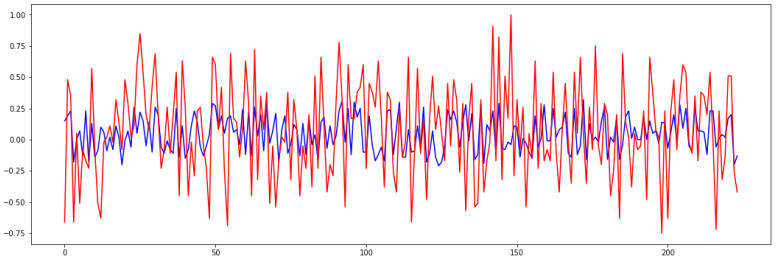
Gambar 5.9 Prediksi *Valence* (sumbu *x*) dengan Fitur *Timbre*



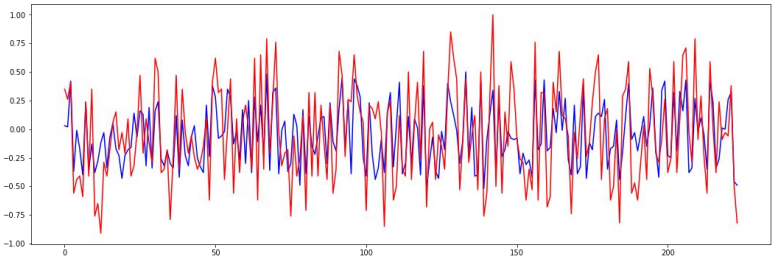
Gambar 5.11 Prediksi *Arousal* (sumbu *y*) dengan Fitur *Timbre*



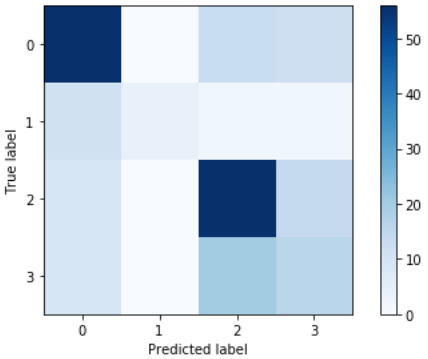
Gambar 5.10 *Confusion Matrix* Uji Coba dengan Fitur *Timbre*



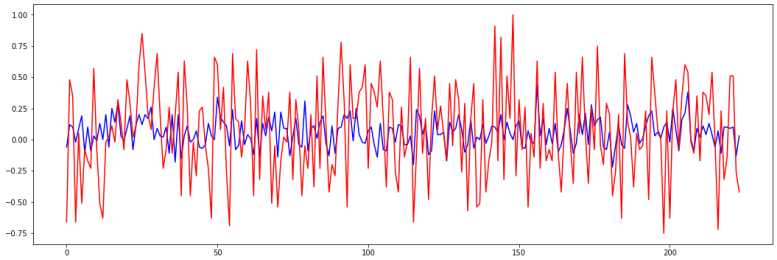
Gambar 5.12 Prediksi *Valence* (sumbu *x*) dengan Fitur *SCF* & *SFM*



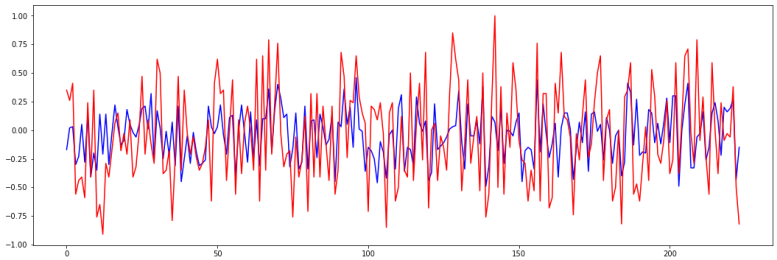
Gambar 5.13 Prediksi *Arousal* (sumbu *y*) dengan Fitur *SCF* & *SFM*



Gambar 5.14 *Confusion Matrix* Uji Coba dengan Fitur *SCF* & *SFM*

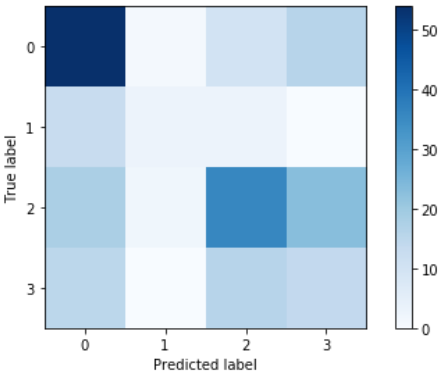


Gambar 5.15 Prediksi *Valence* (sumbu *x*) dengan Fitur *Chromagram*



Gambar 5.16 Prediksi *Arousal* (sumbu *y*) dengan Fitur *Chromagram*

.



Gambar 5.17 *Confusion Matrix* Uji Coba dengan Fitur *Chromagram*

Tabel 5.6 menunjukkan *classification report* untuk uji coba 1 dengan fitur *MFCC*. Sementara Tabel 5.7 menunjukkan

classification report untuk uji coba 1 dengan fitur *Timbre*. Sementara Tabel 5.8 menunjukkan *classification report* untuk uji coba 1 dengan fitur *SCF* & *SFM*. Sementara Tabel 5.9 menunjukkan *classification report* untuk uji coba 1 denga fitur *Chromagram.*

Tabel 5.6 *Classification Report* dengan Fitur *MFCC*

Akurasi	***Cluster***	**Precision**	***Recall***
0.54	***Cluster* 1**	0.69	0.69
	***Cluster* 2**	0.38	0.26
	***Cluster* 3**	0.60	0.62
	***Cluster* 4**	0.24	0.27

Tabel 5.7 *Classification Report* dengan Fitur *Timbre*

Akurasi	***Cluster***	**Precision**	***Recall***
0.59	***Cluster* 1**	0.65	0.73
	***Cluster* 2**	0.33	0.11
	***Cluster* 3**	0.66	0.70
	***Cluster* 4**	0.36	0.36

Tabel 5.8 *Classification Report* dengan Fitur *SCF & SFM*

Akurasi	***Cluster***	**Precision**	***Recall***
0.59	***Cluster* 1**	0.66	0.69
	***Cluster* 2**	1.00	0.21
	***Cluster* 3**	0.62	0.71
	***Cluster* 4**	0.36	0.36

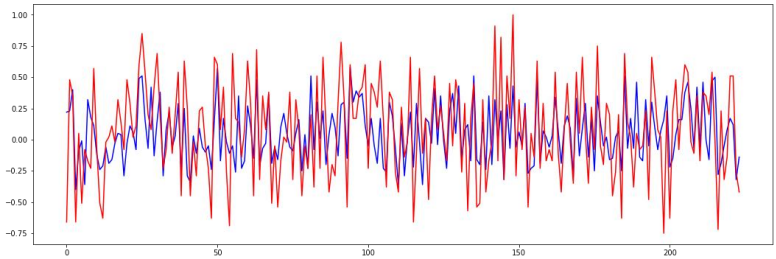
Tabel 5.9 *Classification Report* dengan Fitur *Chromagram*

Akurasi	***Cluster***	**Precision**	***Recall***
0.48	***Cluster* 1**	0.54	0.67
	***Cluster* 2**	0.50	0.16
	***Cluster* 3**	0.55	0.46
	***Cluster* 4**	0.26	0.31

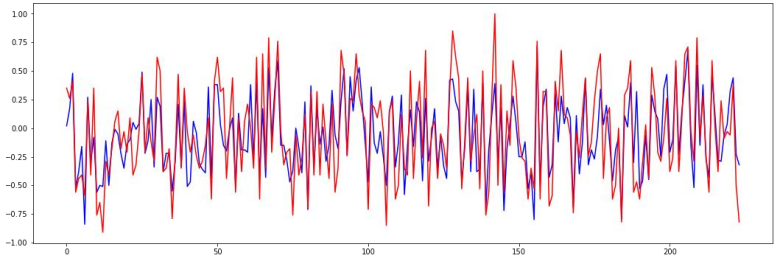
5.7.2 Uji Coba 2

Pada Uji coba ini akan di uji pengaruh *Timbre* terhadap performa model.

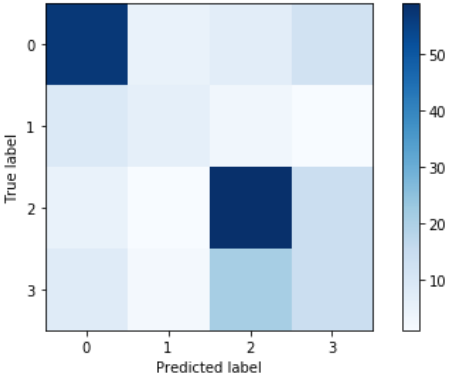
Gambar 5.17 menunjukkan prediksi untuk model *valence* (sumbu *x*) dan Gambar 5.18 menunjukkan prediksi untuk model *arousal* (sumbu *y*) untuk fitur *MFCC + Timbre.* Gambar 5.20 menunjukkan *confusion matrix* dari uji coba dengan fitur *MFCC + Timbre.*



Gambar 5.18 Prediksi *Valence* (sumbu *x*) dengan Fitur *MFCC + Timbre*



Gambar 5.19 Prediksi *Arousal* (sumbu *y*) dengan Fitur *MFCC + Timbre*



Gambar 5.20 *Confusion Matrix* Uji Coba dengan Fitur *MFCC + Timbre*

Tabel 5.10 menunjukkan *classification report* untuk uji coba 2 denga fitur *MFCC + Timbre.*

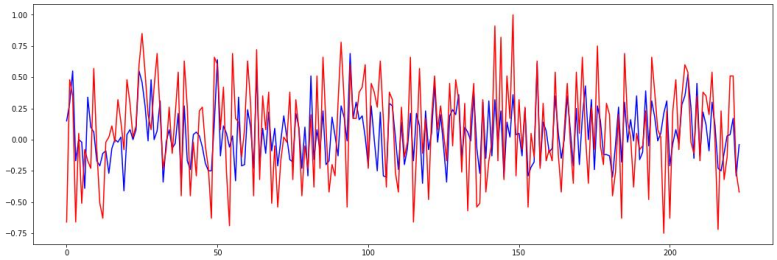
Tabel 5.10 *Classification Report* Fitur dengan Fitur *MFCC + Timbre*

Akurasi	***Cluster***	***Precision***	***Recall***
0.61	***Cluster* 1**	0.72	0.70
	***Cluster* 2**	0.43	0.32
	***Cluster* 3**	0.66	0.75
	***Cluster* 4**	0.34	0.31

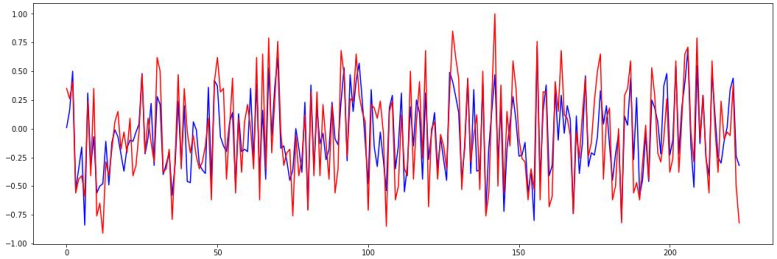
5.7.3 Uji Coba 3

Pada Uji coba ini akan di uji pengaruh *SFC &SFM* terhadap performa model.

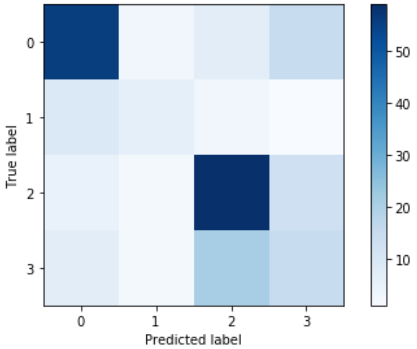
Gambar 5.21 menunjukkan prediksi untuk model *valence* (sumbu *x*) dan Gambar 5.22 menunjukkan prediksi untuk model *arousal* (sumbu *y*) untuk fitur *MFCC + Timbre + SCF* & *SFM*. Gambar 5.23 menunjukkan *confusion matrix* dari uji coba dengan fitur *MFCC + Timbre + SCF* & *SFM*.



Gambar 5.21 Prediksi *Valence* (sumbu *x*) dengan Fitur *MFCC + Timbre + SCF & SFM*



Gambar 5.23 Prediksi *Arousal* (sumbu *y*) dengan Fitur *MFCC + Timbre + SCF & SFM*



Gambar 5.22 *Confusion Matrix* Uji Coba dengan Fitur *MFCC + Timbre + SCF & SFM*

Tabel 5.13 menunjukkan *classification report* untuk uji coba 3 dengan fitur *MFCC + Timbre + SCF & SFM.*

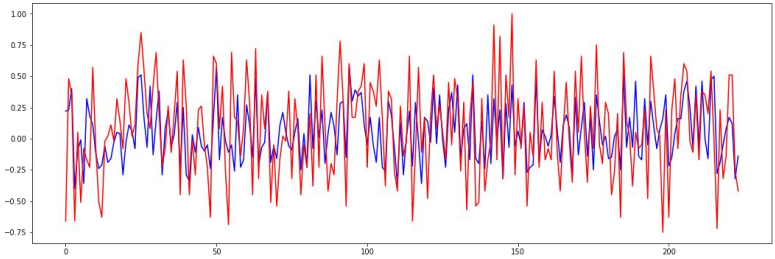
Tabel 5.11 *Classification Report* Fitur dengan *MFCC + Timbre + SCF & SFM*

Akurasi	***Cluster***	***Precision***	***Recall***
0.61	***Cluster* 1**	0.73	0.69
	***Cluster* 2**	0.46	0.32
	***Cluster* 3**	0.66	0.75
	***Cluster* 4**	0.34	0.33

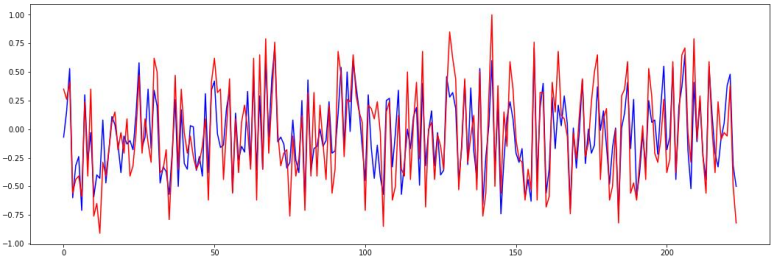
5.7.4 Uji Coba 4

Pada Uji coba ini akan di uji pengaruh *Chromagram* terhadap performa model.

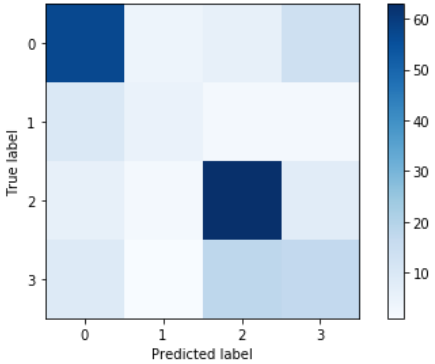
Gambar 5.24 menunjukkan prediksi untuk model *valence* (sumbu *x*) dan Gambar 5.25 menunjukkan prediksi untuk model *arousal* (sumbu *y*) untuk fitur *MFCC + Timbre + SCF & SFM + Chromagram.* Gambar 5.26 menunjukkan *confusion matrix* dari uji coba dengan fitur *MFCC + Timbre +SCF & SFM + Chromagram.*



Gambar 5.24 Prediksi *Valence* (sumbu *x*) dengan Fitur *MFCC + Timbre + SCF & SFM + Chromagram*



Gambar 5.25 Prediksi *Arousal* (sumbu *y*) dengan Fitur *MFCC + Timbre + SCF & SFM + Chromagram*



Gambar 5.26 *Confusion Matrix* Uji Coba dengan Fitur *MFCC + Timbre +SCF & SFM + Chromagram*

, Sementara Tabel 5.18 menunjukkan *classification report* untuk uji coba 4 denga fitur *MFCC + Timbre + SCF & SFM + Chromagram*

Tabel 5.12 *Classification Report* Fitur *MFCC + Timbre +SCF & SFM + Chromagram*

Akurasi	***Cluster***	***Precision***	***Recall***
0.63	***Cluster* 1**	0.70	0.70
	***Cluster* 2**	0.42	0.26
	***Cluster* 3**	0.71	0.80
	***Cluster* 4**	0.41	0.38

5.7.5 KFold-Cross Validation

Menggunakan 10 *fold* setiap uji coba akan dimasukkan ke dalam fungsi *cross validation*. *Mean* adalah hasil rata-rata dari setiap *fold.* Tabel 5.13 akan menunjukkan hasil dari *10 fold cross validation* untuk uji coba dengan fitur *MFCC.*

Tabel 5.14 akan menunjukkan hasil dari *10 fold cross validation* untuk uji coba dengan fitur *timbre.* **Error! Reference ource not found.** akan menunjukkan hasil dari *10 fold cross validation* untuk uji coba dengan fitur *SCF & SFM.* Tabel 5.16 akan menunjukkan hasil dari *10 fold cross validation* untuk uji coba dengan fitur *Chromagram.*

Tabel 5.17 akan menunjukkan hasil dari 10 *fold cross validation* untuk uji coba dengan fitur *MFCC + Timbre.* Tabel 5.18 akan menunjukkan hasil dari 10 *fold cross validation* untuk uji coba dengan fitur *MFCC + Timbre.+ SCF & SFM.*

Tabel 5.19 akan menunjukkan hasil dari 10 *fold cross validation* untuk uji coba dengan fitur *MFCC + Timbre.+ SCF & SFM + Chromagram.*

Tabel 5.13 Hasil *Cross Validation Fitur MFCC*

Fold	Arousal	Valence
1	0.39	0.09
2	0.47	0.04
3	0.52	0.02
4	0.51	-0.16
5	0.66	0.35
6	0.57	0.08
7	0.51	0.32
8	0.60	0.26
9	0.64	0.26
10	0.52	0.11
Mean	0.54	0.14

Tabel 5.14 Hasil *Cross Validation Fitur Timbre*

Fold	Arousal	Valence
1	0.34	0.32
2	0.24	0.31
3	0.28	0.28
4	0.41	0.04
5	0.54	0.45
6	0.48	0.30
7	0.51	0.39
8	0.56	0.36
9	0.33	0.12
10	0.45	0.05
Mean	0.41	0.26

Tabel 5.15 Hasil *Cross Validation Fitur SCF & SFM*

Fold	Arousal	Valence
1	0.36	-0.01
2	0.38	0.04
3	0.23	-0.01
4	0.16	-0.40
5	0.50	0.27
6	0.32	0.14
7	0.33	0.08
8	0.36	0.11
9	0.48	0.28
10	0.49	-0.003
Mean	0.36	0.05

Tabel 5.16 Hasil *Cross Validation Fitur Chromagram*

Fold	Arousal	Valence
1	0.1	-0.23
2	0.25	0.012
3	0.46	-0.12
4	0.24	-0.20
5	0.14	0.15
6	0.33	0.14
7	0.15	0.10
8	0.13	-0.12
9	0.27	0.1
10	0.37	0.13
Mean	0.24	0

Tabel 5.17 Hasil *Cross Validation Fitur MFCC + Timbre*

Fold	Arousal	Valence
1	0.48	0.34
2	0.54	0.30
3	0.59	0.26
4	0.64	0.13
5	0.65	0.49
6	0.62	0.27
7	0.64	0.46
8	0.66	0.35
9	0.60	0.32
10	0.59	0.17
Mean	0.60	0.31

Tabel 5.18 Hasil *Cross Validation Fitur MFCC + Timbre + SCF & SFM*

Fold	Arousal	Valence
1	0.5	0.35
2	0.53	0.3
3	0.55	0.25
4	0.63	0.13
5	0.66	0.51
6	0.63	0.27
7	0.65	0.46
8	0.64	0.34
9	0.59	0.31
10	0.61	0.17
Mean	0.6	0.31

Tabel 5.19 Hasil *Cross Validation Fitur MFCC + Timbre + SCF & SFM + Chromagram*

Fold	Arousal	Valence
1	0.45	0.32
2	0.58	0.35
3	0.61	0.23
4	0.53	0.05
5	0.66	0.4
6	0.67	0.31
7	0.64	0.43
8	0.66	0.29
9	0.63	0.28
10	0.66	0.28
Mean	0.61	0.29

5.8 Hasil dan Evaluasi

Pada uji coba 1, dimana fitur digunakan secara terpisah dan satu persatu, dapat diketahui bahwa fitur *MFCC* menghasilkan nilai R^2 yang lebih baik untuk kedua model, yaitu 0.64 untuk *arousal* dan 0.29 untuk *valence*. Seperti yang dijelaskan pada Bagian 2.5, hal ini terjadi karena ketiga fitur yang lain belum cukup merepresentasikan sebuah lagu, dimana fitur *timbre* berfokus pada karakteristik sinyal, *SCF* & *SFM* untuk memeriksa sinyal *noise* pada sinyal input, dan *Chromagram* digunakan untuk memeriksa kelas *pitch* dominan. Seperti pada Tabel 5.20 yang berisikan hasil uji coba 1.

Pada uji coba 2, dimana fitur *MFCC* digabungkan dengan *Timbre* menghasilkan nilai R^2 untuk kedua model, yaitu 0.67 untuk *arousal* dan 0.29 untuk *valence*. Menambahkan fitur *Timbre* meningkatkan performa model *arousal* namun tidak pada model *valence.* Seperti pada Tabel 5.21.

Pada uji coba 3, dimana fitur *MFCC + Timbre* ditambah dengan fitur *SCF* & *SFM*. Hasil uji coba seperti pada Tabel 5.22, penambahan *SCF* & *SFM* tidak berpengaruh besar terhadap model *arousal*, sementara untuk model *valence,* penambahan fitur ini meningkatkan performa menjadi 0.38.

Pada uji coba 4, digunakan keempat fitur digunakan, sehingga hasil uji coba seperti yang ditunjukkan pada Tabel 5.23,

Dari uji coba 1 hingga 4 dapat diketahui bahwa fitur yang paling representatif adalah fitur *MFCC*, dan kombinasi dengan performa paling baik adalah kombinasi *MFCC + Timbre + SCF & SFM + Chromagram.*

Hasil pemetaan menunjukkan akurasi tertinggi sebesar 63%, performa pemetaan tergolong rendah terhadap *cluster* 2 dan *cluster* 4 karena jumlah data dalam kelas tersebut yang tergolong kecil dibanding kelas lain.

Tabel 5.20 Hasil Nilai R^2 Uji Coba 1

No	Fitur	Dimensi	Arousal	Valence
1	*MFCC*	26	0.64	0.29
2	*Timbre*	8	0.55	0.30
3	*SCF & SFM*	3	0.48	0.18
4	*Chromagram*	24	0.30	0.12

Tabel 5.21 Hasil Nilai R^2 Uji Coba 2

No	Fitur	Dimensi	Arousal	Valence
1	*MFCC + Timbre*	34	0.67	0.29

Tabel 5.22 Hasil Nilai R^2 Uji Coba 3

No	Fitur	Dimensi	Arousal	Valence
1	*MFCC + Timbre + SCF & SFM*	37	0.68	0.38

Tabel 5.23 Hasil Nilai R^2 Uji Coba 4

No	Fitur	Dimensi	Arousal	Valence
1	*MFCC + Timbre + SCF & SFM + Chromagram*	61	0.68	0.38

BAB VI
KESIMPULAN DAN SARAN

Bab ini membahas tentang kesimpulan yang didasari oleh hasil uji coba yang telah dilakukan pada bab sebelumnya. Kesimpulan nantinya sebagai jawaban dari rumusan masalah yang dikemukakan. Selain kesimpulan, juga terdapat saran yang ditujukan untuk pengembangan penelitian lebih lanjut di masa depan.

6.1 Kesimpulan

Dalam pengerjaan Tugas Akhir ini setelah melalui tahap perancangan aplikasi, implementasi metode, serta uji coba, diperoleh kesimpulan sebagai berikut:

1. Fitur *Mel Frequency Cepstral Coefficient* adalah *baseline* fitur yang memiliki peran besar untuk performa model, Sementara *Chromagram* secara individu adalah fitur yang kurang berperan dalam performa model, kombinasi terbaik adalah *MFCC + Timbre + SCF* & *SFM + Chromagram.*
2. Dengan membangun dua model, model *valence* dan *arousal* kinerja pemetaan menghasilkan akurasi tertinggi 63%
3. Menggunakan *Gaussian Process Regression* dengan *kernel RationalQuadratic* dapat memprediksi titik aktual dari sebuah lagu dalam bidang model Russel dengan nilai R^2 sebesar 0.68 untuk model *arousal* dan 0.38 untuk model *valence.*
4. Telah berhasil dibangun model *valence* dan *arousal* yang dapat memetakan sebuah lagu ke dalam bidang emosi model Russell dengan performa model *arousal* 0.68 dan model *valence* 0.38, serta akurasi pemetaan 63%

6.2 Saran

Saran yang diberikan untuk pengembangan pemetaan emosi sebuah lagu berdasarkan *Russell's two-dimensional model* menggunakan *Gaussian Processes*, yaitu:

1. Memperhatikan persebaran lagu setiap *cluster*.
2. Melakukan eksplorasi parmeter dari *Gaussian process* seperti *optimizer* dan *kernel.*
3. Melakukan eksporasi dalam representasi *vector feature* yang akan merepresentasikan setiap *window* dari lagu.
4. Melakukan eksplorasi untuk *low level feature*
5. Mencari bagian *chorus* dari lagu sebagai bagian lagu yang representatif.
6. Menggunakan teknik *oversampling* dataset dengan mempertimbangkan nilai *valence-arousal.*

DAFTAR PUSTAKA

[1] A. K. Datta, S. S. Solanki, R. Sengupta, S. Chakraborty, K. Mahto and A. Patranabis, Signal Analysis of Hindustani Classical Music, Springer Singapore, 2017.

[2] C. Laurier and P. Herrera, "Automatic Detection of Emotion in Music," in *Machine Learning*, IGI Global, 2012, pp. 1330-1354.

[3] Y. E. Kim, E. M. Schmidt, R. Migneco, B. G. Morton, P. Richardson, J. Scott, J. A. Speck and D. Turnbull, "MUSIC EMOTION RECOGNITION: A STATE OF THE ART REVIEW," in *11th International Society for Music Information Retrieval Conference*, Utrecht, Netherlands, 2010.

[4] J. A. Russell, M. Lewicka and T. Niit, "A cross-cultural study of a circumplex model of affect.," *Journal of Personality and Social Psychology,* vol. 57, pp. 848-856, 1989.

[5] K. Markov and T. Matsui, "Speech and Music Emotion Recognition Using Gaussian Processes," in *Modern Methodology and Applications in Spatial-Temporal Modeling*, Springer Japan, 2015, pp. 63-85.

[6] M. Soleymani, M. N. Caro, E. M. Schmidt, C.-Y. Sha and Y.-H. Yang, "1000 songs for emotional analysis of music," in *Proceedings of the 2nd ACM international workshop on Crowdsourcing for multimedia - CrowdMM 13*, 2013.

[7] A. Lerch, An Introduction to Audio Content Analysis, John Wiley & Sons, Inc., 2012.

[8] S. Dixon, "ONSET DETECTION REVISITED," in *the 9th Int. Conference on Digital Audio Effects*, Montreal, Canada,

2006.

[9] W. Mendenhall, T. Sincich and N. Boudreau, Statistics for Engineering and the Sciences, CRC Press, 2016.

[10] C. K. I. (. o. E. W. Carl Edward (University of Cambridge) Rasmussen, Gaussian Processes for Machine Learning, MIT Press Ltd, 2005.

[11] ffmpeg, "About FFmpeg," [Online]. Available: https://ffmpeg.org/about.html. [Accessed 20 05 2019].

[12] B. McFee, C. Raffel, D. Liang, D. Ellis, M. McVicar, E. Battenberg and O. Nieto, "librosa: Audio and Music Signal Analysis in Python," in *Proceedings of the 14th Python in Science Conference*, 2015.

[13] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg and others, "Scikit-learn: Machine learning in Python," *Journal of machine learning research,* vol. 12, pp. 2825-2830, 2011.

[14] T. E. Oliphant, A guide to NumPy, vol. 1, Trelgol Publishing USA, 2006.

[15] J. D. Hunter, "Matplotlib: A 2D graphics environment," *Computing in science & engineering,* vol. 9, pp. 90-95, 2007.

[16] "MediaEval Benchmarking Initiative for Multimedia Evaluation," MediaEval Benchmark, [Online]. Available: http://www.multimediaeval.org/mediaeval2013/emotion2013/.

LAMPIRAN

(Halaman ini sengaja dikosongkan

BIODATA PENULIS

Tegar Satrio Utomo, lahir di Jakarta pada tanggal 11 Juli 1996. Penulis menempuh pendidikan mulai dari SDIF Al-Fikri Depok (2002-2008), SMP Negeri 3 Depok (2008-2011), SMA Negeri 3 Depok (2011-2014), dan sekarang sedang menjalani pendidikan S1 Informatika di ITS. Penulis aktif dalam organisasi dan kepanitiaan Himpunan Mahasiswa Teknik Computer (HMTC) dan Schematics. Diantaranya adalah menjadi staff Departemen Dalam Negeri HMTC ITS 2016-2017, Ketua Dewan Perwakilan Angkatan HMTC ITS 2017-2018, Wakil Koordinator 2 Biro Keamanan dan Perizinan Schematics ITS 2016 dan staff ahli Biro Keamanan dan Perizinan Schematics ITS 2017. Komunikasi dengan penulis dapat *email*: **mrsatriotegar@gmail.com**.