TUGAS AKHIR – KS184822

KLASIFIKASI *GENRE* MUSIK BERDASARKAN *MEL FREQUENCY CEPSTRUM COEFFICIENT* (MFCC) DENGAN MENGGUNAKAN METODE *SUPPORT VECTOR MACHINE* (SVM) DAN *RANDOM FOREST* (RF)

MUHAMMAD ABID AS SAROFI
NRP 062116 4000 0082

Dosen Pembimbing
Irhamah, S.Si., M.Si., Ph.D.
Adatul Mukarromah, S.Si., M.Si.

PROGRAM STUDI SARJANA
DEPARTEMEN STATISTIKA
FAKULTAS SAINS DAN ANALITIKA DATA
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA 2020

TUGAS AKHIR – KS184822

KLASIFIKASI *GENRE* MUSIK BERDASARKAN *MEL FREQUENCY CEPSTRUM COEFFICIENT* (MFCC) DENGAN MENGGUNAKAN METODE *SUPPORT VECTOR MACHINE* (SVM) DAN *RANDOM FOREST* (RF)

MUHAMMAD ABID AS SAROFI
NRP 062116 4000 0082

Dosen Pembimbing
Irhamah, S.Si., M.Si., Ph.D.
Adatul Mukarromah, S.Si., M.Si.

PROGRAM STUDI SARJANA
DEPARTEMEN STATISTIKA
FAKULTAS SAINS DAN ANALITIKA DATA
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA 2020

FINAL PROJECT – KS184822

MUSIC GENRE CLASSIFICATION BASED ON MEL FREQUENCY CEPSTRUM COEFFICIENT (MFCC) USING SUPPORT VECTOR MACHINE (SVM) AND RANDOM FOREST (RF)

MUHAMMAD ABID AS SAROFI
NRP 062116 4000 0082

Supervisors
Irhamah, S.Si., M.Si., Ph.D.
Adatul Mukarromah, S.Si., M.Si.

UNDERGRADUATE PROGRAMME
DEPARTMENT OF STATISTICS
FACULTY OF SCIENCE AND DATA ANALYTICS
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA 2020

LEMBAR PENGESAHAN

KLASIFIKASI *GENRE* MUSIK BERDASARKAN *MEL FREQUENCY CEPSTRUM COEFFICIENT* (MFCC) DENGAN MENGGUNAKAN METODE *SUPPORT VECTOR MACHINE* (SVM) DAN *RANDOM FOREST* (RF)

TUGAS AKHIR
Diajukan Untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Statistika
pada
Program Studi Sarjana Departemen Statistika
Fakultas Sains dan Analitika Data
Institut Teknologi Sepuluh Nopember

Oleh :
MUHAMMAD ABID AS SAROFI
NRP. 062116 4000 0082

Disetujui oleh Pembimbing:
Irhamah, S.Si., M.Si., Ph.D. ( )
NIP. 19780406 200112 2 002
Adatul Mukarromah, S.Si., M.Si. ( )
NIP. 19800418 200312 2 001

Mengetahui,
Kepala Departemen Statistika

Dr. Dra. Kartika Fithriasari., M.Si.
NIP. 19691212 199303 2 002

SURABAYA, JANUARI 2020

KLASIFIKASI *GENRE* MUSIK BERDASARKAN *MEL FREQUENCY CEPSTRUM COEFFICIENT* (MFCC) DENGAN MENGGUNAKAN METODE *SUPPORT VECTOR MACHINE* (SVM) DAN *RANDOM FOREST* (RF)

Nama Mahasiswa	**:**	**Muhammad Abid As Sarofi**
NRP	**:**	**062116 4000 0082**
Departemen	**:**	**Statistika-FSAD-ITS**
Dosen Pembimbing	**:**	**Irhamah, S.Si., M.Si., Ph.D.**
		Adatul Mukarromah, S.Si., M.Si.

Abstrak

Genre musik adalah pengelompokan musik sesuai dengan kemiripan antara satu musik dengan musik yang lainnya, seperti kemiripan dalam hal frekuensi musik, struktur ritmik, dan konten harmoni. Genre musik merupakan hal yang penting dalam pengelompokan musik. Pengelompokan tersebut dilakukan secara manual paa umumnya dengan cara mendengarkan secara langsung lagu tersebut. Namun, hal tersebut dapat menimbulkan ketidakefisiensian dalam mengelompokkan lagu. Tujuan dari penelitian ini adalah untuk mendapatkan metode yang memiliki performa klasifikasi terbaik diantara dua metode yang digunakan yaitu SVM dan Random Forest, dimana metode yang terbaik nantinya akan digunakan dalam pembuatan GUI. Fitur ekstraksi yang digunakan dalam penelitian adalah MFCC, karena MFCC mampu mengadaptasi pendengaran manusia. Hasil dari klasifikasi audio pada GTZAN dataset dengan menggunakan kedua metode menunjukkan bahwa Random Forest merupakan metode yang terbaik daripada Support Vector Machine karena memiliki nilai accuracy, precision, sensitivity dan Fscore yang lebih besar.

Kata kunci: MFCC, Genre Musik, Random Forest, SVM.

(Halaman ini sengaja dikosongkan)

MUSIC GENRE CLASSIFICATION BASED ON MEL FREQUENCY CEPSTRUM COEFFICIENT (MFCC) USING SUPPORT VECTOR MACHINE (SVM) AND RANDOM FOREST (RF)

Name	**:**	**Muhammad Abid As Sarofi**
Student Number	**:**	**062116 4000 0082**
Department	**:**	**Statistika-FSDA-ITS**
Supervisors	**:**	**Irhamah, S.Si., M.Si., Ph.D.**
		Adatul Mukarromah, S.Si., M.Si.

Abstract

A music genre is a category that identifies some pieces of music as belonging to a share tradition or set of conventions. Music can be divided into different genres in many different ways, such as similarity in terms of music frequency, rhytmic structure, and harmony content. Nowdays, companies use music classification to be able to give recommendations to their customers. The first step in that direction is determining music genres. Usually, the music genre categorizing is done manually by listening directly to the music. However, inefficiency became a problem that must be considered in doing music classification manually, because it will take a lot of time. Therefore, it is needed to conduct a study based on machine learning to classify music genres. In this study, two methods i.e. Support Vector Machine (SVM) and Random Forest (RF) are compared to classify music clips into different genres, whereas the best method are use to make the GUI. Mel Frequency Cepstrum Coefficient (MFCC) is used for feature extraction because it is easy to implement, robust to noise and represent frequencies that can be captured by the human ear. The study conducted on GTZAN dataset shows that audio classification using Random Forest has a higher accuracy, precision, sensitivity and Fscore than SVM.

Keywords: MFCC, Music Genre, Random Forest, SVM

(Halaman ini sengaja dikosongkan)

KATA PENGANTAR

Puji syukur penulis panjatkan atas rahmat dan hidayah yang diberikan Allah SWT sehingga penulis dapat menyelesaikan laporan Tugas Akhir yang berjudul "Klasifikasi *Genre* Musik Berdasarkan *Mel Frequency Cepstrum Coefficient* (MFCC) dengan Menggunakan Metode *Support Vector Machine* (SVM) dan *Random Forest* (RF)" dengan lancar.

Penulis menyadari bahwa Tugas Akhir ini dapat terselesaikan tidak terlepas dari bantuan dan dukungan berbagai pihak. Oleh karena itu, penulis menyampaikan terima kasih kepada:

1. Kedua orang tua, atas segala do'a, nasehat, kasih sayang, dan dukungan yang diberikan kepada penulis demi kesuksesan dan kebahagiaan penulis
2. Irhamah, S.Si., M.Si., Ph.D. dan Adatul Mukarromah S.Si., M.Si., selaku dosen pembimbing penulis dalam menyelesaikan laporan Tugas Akhir ini dengan sabar dan tak lupa memberikan semangat dan motivasi kepada penulis dalam menyelesaikan penelitian ini
3. Dr. Kartika Fithriasari selaku Kepala Departemen Statistika yang telah memberikan fasilitas, sarana, dan prasarana dan juga selaku dosen penguji yang selalu sabar dalam mengomentari serta memberikan masukan dan saran dalam penyelesaian tugas akhir
4. Dr. R. Mohamad Atok, S.Si., M.Si., selaku dosen wali penulis selama masa studi yang telah banyak memberikan saran dan arahan dalam proses belajar di Departemen Statistika
5. Dr. Bambang Widjanarko Otok selaku dosen penguji yang selalu sabar dalam mengomentari serta memberikan masukan dan saran dalam penyelesaian tugas akhir
6. Dr. Santi Wulan Purnami, S.Si., M.Si., selaku Sekretaris Departemen 1 Bidang Akademik dan Kemahasiswaan yang telah memberikan fasilitas, sarana dan prasarana

7. Seluruh dosen Statistika ITS yang telah memberikan ilmu dan pengetahuan yang tak ternilai harganya, serta segenap karyawaan Departemen Statistika ITS
8. Nadhifa Ayu Shafirra, teman seperjuangan saya yang telah berjuang bersama-sama melalui masa perkuliahan ini, dari STATION 2018, Seleksi MAWAPRES hingga pengerjaan tugas akhir
9. *Partner* satu dosen pembimbing yang saling menguatkan, Ni Luh Putu Ika Candrawengi dan segenap teman-teman seperjuangan PW 121 yang telah menjadi tempat berbagi keluh kesah penulis selama pengerjaan tugas akhir
10. Sahabat terdekat dan teman hidup di Surabaya Naufal, Reza, Jefry, Sofi, Rifqi dan Fadhli yang telah menjadi tempat curhat saya.
11. Teman-teman Statistika ITS ∑27 angkatan 2016, yang selalu memberikan dukungan kepada penulis selama ini.
12. Semua teman, relasi dan berbagai pihak yang tidak bisa penulis sebutkan namanya satu persatu yang telah membantu dalam penulisan laporan ini.

Besar harapan penulis untuk mendapatkan kritik dan saran yang membangun sehingga Tugas Akhir ini dapat memberikan manfaat bagi semua pihak yang terkait.

Surabaya, 2020

Penulis

DAFTAR ISI

Halaman

DAFTAR GAMBAR

Halaman

(Halaman ini sengaja dikosongkan)

DAFTAR TABEL

Halaman

(Halaman ini sengaja dikosongkan)

DAFTAR LAMPIRAN

Halaman

(Halaman ini sengaja dikosongkan)

BAB I
PENDAHULUAN

1.1. Latar Belakang

Musik adalah suatu cabang dari seni yang menjadikan berbagai macam suara kedalam pola-pola yang dapat dimengerti dan dipahami oleh manusia (Banoe, 2003). Jamalus (1998) mendefinisikan musik sebagai suatu karya seni dalam bentuk lagu atau komposisi musik, yang mengungkapkan pikiran dan perasaan penciptanya melalui unsur-unsur musik yaitu irama, melodi, harmoni, bentuk/struktur lagu dan ekspresi sebagai satu kesatuan. Musik sudah lama menjadi bagian dari kehidupan masyarakat Indonesia. Masuknya musik ke Indonesia diawali dengan perpindahan bangsa-bangsa, terutama Asia Tengah yang berimigrasi ke Asia Tenggara. Bangsa-bangsa tersebut singgah cukup lama di Indonesia hingga mempengaruhi masyarakat Indonesia dalam hal budaya, salah satunya adalah seni musik.

Musik pada saat itu berfungsi sebagai sarana pemujaan dan bagian dari ritual masyarakat. Seiring berjalannya waktu, musik mulai bisa dinikmati oleh semua kalangan. Musik yang dulunya dianggap sakral, lambat laun menjadi sarana hiburan yang menarik. Berbagai kegiatan sehari-hari pun tidak bisa lepas dari musik. Bahkan musik sudah mulai digunakan sebagai sarana pendidikan terutama bagi anak-anak. Saat itu, mulai banyak berkembang musik-musik khas dari berbagai daerah yang kini kita sebut sebagai musik tradisional (Harenda, 2018).

Berkembang pesatnya teknologi informasi di Indonesia menciptakan era digital yang dapat memudahkan seseorang mengakses informasi, hiburan, menjalin komunikasi, pemenuhan kebutuhan sehari-hari dan lain sebagainya lebih mudah. Teknologi informasi di Indonesia sendiri ikut berkembang pesat dimana pengguna internet di Indonesia saat ini berjumlah 132,7 juta atau 52% dari jumlah penduduk di Indonesia (Pakpahan, 2017). Pemanfaatan kemajuan teknologi berdampak pada layanan penyedia jasa sebagai peluang usaha dan industri seperti penyedia

jasa *streaming* video, musik dan TV *online*, dan lain sebagainya. Kini musik dapat dimanfaatkan dan dinikmati melalui berbagai *platform* salah satunya adalah jejaring sosial. Kebiasaan seseorang dalam mendengarkan musik mengalami perubahan yang signifikan. Asosiasi Penyelenggara Jasa Internet Indonesia (2016), menjelaskan bahwa 35,5% dari populasi pengguna internet di Indonesia mendengarkan musik melalui *streamming* secara *online*. Sehingga, generasi digital saat ini dikenal lebih memilih mendengarkan musik secara *online* melalui perangkat digital daripada cara yang lebih konvensional. Layanan *streamming* musik online sudah tidak bisa dipisahkan dari keseharian masyarakat Indonesia khususnya di daerah perkotaan. 88% penggunaan layanan *streamming music* sudah menjadi bagian integral kehidupan masyarakat Indonesia. 52% dari pengguna tersebut juga berlangganan terhadap layanan musik *online* (Dailysocial.id, 2018).

Salah satu layanan atau aplikasi yang digunakan untuk mendengarkan musik adalah Spotify. Spotify adalah layanan *streamming music digital*, *podcast*, dan video yang dapat mengakses jutaan lagu dan konten lain dari beragam artis di seluruh dunia (Spotify, 2018). Tidak hanya Spotify, Joox juga merupakan aplikasi yang dapat digunakan untuk mendengarkan musik secara *online*. Joox merupakan layanan musik *streaming* yang dapat digunakan untuk mendengarkan musik, mencari *top charts* dari sebuah lagu yang sering bereda dan lain sebagainya. (Joox, 2018). Kaonang (2016) menjelaskan bahwa sejak kehadiran Spotify di Indonesia pada tahun 2016 hanya dengan waktu tiga bulan, konsumen di Indonesia telah menghabiskan waktu hampir 20 juta jam untuk menikmati musik pada layanan Spotify. Rata-rata waktu yang dihabiskan pengguna untuk menikmati lagu di Spotify di Indonesia setiap harinya adalah 90 menit. Waktu yang sering digunakan pengguna untuk mendengarkan musik antara lain antara pukul 12 siang hingga 4 sore, dan jam 8 malam hingga 11 malam tiap harinya. *Genre* musik yang ada dalam Spotify juga beragam, seperti *pop*, *jazz*, klasik dan lain sebagainya.

Musik sendiri dapat dibedakan kedalam berbagai *genre* atau kategori. Kamus Besar Bahasa Indonesia (2016), mengartikan *genre* sebagai jenis, tipe, atau kelompok sastra atas dasar bentuknya. *Genre* musik juga dapat diartikan sebagai pengelompokan musik berdasarkan kemiripan jenis, tipe, atupun tema musik tersebut. Jenis-jenis pembagian *genre* musik berdasarkan kemiripan, jenis, tipe, ataupun tema musik tersebut. Distribusi musik secara elektronik menyebabkan katalog musik menjadi semakin besar seiring berjalannya waktu. Saat ini tersedia jutaan musik yang bisa dinikmati secara *online* dengan menggunakan servis dari penyedia *streaming* musik. *Genre* musik adalah cara yang paling umum digunakan untuk mengoragnisasi *database music digital* (Nanni, dkk., 2016). Mengaitkan *genre* dengan sebuah musik dapat membantu pendengar musik untuk menemukan musik yang dicari disebuah katalog musik yang sangat besar. Pemberian *genre* pada musik dilakukan secara manual oleh seorang ahli (Tzanetakis dan Cook, 2002). Saat ini, dengan bertambahnya jumlah musik yang beredar, pemberian *genre* secara manual untuk ditampilkan secara *online* akan membutuhkan waktu dan tenaga ahli. Pemberian *genre* secara otomatis dapat membantu, mengurangi, atau mengganti peran manusia dalam pemberian *genre* pada sebuah musik. Aliran *genre* musik dapat dibedakan kedalam tiga hal yaitu seni, populer dan tradisional. Aliran populer beranggotakan *genre* musik seperti *disco, hiphop, jazz, pop* dan *reggae,* serta beberapa *genre* lain seperti *ska*, *rhythm* dan lain sebagainya yang seperti sedang berkembang saat ini. *Genre* musik dalam aliran musik populer seperti *disco, hiphop, jazz, pop* dan *reggae* memiliki kesamaan dalam karakteritik musiknya. Hal tersebut dikarenakan, *genre-genre* tersebut bersumber dari satu *genre* yang sama kemudian terus dikembangkan, sehingga terdapat pola yang sama. (Chatman, 2018).

Fitur ekstraksi perlu diterapkan dalam melakukan klasifikasi pada data audio. Fitur ekstraksi yang digunakan pada penelitian ini adalah *Mel Frequency Cepstrum Coefficient* (*MFCC*). MFCC mempunyai kemampuan seperti pendengaran manusia karena

MFCC dapat mengadaptasi pendengaran manusia dalam bentuk digital. Keunggulan penggunaan MFCC adalah MFCC mampu menangkap karakteristik suara yang sangat penting bagi pengenalan suara atau dapat menangkap informasi yang penting dalam sinyal suara, menghasilkan data seminimal mungkin tanpa menghilangkan informasi penting yang ada di dalam sinyal dan mempunyai kemampuan seperti organ pendengaran suara dalam mengenali sinyal suara (Putra, 2011)

Panchwagh dan Katkar (2016) menggunakan 3 fitur ekstraksi yang berbeda dalam penelitian mereka terkait *audio signal* yaitu *Mel Frequency Cepstrum Coefficients (MFCC), Linear Predictive Coefficient (LPC)*, dan *Zero Crossing Rate (ZCR)*. Fitur fitur musik yang telah diekstrak tersebut diklasifikasikan dengan menggunakan metode *supervised* dan *unsupervised* (Panchwagh & Katkar, 2016). *Classifier* yang digunakan adalah *Naïve Bayes*, *Bayes Net*, J48, SMO dan *Logistic*. *Classifier* tersebut diimplementasikan menggunakan *library* Java-Weka. Hasil penelitian terebut menunjukkan bahwa *classifier* SMO dengan metode *PKIDiscretize* menghasilkan akurasi terbaik bahkan mencapai 100% saat menggunakan fitur MFCC. Perbandingan dari penggunaan tiga fitur ekstraksi yang dipakai menunjukkan bahwa MFCC memberikan hasil akurasi yang tertinggi.

Fansuri (2011) juga melakukan penelitian terkait klasifikasi *genre* musik dengan melakukan simulasi pada *genre* musiknya. Ekstraksi fitur yang digunakan adalah MFCC dengan metode yang digunakan adalah *Learning Vector Quantization* (*LVQ*). *Genre* musik yang dilakukan dalam penelitian tersebut adalah *rock*, klasik, keroncong dan *jazz* dengan durasi waktu 5, 10, 20 dan 25 detik pada masing-masing file musik. Pembagian data *training* dan data *testing* menggunakan *K-Fold Cross Validation* (*KCV*) dengan angka *Fold* dari 2 hingga 10. Penelitian dengan menggunakan fitur ekstraksi MFCC tersebut menghasilkan akurasi sebesar 93,75%.

Penelitian dalam mengklasifikasikan *audio speech* dan *non-speech* juga pernah dilakukan oleh Thambi, dkk. Penelitian tersebut menggunakan beberapa metode antara lain *ADTree*

dengan nilai akurasi sebesar 93,04%, *LADTree* dengan akurasi sebesar 93,46%, *C4.5* dengan akurasi sebesar 94,67%, *CART* dengan akuasi sebesar 94,96%, *Random Tree* dengan akurasi sebesar 93,10%, *Random Forest* dengan akurasi sebesar 96,65%, *REPTree* dengan akurasi sebesar 95,53% dan *BFTree* dengan akurasi sebesar 94,81%. Nilai akurasi tertinggi didapatkan dengan menggunakan metode *Random Forest*. (Thambi dkk, 2014). Kulyukin (2018) melakukan penelitian terkait monitoring lebah dengan menggunakan Regresi Logistik, KNN, *Random Forest* dan *SVM*. Hasil penelitian tersebut menunjukkan bahwa *Random Forest* memberikan akurasi tertinggi diantara metode yang lain dengan akurasi sebesar 99,97%. Beberapa penelitian tersebut menunjukkan bahwa *Random Forest* memberikan hasil akurasi yang baik dalam mengklasifikasi data audio.

Souli dan Lachiri (2017) juga melakukan penelitian terkait klasifikasi audio dengan menggunakan fitur *scattering* dan metode *Support Vectors Machines* (*SVM*) untuk pengawasan kesehatan. Penggunaan *Support Vector Machines* (*SVM*) dengan kernel Gaussian digunakan untuk mengklasifikasi karena mampu menangani *high-dimensional data*. Metode *Support Vectors Machines* (*SVM*) tersebut mampu menghasilkan tingkat akurasi sebesar 92,22%. Dhanalakshmi (2009) melakukan pengklasifikasian sinyal audio dalam membedakan sinyal audio tersebut kedalam beberapa kategori diantaranya adalah musik, berita, olahraga, iklan, kartun dan film. Hasil penelitian tersebut menunjukkan bahwa metode *Support Vector Machines* (*SVM*) mampu mengklasifikasikan sinyal audio dengan tingkat akurasi sebesar 92,3%. Shuiping (2011) men*design* dan mengimplementasikan sistem klasifikasi audio dengan menggunakan fitur ekstraksi berdasarkan frekwensi yaitu MFCC dengan menggunakan metode SVM. Hasil penelitian tersebut menunjukkan bahwa adanya efektifitas pada pengklasifikasian sinyal audio menggunakan metode SVM dengan nilai akurasi sebesar 90%.

Oleh karena itu, penelitian ini bertujuan untuk mengklasifikasikan genre musik dengan menggunakan fitur ekstraksi MFCC dan dengan membandingkan akurasi antara metode yang digunakan yaitu *Support Vector Machine (SVM)* dan *Random Forest* (*RF*) dengan sumber data yang digunakan adalah data sekunder yaitu GTZAN (George Tzanetakis) *dataset* yang didapatkan dari *Music Analysis, Retrieval and Synthesis for Audio Signals* (MARSYAS) berupa audio musik dengan jumlah *genre* sebanyak 5 *genre* diantaranya adalah *genre disco, hiphop, jazz, pop* dan *reggae*.

1.2. Perumusan Masalah

Berdasarkan uraian latar belakang diatas yang menunjukkan adanya permaslahan dalam penelitian mengenai *genre* musik yang sedang berkembang yang dapat di deteksi dengan melakukan klasifikasi pada penelitian. Hal tersebut menjadi persoalan karena adanya ketidakefisiensian dalam pengelompokan lagu yang dilakukan secara manual pada umumnya dengan cara mendengarkan secara langsung lagu tersebut, pasalnya hal tersebut akan menyita waktu yang lumayan banyak dan membutuhkan banyak tenaga ahli dalam memetakan lagu tersebut. Selain hal ketidakefisiensian dalam melakukan klasifikasi secara manual, hal tersebut juga dikarenakan beberapa *genre* yang berbeda memiliki karakteristik yang sama. Salah satu perusahaan yang membutuhkan kebutuhan tersebut adalah perusahaan yang bergerak di bidang industri kreatif khususnya di bidang musik dan seni seperti Spotify dan Joox. Ketepatan pengklasifikasian *genre* pada layanan seperti itu perlu diperhatikan agar penikmat musik dapat mendengarkan musik sesuai *genre*nya. Oleh karena itu, pada penelitian ini akan dilakukan klasifikasi musik atau *audio* untuk mengetahui *genre* musik tersebut dan juga dapat mengetahui metode apa yang mampu mengklasifikasikan lagu dengan baik yang dapat diketahui dengan hasil ketepatan klasifikasi yang paling tinggi diantara metode *Support Vector Machine (SVM)* dan *Random Forest (RF)* dengan ekstraksi fitur ekstraksi yang digunakan adalah *Mel Frequency Cepstrum Coefficient* (*MFCC*)

pada GTZAN (George Tzanetakis) *dataset* dari *Music Analysis, Retrieval and Synthesis for Audio Signals* (MARSYAS) dengan jumlah *genre* musik sebanyak 5 *genre* yaitu genre musik *disco*, *hiphop*, *jazz*, *pop* dan *reggae*.

1.3. Tujuan

Berdasarkan rumusan masalah pada uraian di atas, sehingga tujuan yang ingin dicapai dalam penelitian ini adalah sebagai berikut.

1. Mengklasifikasikan *genre* musik pada GTZAN (George Tzanetakis) *dataset* yang diperoleh dari *Music Analysis, Retrieval and Synthesis for Audio Signals* (MARSYAS) dengan menggunakan *Support Vector Machine (SVM)* dan *Random Forest (RF)*.
2. Mendapatkan hasil klasifikasi terbaik yang memiliki ketepatan klasifikasi tertinggi dalam hal klasifikasi *genre* musik pada GTZAN (George Tzanetakis) *dataset* yang diperoleh dari *Music Analysis, Retrieval and Synthesis for Audio Signals* (MARSYAS) antara metode *Support Vector Machine* (*SVM*) dan *Random Forest* (*RF*).
3. Membuat *Graphic User Interface* (GUI) dengan menggunakan metode yang memiliki performa klasifikasi terbaik.

1.4. Manfaat Penelitian

Manfaat yang diharapkan pada penelitian ini sebagai berikut.

1. Bagi keilmuan statistika

Dapat menjadi referensi untuk penelitian-penelitian selanjutnya dalam melakukan klasifikasi musik atau *audio* berdasarkan *genre* ataupun melakukan klasifikasi dengan menggunakan metode *Support Vector Machine (SVM)* dan *Random Forest (*RF*)*.

2. Bagi pembaca

Dapat menjadi referensi kepada pembaca, khususnya yang melakukan penelitian dalam bidang *audio signal* dengan

menggunakan fitur ekstraksi *Mel Frequency Cepstrum Coefficient* (MFCC).

3. Bagi pengembang layanan musik *streaming online*

Dapat menjadi bahan evaluasi bagi pengembang layanan *music streaming online* untuk lebih tepat dalam melakukan klasifikasi musik berdasarkan *genre*nya.

1.5. Batasan Masalah

Batasan permasalahan dalam penelitian ini yaitu data yang digunakan adalah data sekunder berupa *audio* dengan format *wav* (*waveform audio format*) dan *genre* yang digunakan hanya 5 *genre* meliputi *genre* musik *disco, hiphop, jazz, pop* dan *reggae*. Keterbatasan sumber data yang *open source* juga menjadi batasan permasalahan pada penelitian tugas akhir ini, karena data musik terdapat hak cipta apabila data tersebut diambil dari sumber lainnya seperti Youtube.

BAB II
TINJAUAN PUSTAKA

2.1. *Audio Mining*

Audio mining adalah sebuah teknik dimana konten sinyal *audio* dapat dianalisis secara otomatis. Hal tersebut paling umum digunakan dalam bidang pengenalan suara otomatis, dimana seseorang mencoba mengidentifikasi ucapan apapun dalam *audio*. Tak hanya untuk identifikasi suara saja, *audio mining* juga dapat digunakan untuk mengetahui karateristik dari suara tersebut. Salah satu kasus dalam dunia *digital* adalah untuk pengenalan masing-masing karakteristik *genre* dari sebuah musik. Untuk melakukan identifikasi tersebut perlu dilakukan ekstraksi fitur.

Ekstraksi fitur berfungsi untuk mengenal karakteristik dari suatu musik atau lagu. Ekstraksi fitur dapat dilakukan dengan dua cara yaitu berdasarkan *time domain* yaitu *energy*, *zero-crossing rate* (ZCR) dan *entropy of energy* ataupun *frequency domain* yaitu *spectral centroid and spead*, *spectral entropy*, *spectral flux*, *spectral rolloff*, MFCC dan *chroma vector* (Giannakopoulus dan Pikaris, 2014). Pada penelitian ini fitur ekstraksi yang digunakan untuk mengenal karakteristik adalah *MFCC (Mel Frequency Cepstral Coefficient)* (Tzanetakis, 2001). *MFCC* mempunyai kemampuan seperti pada pendengaran manusia, jadi *MFCC* dapat mengadaptasi pendengaran manusia dalam bentuk *digital*. Selain itu penggunaannya adalah untuk mengekstrak data sinyal suara. Keunggulan penggunaan *MFCC* adalah *MFCC* mampu menangkap karakteristik suara yang sangat penting bagi pengenalan suara atau dapat menangkap informasi yang penting dalam sinyal suara, menghasilkan data seminimal mungkin tanpa menghilangkan informasi penting yang ada di dalam sinyal dan mempunyai kemampuan seperti organ pendengaran suara dalam mengenali sinyal suara (Putra dan Resmawan, 2011).

Tahapan dalam proses *Mel Frequency Cepstral Coefficient* adalah sebagai berikut (Fayek, 2016).

1. *Pre – Emphasis*

Pre-emphasis digunakan untuk melakukan penekanan pada sinyal yang dapat memperkuat frekuensi yang tinggi. Tahapan ini berguna dalam menyeimbangkan *spectrum* frekuensi karena frekuensi yang tinggi memiliki *magnitude* lebih kecil dibandingkan dengan frekuensi yang lebih rendah. Tahapan *pre-emphasis* dapat diaplikasikan dalam sebuah sinyal dengan menggunakan persamaan sebagai berikut (Fayek, 2016).

$$y(\mathrm{g}) = x(\mathrm{g}) - \alpha x(\mathrm{g}-1) \qquad (2.1)$$

dimana:

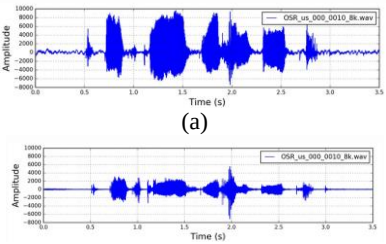
$y(\mathrm{g})$: nilai amplitudo hasil *pre-emphasis* pada waktu ke-*g*

$x(\mathrm{g})$: nilai amplitudo awal pada waktu ke-*g*

α : nilai koefisien filter

dengan α / koefisien filter yang digunakan bernilai 0,97.

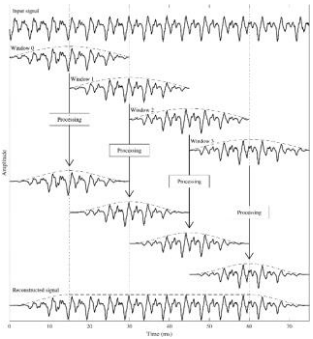
Secara visual penerapan *pre-emphasis* memberikan efek seperti pada Gambar 2.1.



Gambar 2.1 (a) Ilustrasi sinyal sebelum tahap *pre-emphasis* (b) Ilustrasi sinyal setelah tahap *pre-emphasis*.

2. *Framing*

Framing adalah tahapan membagi sinyal kedalam beberapa *frame* yang lebih kecil. *Framing* dilakukan karena frekuensi dalam sinyal berubah dari waktu ke waktu. Untuk menghindarinya, dapat dengan mengasumsikan bahwa frekuensi dalam sinyal adalah diam selama periode waktu yang sangat singkat. Ilustrasi dari tahapan *framing* dapat dilihat pada Gambar 2.2.



Gambar 2.2 Ilustrasi *framing* pada fitur ekstraksi MFCC

3. *Window*

Tahapan *window* dilakukan setelah memotong sebuah sinyal dan menjadikan sinyal tersebut kedalam beberapa *frame*. Fungsi yang diterapkan pada tahapan ini adalah *Hamming Window* pada setiap *frame*. Persamaan *Hamming Window* adalah sebagai berikut (Fayek, 2016).

$$w[n] = 0.54 - 0.46\cos\left(\frac{2\pi n}{N-1}\right) \tag{2.2}$$

dimana:

$w[n]$: nilai amplitudo hasil *windowing* pada *window length* ke-*n* dengan, $0 \le n \le N-1, N$ adalah panjang *window*.

4. *Fourier-Transform dan Power Specturm*

Tahapan *Fourier-Transform* (FFT) mengaplikasikan *N-point FFT* pada setiap *frame* untuk menghitung *spectrum* frekuensi atau juga disebut sebagai *Short-Time Fourier-Transform* (STFT), dengan nilai *N* sebesar 512. Perhitungan FFT dapat dilakukan sesuai pada persamaan berikut (Fayek, 2016).

$$FFT\left(x_k\right) = \sum_{n=0}^{N-1} x_n e^{-i2\pi kn/N} \tag{2.3}$$

dimana:

x_k : nilai FFT ke-*k* dimana, $0 \le k \le \frac{N}{2} + 1$

x_n : nilai amplitudo hasil *windowing* ke-*n*

Selanjutnya dapat dihitung *power spectrum* (peridogram) pada setiap *frame* dengan persamaan sebagai berikut.

$$P = \frac{\left|FFT\left(x_i\right)\right|^2}{N} \tag{2.4}$$

dimana x_i adalah frame ke-*i* pada sinyal *x*.

5. *Filter Banks*

Filter banks adalah tahapan dengan mengaplikasikan *triangular filters*, dengan nilai filter sebanyak 40 pada skala *mel* ke *power spectrum* untuk mengekstraksi *frequency bands*. Skala *mel* bertujuan untuk meniru persepsi suara telinga manusia yang *non-linear*, dengan menjadi lebih diskriminatif pada frekuensi yang lebih rendah dan kurang diskri-minatif pada frekuensi yang lebih tinggi. Hal tersebut dapat dilakukan dengan cara mengkonversi antara *Hertz* (*f*) dan *Mel* (*m*) menggunakan persamaan sebagai berikut (Fayek, 2016).

$$m = 2595 \log_{10}\left(1 + \frac{f_{sr}}{700}\right) \tag{2.5}$$

dimana:

f_{sr} : setengah dari nilai *sample rate*

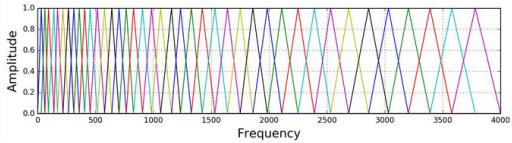
Selanjutnya dapat dilakukan konversi dari nilai *mel* ke *hertz* dengan menggunakan persamaan sebagai berikut (Fayek, 2016).

$$f = 700\left(10^{m/2595} - 1\right) \tag{2.6}$$

dimana:

m : nilai *mel scale*

Setiap *filter* dalam *filter bank* berbentuk segitiga yang memiliki respons 1 pada bagian tengah frekuensi dan menurun secara linear menuju 0 hingga mencapai frekuensi tengah dari dua filter yang berdekatan dimana responsnya adalah 0, seperti pada Gambar 2.3.



Gambar 2.3. Ilustrasi tahapan *filter banks* dalam MFCC

Tahapan tersebut dapat dimodelkan seperti persamaan berikut (Fayek, 2016).

$$H_m(k) = \begin{cases} 0, & k < f(m-1) \\ \dfrac{k - f(m-1)}{f(m) - f(m-1)}, & f(m-1) \le k < f(m) \\ 1, & k = f(m) \\ \dfrac{f(m+1) - k}{f(m+1) - f(m)}, & f(m) < k \le f(m+1) \\ 0, & k > f(m+1) \end{cases} \quad (2.7)$$

Setelah mengaplikasikan *filter bank* ke dalam *power spectrum* dari sinyal, selanjutnya dikonversi ke dalam nilai desibel dengan persamaan sebagai berikut (Fayek, 2016).

$$f_{banks}(k) = 20\log_{10} H_m(k) \quad (2.8)$$

6. *Mel-Frequency Cepstrum Coefficients (MFCC)*

Tahapan terakhir adalah menerapkan *Discrete Cosine Transform* (DCT) pada koefisien *filter bank*. Dalam *automatic speech recognition* (ASR), nilai koefisien *cepstrum* antara 2 hingga 13 yang dihasilkan untuk dipertahankan dan sisanya tidak digunakan. Persamaan *Discrete Cosine Transform* (DCT) adalah sebagai berikut (Fayek, 2016).

$$y(k) = f_{sf}\left(2\sum_{n=0}^{N-1} x_n \cos\left(\frac{\pi k(2n+1)}{2N}\right)\right) \quad (2.9)$$

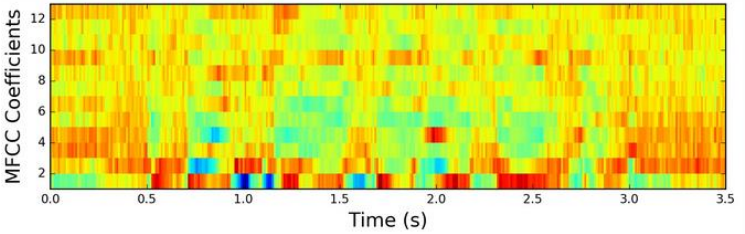
$$f_{sf} = \begin{cases} \sqrt{\dfrac{1}{4N}}, & k = 0 \\ \sqrt{\dfrac{1}{2N}}, & lainnya \end{cases} \quad (2.10)$$

dimana:

$y(k)$: nilai MFCC ke-*k*

$f_{sf}(k)$: *scaling factor*

Nilai koefisien *cepstrum* yang digunakan sebesar 12. Nilai ini digunakan karena telah mewakili perubahan cepat dalam koefisien *filter bank*. Selain itu, pada dasarnya MFCC hanya menghasilkan 39 koefisien, hanya 12 koefisien pertama saja yang merupakan nilai MFCC, sedangkan koefisien lainnya merupakan *delta* MFCC, *delta delta* MFCC, *frame energy*, *delta frame energy*, *delta delta frame energy* (Cen, 2016). Sehingga dihasilkan nilai MFCC yang dapat divisualisasikan seperti Gambar 2.4.



Gambar 2.4. Visualisasi nilai MFCC dalam bentuk *spectrogram*

Nilai MFCC merepresentasikan panjang gelombang dalam spektrum (spektral). Koefisien pertama pada MFCC merepresentasikan rata-rata *power* dalam spektrum. Koefisien kedua pada MFCC berhubungan dengan *spectral centroid*. Koefisien yang lebih tingi mewakili nilai spektral yang lebih detail, dalam prakiknya nilai koefisien MFCC ke 8-13 digunakan untuk merepresentasikan bentuk dari spektrum (Mitrovic, Zeppelzauer, & Breiteneder, 2010).

2.2. *Audio Classification*

Klasifikasi adalah salah satu metode dalam *data mining*. Dalam klasifikasi tersebut, label dari setiap kelas sudah ditentukan terlebih dahulu. Label pada kasus ini adalah *genre* musik dari masing-masing *file* audio. Proses klasifikasi sendiri merupakan proses untuk menemukan model atau membedakan kelas atau data

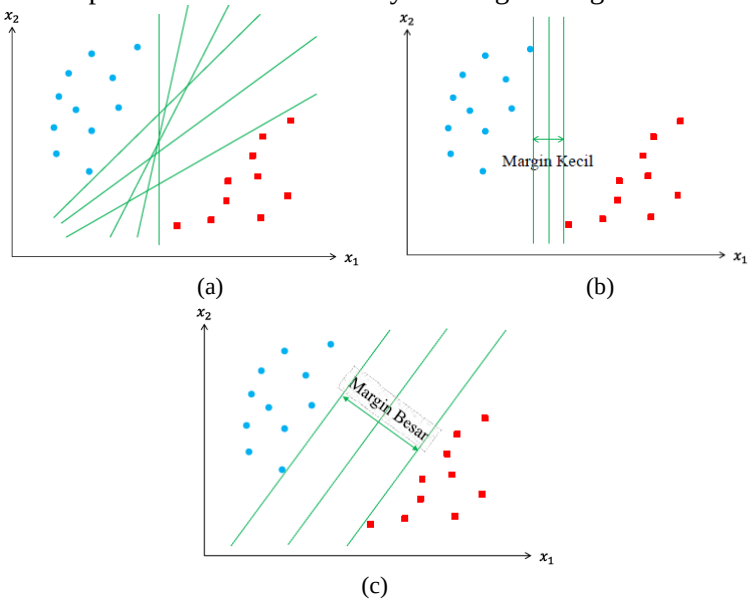
yang bisa digunakan untuk memprediksi kelas dari objek yang label kelasnya tidak diketahui. Klasifikasi merupakan bentuk analisis data yang dapat menggambarkan ekstrak model dari suatu data yang penting (Han dkk, 2012). Proses dalam klasifikasi terdiri dari *learning* model dan klasifikasi. *Learning* model adalah fase dimana data *training* akan dianalisa dengan algoritma kla-sifikasi sedangkan tahap klasifikasi adalah tahap yang memperkirakan data *testing* berdasarkan *rule* klasifikasi dan didapatkan pula hasil akurasi klasifikasinya. *Audio Classification* atau analisis klasifikasi dalam sebuah *audio* merupakan sebuah metode *supervised process* yang digunakan untuk mengklasifikasikan beberapa data kelompok berupa *audio* dengan menggunakan algoritma klasifikasi yang berbeda-beda. Pada penelitian kali ini, algoritma yang digunakan adalah *Support Vector Machine (SVM)* dan *Random Forest (RF).*

2.2.1 *Support Vector Machine* (SVM)

Pada tahun 1992 Vladimir Vapnik, Bernhrd Boser, dan Isabelle Guyon memperkenalkan *Support Vector Machine (SVM)* untuk pertama kalinya. SVM merupakan suatu algoritma untuk mencari *hyperplane* yang dapat memisahkan data antara dua kelas oleh pemisah linear secara optimal. Meskipun dalam prosesnya cenderung diperlukan waktu yang lama, akurasi yang dihasilkan oleh SVM cenderung tinggi dan tidak terjadi kasus *overfitting* (Han dkk., 2012).

SVM dapat digunakan pada data yang dapat dipisah secara linear maupun tidak dapat dipisah secara linear. Salah satu contoh kasus dasar pada klasifikasi adalah data yang bisa dipisahkan secara linier oleh suatu garis lurus. Ilustrasi data yang dapat dipisahkan oleh garis lurus dapat dilihat pada Gambar 2.5 (a), dimana terdapat banyak garis yang bisa digunakan untuk memisahkan data tersebut ke dalam kelas yang bersesuaian. Contoh garis pemisah yang dapat digunakan dapat dilihat pada Gambar 2.5 (b) dan Gambar 2.5 (c), dimana dapat diketahui bahwa pemisah pada Gambar 2.5 (c) lebih baik untuk digunakan karena

memiliki margin yang lebih besar, sehingga data dapat secara akurat dipisahkan ke dalam kelasnya masing-masing.



Gambar 2.5. (a) Ilustrasi *hyperplane* yang bisa digunakan pada data yang bisa dipisah secara linear (b) *hyperplane* yang menghasilkan margin yang kecil, (c) *hyperplane* yang menghasilkan margin yang besar

Gambar 2.5 (a) menggambarkan bahwa terdapat banyak garis pemisah yang bisa dibuat untuk memisahkan data kategori +1 (warna biru) dengan kategori -1 (warna merah). Namun akan dicari suatu garis pemisah yang terbaik, yang dapat meminimumkan kesalahan klasifikasi, yaitu garis yang menghasilkan margin terbesar. Margin adalah jarak antara *hyperplane* atau pemisah dengan data terdekat dari masing-masing kelas data. Persamaan untuk *hyperplane* dapat ditulis sebagai berikut.

$$\mathbf{W}.\mathbf{X}+b=0, \tag{2.11}$$

dengan $\mathbf{W}=\{w_1, w_2, ..., w_p\}$ adalah vektor pembobot, p adalah banyak variabel X, dan b adalah suatu konstanta atau biasa disebut dengan bias. Persmaan (2.11) dapat dimodifikasi sehingga didapat persamaan untuk setiap sisi margin berikut.

M_1: $\boldsymbol{W.X} + b \geq 1$ untuk $y_m = +1$, (2.12)

M_2: $\boldsymbol{W.X} + b \leq -1$ untuk $y_m = -1$ (2.13)

Data yang berada pada daerah M_1 akan dikategorikan kedalam kelas +1, sedangkan data yang berada pada daerah M_2 akan dikategorikan kedalam kelas -1. Jika persamaan tersebut dikalikan dengan masing-masing nilai kelasnya, y_m, maka didapat persamaan sebagai berikut.

$$y_m\left(W.X + b\right) \geq 1, \nabla_m \tag{2.14}$$

Data yang berada pada tepat pada tepi margin, yaitu M_1 dan M_2, disebut sebagai *support vectors*.

Berdasarkan persamaan (2.14) dapat diketahui bahwa jarak terdekat *hyperplane* ke tepi margin adalah $\frac{1}{||\boldsymbol{W}||}$, dimana $\left|\left|\boldsymbol{W}\right|\right|$adalah *Euclidian norm* dari $\boldsymbol{W}$, dengan rumus $\sqrt{w_1^2 + w_2^2 + \cdots + w_n^2}$. Sehingga jarak antar tepi margin adalah $\frac{2}{||\boldsymbol{W}||}$. Karena akan dicari *hyperplane* yang memberikan margin maksimal, yaitu *max* $\frac{2}{||\boldsymbol{W}||}$, hal ini kongruen dengan mencari nilai minimal dari *min*$\frac{||\boldsymbol{W}||}{2}$, atau *min* $\frac{||\boldsymbol{W}||^2}{2}$. Maka fungsi obyektif pada permasalahan ini adalah

$$\min \frac{\left\|\mathbf{W}\right\|^2}{2} \tag{2.15}$$

dengan fungsi batasan sebagai berikut.

$$\sum_{m=1}^{M} a_m \left[y_m \left(\mathbf{W.X}_m + b \right) - 1 \right] \tag{2.16}$$

Kemudian dengan menggunakan *Lagrange Multiplier*, didapat persamaan sebagai berikut.

$$L_{pd} = \frac{\left\|\mathbf{W}\right\|^2}{2} - \sum_{m=1}^{M} a_m \left[y_m \left(\mathbf{W.X}_m + b \right) - 1 \right] \tag{2.17}$$

Selanjutnya dengan menggunakan kondisi *Karush-Kuhn-Tucker* (KKT), yaitu

$$\frac{\partial L_{pd}}{\partial \mathbf{W}} = 0 \leftrightarrow \mathbf{W} - \sum_{m=1}^{M} a_m y_m X_m = 0$$
$$\mathbf{W} = \sum_{m=1}^{M} a_m y_m X_m \tag{2.18}$$

$$\frac{\partial L_{pd}}{\partial b} = 0 \leftrightarrow 0 - \sum_{m=1}^{M} a_m y_m = 0 \leftrightarrow \sum_{m=1}^{M} a_m y_m = 0 \tag{2.19}$$

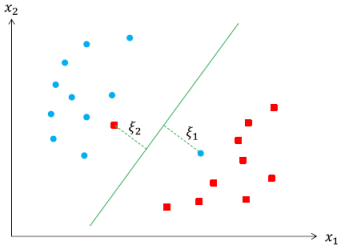
$$a_m \left[y_m \left(\mathbf{W}.\mathbf{X}_m + b \right) - 1 \right] = 0 \text{, dimana } a_m \geq 0 \tag{2.20}$$

dengan mensibstusi persamaan (2.18), (2.19) dan (2.20) ke persamaan (2.17), maka akan didapatkan persamaan sebagai berikut,

$$L_d = -\frac{1}{2} \sum_{m_1=1}^{M} \sum_{m_2=1}^{M} a_{m_1} a_{m_2} y_{m_1} y_{m_2} \left(\mathbf{X}_{m_1}.\mathbf{X}_{m_2} \right) + \sum_{m=1}^{M} a_m \tag{2.21}$$

dengan mensubstitusikan nilai $y_{m_1}, y_{m_2}, \boldsymbol{X}_{m_1}, \boldsymbol{X}_{m_2}$ kepersamaan (2.21), didapat suatu persamaan L_d yang kemudian akan digunakan untuk mendapatkan nilai-nilai a_m (*support vectors*) yang membuat L_d optimum dengan cara mencari turunan parsial L_d terhadap a.

Kasus data yang bisa dipisahkan secara linier merupakan kasus yang sulit untuk ditemui. Umumnya, terdapat beberapa kelas data yang berada pada daerah kelas data lainnya, kasus ini disebut *linearly non-separable* data atau data yang tidak bisa dipisah secara linier. Salah satu ilustrasi untuk kasus ini dapat dilihat pada Gambar 2.6.



Gambar 2.6. Ilustrasi kasus data yang tidak bisa dipisah secara linier.

Pencarian *hyperplane* yang optimal pada kasus ini akan memperhatikan data-data yang tidak berada pada kelasnya (*misclassification error*), yang dilambangkan dengan ξ. Sehingga persamaan (2.16) menjadi

$$y_m\left(\mathbf{W}.\mathbf{X}+b\right)\geq 1-\xi_m, \nabla_m \tag{2.22}$$

dan didapat persamaan Lagrange Multiplier sebagai berikut

$$L_{pd}=\frac{\|\mathbf{W}\|^2}{2}+C\sum_{m}^{M}\xi_m-\sum_{m=1}^{M}a_m\left[y_m\left(\mathbf{W}.\mathbf{X}_m+b\right)-1+\xi_m\right]-\sum_{m=1}^{M}\beta_m\xi_m \tag{2.23}$$

dengan *C* adalah suatu nilai pengali *Lagrange*. Kemudian dengan menggunakan kondisi *Karush-Kuhn-Tucker* (KKT), yaitu

$$\begin{gathered}\frac{\partial L_{pd}}{\partial \mathbf{W}}=0\leftrightarrow \mathbf{W}-\sum_{m=1}^{M}a_m y_m \mathbf{X}_m=0\\ \mathbf{W}=\sum_{m=1}^{M}a_m y_m \mathbf{X}_m\end{gathered} \tag{2.24}$$

$$\frac{\partial L_{pd}}{\partial b}=0\leftrightarrow 0-\sum_{m=1}^{M}a_m y_m=0\leftrightarrow \sum_{m=1}^{M}a_m y_m=0 \tag{2.25}$$

$$\frac{\partial L_{pd}}{\partial \xi_m}=0\leftrightarrow 0+C-a_m-\beta_m=0\leftrightarrow C=a_m+\beta_m \tag{2.26}$$

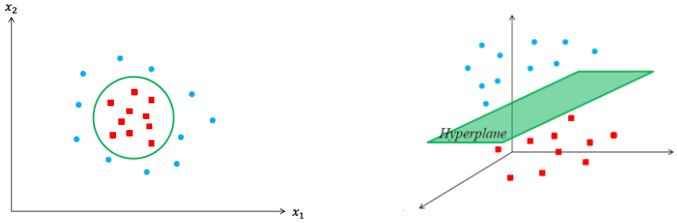
dengan mensubstitusi persamaan (2.24), (2.25) dan (2.26) ke persamaan (2.23), didapat persamaan sebagai berikut.

$$L_d=-\frac{1}{2}\sum_{m_1=1}^{M}\sum_{m_2=1}^{M}a_{m_1}a_{m_2}y_{m_1}y_{m_2}\left(\mathbf{X}_{m_1}.\mathbf{X}_{m_2}\right)+\sum_{m=1}^{M}a_m \tag{2.27}$$

Menggunakan langkah yang sama pada kasus data yang bisa terpisah secara linier, yaitu mensubstitusikan nilai $y_{m_1}, y_{m_2}, \boldsymbol{X}_{m_1}, \boldsymbol{X}_{m_2}$ ke persamaan (2.27), didapat suatu persamaan L_d yang kemudian akan digunakan untuk mendapatkan nilai-nilai a_m (*support vectors*) yang membuat L_d optimum dengan cara mencari turunan parsial L_d terhadap a.

Pada kasus nyata, sangat jarang dijumpai data yang dapat terpisah secara linier, oleh karena itu digunakan suatu fungsi kernel

(Φ) untuk memetakan data ke dalam ruang vektor yang berdimensi tinggi, seperti yang telah diilustraksikan pada Gambar 2.7, sehingga data hasil transformasi bisa terpisah secara linier.



(a) (b)

Gambar 2.7. (a) Ilustrasi kasus data non-linear, (b) hasil transformasi data beserta *hyperplane* yang didapat.

Setelah dilakukan transformasi, selanjutnya menemukan *support vector* dari data yang sudah di transformasi ke ruang yang berdimensi tinggi, yang bisa didapat menggunakan pengembangan dari persamaan (2.21) atau (2.27), yaitu

$$L_d = -\frac{1}{2}\sum_{m_1=1}^{M}\sum_{m_2=1}^{M} a_{m_1} a_{m_2} y_{m_1} y_{m_2} \left(\Phi\left(X_{m_1}\right).\Phi\left(X_{m_2}\right)\right) + \sum_{m=1}^{M} a_m \qquad (2.28)$$

dengan $\Phi(\boldsymbol{X}_{m_1})$ atau $\Phi(\boldsymbol{X}_{m_2})$ adalah data hasil transformasi. Namun, transformasi Φ tidak dapat diketahui dan sangat sulit dipahami, sehingga perhitungan *dot product* dapat secara implisit digantikan oleh fungsi kernel, sehingga didapat,

$$L_d = -\frac{1}{2}\sum_{m_1=1}^{M}\sum_{m_2=1}^{M} a_{m_1} a_{m_2} y_{m_1} y_{m_2} K\left(\mathbf{X}_{m_1}.\mathbf{X}_{m_2}\right) + \sum_{m=1}^{M} a_m \qquad (2.29)$$

dengan $K(\boldsymbol{X}_{m_1}.\boldsymbol{X}_{m_1})$ adalah fungsi kernel yang digunakan. Beberapa fungsi kernel yang akan digunakan pada penelitian ini adalah fungsi kernel linier, fungsi kernel polynomial dengan parameter *degree*, dan fungsi kernel radias basis dengan parameter gamma. Rincian rumus dan parameter yang digunakan pada setiap fungsi kernel dapat dilihat pada Tabel 2.1.

Tabel 2.1. Fungsi Kernel yang Umum digunakan pada SVM

Fungsi Kernel	Rumus $K(\mathbf{X}_{m_1}.\mathbf{X}_{m_1})$
Linear	$1+\mathbf{X}_{m_1}.\mathbf{X}_{m_2}{}^{T}$
Radial Basis	$\exp\left(-\gamma\left\|\mathbf{x}-\mathbf{x}_i\right\|^2\right)$

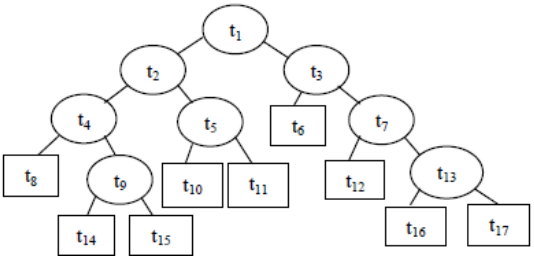
Pada awalnya SVM dikembangkan untuk persoalan klasifikasi dua kelas, kemudian dikembangkan kembali untuk klasifikasi multikelas (Santosa, 2007). Dalam klasifikasi multikelas, *hyperplane* yang terbentuk lebih dari satu. Salah satu metode pendekatannya adalah *One-Againts-All* (OAA, atau disebut juga sebagai *One-versus-Rest*). Penentuan kelas dari suatu data ditentukan berdasarkan nilai terbesar dari *hyperplane* seperti pada persamaan (2.30).

$$\hat{y} = \arg\max_{i=1,\ldots,k}\left(\sum_{m=1}^{M} a_m y_m K\left(\mathbf{X}_{m_1}\bullet\mathbf{X}_{m_2}\right)+b_i\right) \qquad (2.30)$$

2.2.2 *Random Forest* (RF)

Pada pembahasan mengenai metode *Random Forest* juga akan dijelaskan mengenai CART, pembentukan pohon klasifikasi, pemangkasan pohon klasifikasi, penentuan pohon klasifikasi optimal, pengertian *random forest* dan karakteristik *random forests*.

CART (*Classification and Regression Tree*) adalah pendekatan model nonparametrik yang dapat menjelaskan variabel respon yang dipengaruhi oleh variabel prediktor yang bersifat kontinu maupun kategorik. Data *dependent* tergantung dari partisi serangkaian *node* yang bercabang ke kanan dan ke kiri dapat disebut simpul anak (*child nodes*) yang berasal dari simpul utama (*parent node*). Setelah partisi telah berhenti, *child nodes* disebut sebagai *terminal nodes* (Zheng dkk., 2009).



Gambar 2.8. Ilustrasi Pohon Klasifikasi

Ilustrasi pohon klasifikasi dapa dilihat pada Gambar 2.8. Simpul awal yang merupakan variabel terpenting dalam menduga kelas amatan disebut sebagai simpul utaman (*parent node*) dengan notasi t_1, simpul dalam (*internal nodes*) dinotasikan sebagai simpul akhir (*terminal nodes*) dinotasikan dengan $t_6, t_8, t_{10}, t_{11}, t_{12}, t_{14}, t_{15}, t_{16}$ dan t_{17} dimana setelahnya tidak ada lagi pemilihan. Setiap simpul berada pada kedalaman (*depth*) tertentu dimana t_1 berada pada kealaman 1, t_2 dan t_3 berada pada kedalaman 2, dan begitu seterusnya hingga t_{14}, t_{15}, t_{16} dan t_{17} yang berada pada kedalaman 5.

Proses pembentukan pohon klasifikasi terdiri atas 3 tahapan, yaitu pemilihan, penentuan simpul terminal dan penandaan label kelas. Pada tahap pemilihan (*classifier*) data yang digunakan adalah sampel data *training/learning* (*L*) yang dipilah berdasarkan aturan pemilihan dan kriteria *goodness of split*. Himpunan bagian yang dihasilkan dari proses pemilihan harus lebih homogen dibandingkan simpul induknya. Hal ini dapat dilakukan dengan mendefinisikan fungsi keheterogenan simpul (*impurity* atau *imp(t)*). Fungsi heterogenitas yang umum digunakan adalah Indeks Gini.

Metode ini memiliki kelebihan yaitu proses perhitungan yang sederhana dan relatif cepat, serta mudah dan sesuai untuk diterapkan dalam berbagai kasus (Breiman dkk., 1993). Fungsi Indeks Gini dituliskan dalam persamaan berikut.

$$imp(t) = \sum_{i,j=1} p(j \mid t)\, p(i \mid t), i \neq j \qquad (2.31)$$

dengan $p(j \mid t)$ adalah proporsi kelas *j* pada simpul *t* dan $p(i \mid t)$ adalah proporsi kelas *i* pada simpul *t*. Selanjutnya menentukan kriteria *goodness of split* $(\phi(s,t))$ untuk melakukan evaluasi pemilah dari pemilah *s* pada simpul *t*. *goodness of split* $(\phi(s,t))$ didefinisikan sebagai penurunan heterogenitas sebagai berikut.

$$(\phi(s,t)) = imp(st) = imp(t) - p_L imp(t_L) - p_R imp(t_R) \qquad (2.32)$$

dengan

$imp(t)$ = fungsi heterogenitas pada simpul *t*

p_L = proposi pengamatan simpul kiri

p_R = proposi pengamatan menuju simpul kanan

$imp(t_L)$ = fungsi heterogenitas pada simpul anak kiri

$imp(t_R)$ = fungsi heterogentias pada simpul anak kanan

$$\Delta i(s^*, t_1) = \max_{s \subset S} \Delta i(s,t) \qquad (2.33)$$

Pemilah yang menghasilkan $\phi(s,t)$ lebih tinggi merupakan pemilah terbaik karena mampu mereduksi heterogenitas. Pengembangan pohon dilakukan dengan pencarian pemilah yang mungkin pada simpul t_j yang kemudian akan dipilah menjadi t_2 dan t_3 oleh pemilah s^* dan begitu seterusnya.

Tahapan selanjutnya yaitu menentukan simpul terminal. Suatu simpul *t* akan menjadi simpul terminal atau tidak, akan dipilih kembali bila pada simpul *t* tidak terdapat penurunan keheterogenan secara berarti atau adanya batasan minimum *n* seperti halnya terdapat satu pengamatan pada tiap simpul anak. Jumlah minimum dalam suatu terminal akhir umumnya adalah 5, dan apabila hal itu terpenuhi maka pengembangan pohon dihentikan (Breiman dkk, 1993).

Selanjutnya, dilakukan penandaan label kelas pada terminal *nodes* berdasarkan aturan jumlah terbanyak. Label kelas simpul terminal *t* adalah j_0 yang memberi nilai dugaan kesalahan pengklasifikasian simpul *t* terbesar. Proses pembentukan pohon klasifikasi berhenti saat terdapat hanya satu pengamatan dalam tiap-tiap simpul anak atau adanya batasan minimum *n*, semua pengamatan dalam tiap simpul anak identik dan adanya batasan jumlah level/kedalaman pohon maksimal.

$$p(j_0 \mid t) = \max_j p(j \mid t) = \max_j \frac{N_j(t)}{N(t)} \qquad (2.34)$$

dengan $N_j(t)$ merupakan banyaknya amatan kelas *j* pada terminal *nodes t* dan $N(t)$ merupakan jumlah total pengamatan dalam terminal *node t*. Label kelas untuk terminal *node t* adalah j_0 yang memberikan nilai dugaan kesalahan pengklasifikasian pada simpul *t* paling kecil sebesar $r(t) = 1 - \max_j p(j \mid t)$.

Setelah pohon klasifikasi terbentuk, perlu juga dilakukan pemangkasan pohon kla-sifikasi. Bagian pohon yang kurang penting dilakukan pemangkasan sehingga didapatkan pohon klasifikasi yang optimal. Pemangkasan didasarkan pada suatu penilaian ukuran sebuah pohon tanpa mengorbankan kebaikan ketepatan melalui pengurangan simpul pohon sehingga dicapai ukuran pohon yang layak. Ukuran pemangkasan yang digunakan untuk memperoleh ukuran pohon yang layak tersebut adalah *cost complexity minimum* (Lewis, 2000).

$$R_\alpha(t) = R(t) + \alpha|T| \qquad (2.35)$$

dimana

$R(t)$: *resubtitusion estimate*

α : kompleksitas parameter (*complexity parameter*)

$|T|$: ukuran banyaknya simpul terminal pohon T

Setelah dilakukan pemangkasan agar dicapai ukuran pohon yang layak, selanjutnya dapat dilakukan penentuan pohon klasifikasi optimal. Ukuran pohon yang terlalu besar akan

menyebabkan nilai *cost complexity* yang tinggi karena struktur data yang digambarkan cenderung kompleks sehingga perlu dipilih pohon optimal yang berukuran sederhana tetapi memberikan nilai penduga pengganti yang cukup kecil. Bila *R(T)* dipilih sebagai penduga terbaik, maka akan cenderung dipilih pohon yang besar, sebab pohon yang semakin besar akan membuat nilai *R(T)* semakin kecil.

Gabungan dari pohon klasifikasi (CART) yang saling independen yang berasal dari distribusi yang sama melalui proses *voting* (jumlah terbanyak) untuk memperoleh prediksi klasifikasi adalah metode klasifikasi *random forests*. *Random forests* merupakan pengembangan dari metode *ensemble* yang pertama kali dikembangkan oleh Leo Breiman (2001) yang digunakan untuk meningkatkan ketepatan klasifikasi. Proses pengacakan *random forests* untuk membentuk pohon klasifikasi tidak hanya dilakukan untuk data sampel saja melainkan juga pada pengambilan variabel prediktor. Sehingga, proses ini akan menghasilkan kumpulan pohon klasifikasi dengan ukuran dan bentuk yang berbeda-beda. Hasil yang diharapkan adalah suatu kumpulan pohon klasifikasi yang memiliki korelasi kecil antar pohon. Korelasi yang kecil akan menurunkan hasil kesalahan prediksi *Random Forests* (Breiman, 2001). *Random forest* memiliki karakteristik yang dapat meminimumkan korelasi yang dapat menurunkan hasil kesalahan prediksi *random forest* (Breiman, 2001). Karakteristik *random forests* adalah sebagai berikut.

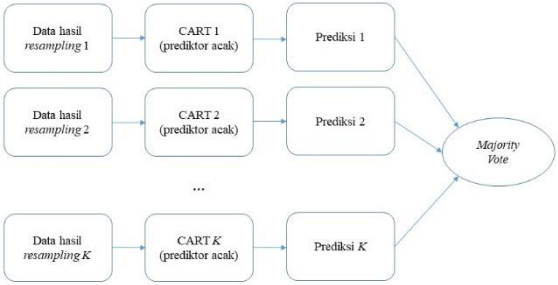
a. Keakuratan akurasi terbaik sebaik *Adaboost* dan kadang-kadang lebih baik dari *Adaboost.*
b. *Random forests* relatif kuat untuk mengatasi data *outlier* dan penganggu yang lain.
c. *Random forests* prosesnya lebih cepat daripada *bagging* atau *boosting.*
d. *Random forests* berguna dalam hal mengestimasi *error*, kekuatan, korelasi dan vari-abel yang penting.
e. *Random forests simple* selain itu juga mudah.

Adaboost (*adaptive boosting*) merupakan salah satu metode *ensemble* seperti b*agging* dan *random forests*. *Bagging* dan *random forests* mendapatkan banyak pohon dari anak gugus data yang berbeda-beda hasil dari proses *bootstrap*. Akan tetapi pada *adaboost*, setiap kali pembuatan pohon, data yang digunakan tetap seperti semula tetapi memiliki sebaran bobot yang berbeda dalam setiap iterasi. Penggunaan bobot juga dilakukan pada saat proses penggabungan dugaan akhir dari banyak pohon yang dihasilkan (Sartono dkk, 2010).

Pada *random forest* pemilihan pemilah hanya melibatkan beberapa variabel prediktor yang terambil secara acak. Algoritma *random forest* dijelaskan sebagai berikut.

a. Mengambil *n* data sampel dari *dataset* awal dengan menggunakan teknik *resampling bootstrap* dengan pengambilan.
b. Menyusun pohon klasifikasi dari setiap *dataset* hasil *resampling bootstrap*, dengan penentuan pemilah terbaik didasarkan pada variabel prediktor yang diambil secara acak. Jumlah variabel yang diambil secara acak dapat ditentukan melalui perhitungan $\log_2(Z+1)$ dimana *Z* adalah banyaknya variabel prediktor (Breiman, 2001) atau $\sqrt{Z}$ (Genuer dkk, 2009).
c. Melakukan prediksi klasifikasi data sampel berdasarkan pohon klasifikasi yang terbentuk.
d. Mengulangi prediksi klasifikasi data sampel berdasarkan pohon klasifikasi yang terbentuk. Pengulangan dilakukan sebanyak *K* kali.
e. Melakukan prediksi klasifikasi data sampel akhir dengan mengkombinasikan hasil prediksi pohon klasifikasi yang diperoleh berdasarkan aturan *majority vote*.

Secara garis besar algoritma *random forests* tersebut dapat dijelaskan pada Gambar 2.9.



Gambar 2.9. Algoritma *Random Forests*

Dalam analisis dengan menggunakan metode *random forests* dimulai dari pengambilan data dengan teknik *resampling bootstrap*. *Bootstrap* adalah suatu metode yang dapat bekerja tanpa membutuhkan asumsi distribusi karena sampel asli digunakan sebagai populasi. *Bootstrap* pertama kali diperkenalkan oleh Efron pada tahun 1979 yang digunakan untuk mencari distribusi sampling dari suatu estimator dengan prosedur *resampling* dengan pengembalian dari data asli (Sungkono, 2013). Berikut adalah algoritma dari *resampling bootsrap*.

a. Mengkonstruksi distribusi empiris $\hat{F}_n$ dari suatu sampel dengan memberikan probabilitas 1/*n* pada setiap X_i dimana i=1, 2,…, n.
b. Mengambil sampel *bootstrap* berukuran *n* secara random dengan pengembalian dari distribusi empiris $\hat{F}_n$ sebut sebagai sampel *bootstrap* pertama X^{*1}.
c. Menghitung statistik $\hat{\theta}$ yang diinginkan dari sampel *bootstrap* X^{*1} sebut sebagai $\hat{\theta}_1^*$.
d. Mengkontruksi langkah b dan c hingga *B* kali diperoleh $\hat{\theta}_1^*, \hat{\theta}_2^*, ..., \hat{\theta}_B^*$.

e. Mengkontruksi suatu distribusi probabilitas dari $\hat{\theta}_B^*$ dengan memberikan probablitias 1/*B* pada setiap $\hat{\theta}_1^*, \hat{\theta}_2^*, ..., \hat{\theta}_B^*$. Distribusi tersebut merupakan estimator *bootstrap* untuk distribusi sampling $\hat{\theta}$ dan dinotasikan dengan $\hat{F}^*$.

f. Pendekatan estimasi *bootsrap* adalah $\hat{\theta}^* = \sum_{b=1}^{B} \hat{\theta}_b^* \frac{1}{B}$.

2.3. Evaluasi Hasil Klasifikasi

Tahapan evaluasi adalah tahapan untuk mengetahui tingkat akurasi dan kinerja dari hasil klasifikasi. Pengukuran ketepatan klasifikasi dilakukan untuk melihat performa klasifikasi yang telah dilakukan. Dalam mengukur ketepatan klasifikasi, perlu diketahui jumlah pada setiap kelas prediksi dan kelas aktual yang terdiri dari *TP* (*True Positive*) yaitu jumlah *genre* musik yang tepat diprediksi dalam kelas genre musik yang sama, *TN* (*True Negative*) yaitu *genre* musik lainnya tepat terprediksi dalam kelas *genre* musik lainnya, *FP* (*False Positive*) yaitu *genre* musik lainnya yang terprediksi dalam kelas *genre* musik, dan *FN* (*False Negative*) yaitu *genre* musik yang terprediksi dalam kelas *genre* musik lainnya. Nilai *TP* dan *TN* digambarkan pada Tabel 2.2 dengan simbol n_{11}, n_{22}, n_{33}, n_{44}, dan n_{55}.

Tabel 2.2. *Confusion Matrix Multiclass*

Kelas Aktual	**Kelas Prediksi**					**Total**
	C_1	C_2	C_3	C_4	C_5	
C_1	n_{11}	n_{12}	n_{13}	n_{14}	n_{15}	$n_{1.}$
C_2	n_{21}	n_{22}	n_{23}	n_{24}	n_{35}	$n_{2.}$
C_3	n_{31}	n_{32}	n_{33}	n_{34}	n_{35}	$n_{3.}$
C_4	n_{41}	n_{42}	n_{43}	n_{44}	n_{45}	$n_{4.}$
C_5	n_{51}	n_{52}	n_{53}	n_{54}	n_{55}	$n_{5.}$
Total	$n_{.1}$	$n_{.2}$	$n_{.3}$	$n_{.4}$	$n_{.5}$	$N..$

Tabel 2.3 *Confusion Matrix*

Kelas Aktual	Kelas Prediksi	
	Positif	Negatif
Positif	*TP*	*FN*
Negatif	*FP*	*TN*

Pada kasus *multiclass* kinerja klasifikasi dapat diukur dengan menggunakan *accuracy*, *precision*, *sensitivity*, dan *Fscore*. *Accuracy* adalah banyaknya pengamatan yang terklasifikasi secara tepat. *Fscore* didapatkan dari nilai kombinasi antara *precision* dan *sensitivity*. *Precision* adalah banyaknya pengamatan yang tepat terprediksi positif dari keseluruhan dengan hasil prediksi positif, sedangkan *sensitivity* adalah banyaknya pengamatan yang tepat diklasifikasikan sesuai kategorinya (Sokolova dan Lapalme, 2009). Perhitungan *accuracy*, *precision*, *sensitivity*, dan *Fscore* dapat dilakukan dengan menggunakan persamaan (2.36), (2.37), (2.38), dan (2.39) dengan tabel yang menjelaskan *confussion matrix* untuk nilai TP, TN, FN dan FP seperti pada Tabel 2.3.

$$accuracy = \frac{\sum_{k=1}^{K} \frac{TP_k + TN_k}{TP_k + FN_k + FP_k + TN_k}}{K} \tag{2.36}$$

$$sensitivity = \frac{\sum_{k=1}^{K} \frac{TP_k}{TP_k + FN_k}}{K} \tag{2.37}$$

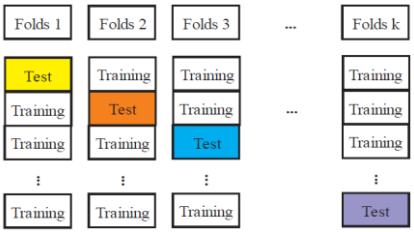
$$precision = \frac{\sum_{k=1}^{K} \frac{TP_k}{TP_k + FP_k}}{K} \tag{2.38}$$

$$F = \frac{2(precision \times sensitivity)}{precision + sensitivity} \tag{2.39}$$

2.4. *K-Fold Cross Validation*

K-fold cross validation adalah salah satu metode yang digunakan untuk mempartisi data menjadi data *training* dan data

testing. Metode ini banyak digunakan peneliti karena dapat mengurangi bias yang terjadi dalam pengambilan sampel. *K-fold cross validation* secara berulang-ulang membagi data menjadi data *training* dan data *testing*, dimana setiap data mendapat kesempatan menjadi data *testing* (Gokgoz dan Subasi, 2015). *K* merupakan besar angka partisi data yang digunakan untuk pembagian *training* dan *testing*. Ilustrasi pembagian data menggunakan *K-fold cross validation* terdapat pada Gambar 2.10.



Gambar 2.10. Ilustrasi Pembagian Data

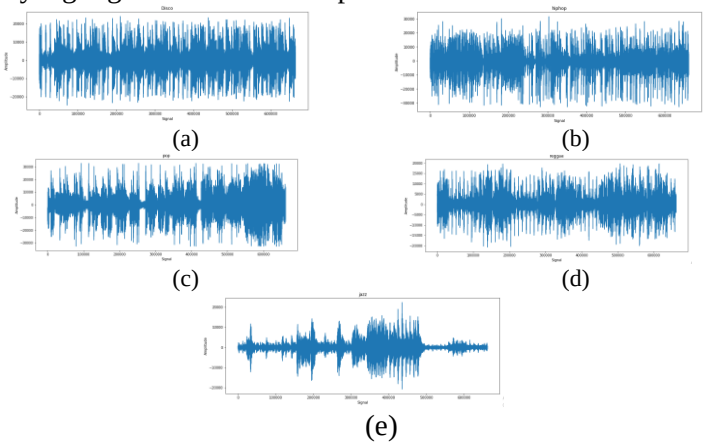
2.5. Musik

Musik adalah cabang seni yang membahas dan menetapkan berbagai suara kedalam pola-pola yang dapat dimengerti dan dipahami manusia (Banoe, 2003). Kamus Besar Bahasa Indonesi, menjelaskan *genre* adalah jenis, tipe, atau kelompok sastra atas dasar bentuknya. *Genre* musik merupakan pengelompokan musik berdasarkan kemiripan jenis, tipe, atupun tema musik tersebut. Jenis-jenis pembagian *genre* musik berdasarkan kemiripan, jenis, tipe, ataupun tema musik tersebut. Salah satu jenis seni musik yang kita kenal ialah musik *popouler* (pop). Musik *populer* sendiri memiliki ba-nyak jenis atau biasa disebut dengan *genre*. Selain pop, jenis musik yang lain ada seperti klasik, *jazz, blues, gospel, RnB, funk, rock, metal, electronic, reggae, hip hop* dan *country*. Masing-masing *genre* memiliki karakteritik masing-masing, seperti misalnya musik klasik yang merujuk pada musik klasik eropa yang memiliki periode seperti barok dan memiliki alunan yang romantis (Al, 2019).

BAB III
METODE PENELITIAN

3.1. Sumber Data

Data yang digunakan dalam penelitian ini adalah data GTZAN (George Tzanetakis) *dataset* yang didapatkan dari *Music Analysis, Retrieval and Synthesis for Audio Signals* (*MARSYAS*) dengan data file berformat *wav* (*waveform audio file*). Data *audio* yang digunakan yaitu data *audio* dengan *genre* musik *disco, hiphop, jazz, pop* dan *reggae* dengan masing-maisng *genre* sebanyak 10 lagu. Spesifikasi *file audio* yang digunakan adalah memiliki format *wav channel mono* dengan *frame rates* 22050Hz. Contoh gambaran data yang digunakan tercantum pada Gambar 3.1.



Gambar 3.1. Contoh Gambaran Data Audio (a) *Genre Disco* (b) *Genre Hiphop* (c) *Genre Pop* (d) *Genre Reggae* (e) *Genre Jazz*

3.2. Variabel Penelitian

Variabel penelitian yang digunakan adalah hasil dari ekstrasi fitur MFCC pada data GTZAN (Geroge Tzanetakis) yang berupa koefisien *Mel Frequency Cepstrum*. Variabel penelitian tersebut tercantum pada Tabel 3.1.

Tabel 3.1. Variabel Penelitian

Variabel	Keterangan	Skala
$X_{i,n}$	Nilai MFCC	Interval
Y	Genre musik yang digunakan: Y_1: Genre *Disco* Y_2: Genre *Hiphop* Y_3: Genre *Jazz* Y_4: Genre *Pop* Y_5: Genre *Reggae*	Nominal

Data yang digunakan kemudian dibagi menjadi data *training* dan data *testing* dengan menggunakan *K-Fold Cross Validation.* Struktur data yang digunakan dalam penelitian ini setelah dilakukan ekstraksi fitur terdiri dari variabel prediktor yaitu MFCC dan variabel respon yaitu klasifikasi *genre* musik yang digunakan dalam penelitian ini yaitu *genre* musik *disco*, *hiphop, jazz, pop,* dan *reggae*. Struktur data pada penelitian ini seperti pada Tabel 3.2.

Tabel 3.2 Struktur Data Penelitian

Lagu ke-	No	X_1	X_2	…	X_{12}	Klasifikasi *(Y)*
1	1	$X_{1,1}$	$X_{2,1}$	…	$X_{12,1}$	Y_1
1	2	$X_{1,2}$	$X_{2,2}$	…	$X_{12,2}$	Y_1
…	…	…	…	…	…	…
…	…	…	…	…	…	…
2	300	$X_{1,300}$	$X_{2,300}$	…	$X_{12,300}$	Y_1
…	…	…	…	…	…	…
50	14.950	$X_{1,14.950}$	$X_{2,14.950}$	…	$X_{12,14.950}$	Y_5

3.3. Langkah Penelitian

Langkah penelitian yang digunakan dalam penelitian tugas akhir ini adalah sebagai berikut.

1. Melakukan studi literatur dan referensi terkait topik dan metode yang dilakukan dalam penelitian.

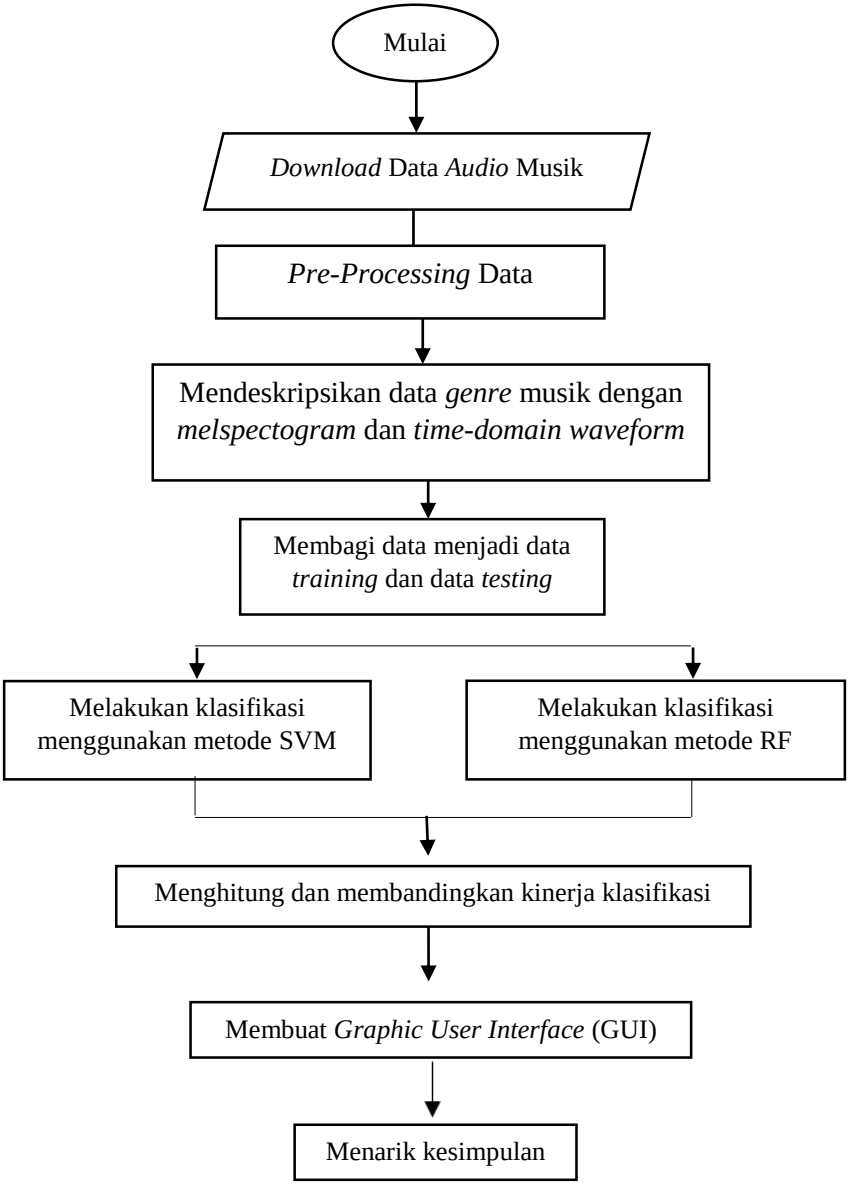
2. Mengumpulkan data dengan cara mengunduh data GTZAN (George Tzanetakis) *dataset* di laman *Music Analysis, Retrieval and Synthesis for Audio Signals* (MARSYAS) dengan nama data yang diunduh dari laman tersebut yaitu *genres.tar*.
3. Melakukan *preprocessing* data, dengan *tahap preprocessing* data meliputi:

a. Meng*import* data audio untuk mendapatkan nilai amplitudo dari masing-masing lagu.
b. Melakukan *setup* awalan dengan menggunakan durasi 3 detik pada *file audio*.
c. Melakukan tahapan *pre-emphasis* pada masing-masing *file audio*.
d. Membagi sinyal ke dalam *frame-frame* yang lebih kecil dengan rentang 25 *ms* untuk ukuran *frame* (0,025 s) dan 10 ms *stride* (15ms *overlap*).
e. Menerapkan *window* pada setiap *frame* sesuai dengan persamaan (2.2)
f. Mengaplikasikan *N-point FFT* pada setiap *frame* untuk menghitung *spectrum* frekuensi dengan nilai $N = 512$ sesuai pada persamaan (2.3) dan (2.4)
g. Mengaplikasikan *triangular filters*, dengan nilai filter sebanyak 40 pada skala *mel* ke *power spectrum* untuk mengekstraksi *frequency bands* sesuai pada persamaan (2.5), (2.6), (2.7) dan (2.8).
h. Menerapkan *Discrete Cosine Transform* (DCT) untuk menghitung nilai MFCC dengan nilai koefisien *cepstrum* sebesar 12.

4. Mengulangi langkah 3 (a) hingga 3 (h) pada semua file *audio* yang digunakan.
5. Mendeskripsikan dan memberikan gambaran umum mengenai *genre* musik yang digunakan dengan menggunakan *mel spectrogram* dan *time-domain waveform* pada *genre* musik yang digunakan.

6. Membagi data menjadi data *training* dan data *testing* dengan menggunakan *K-fold Cross-Validation (*KCV*)* dengan jumlah *K* dari 2 hingga 10.
7. Melakukan klasifikasi dengan menggunakan metode *Support Vector Machine* (SVM). Selanjutnya dihitung nilai ketepatan hasil klasifikasi dengan menggunakan metode tersebut dengan nilai *accuracy, precision, senisitivity*, dan *Fscore*.
8. Melakukan klasifikasi dengan menggunakan metode *Random Forest* (RF). Selanjutnya, melakukan evaluasi ketepatan hasil klasifikasi dengan melakukan evaluasi hasil klasifikasi dengan nilai *accuracy, precision, senisitivity*, dan *Fscore*.
9. Membandingkan kinerja klasifikasi dari metode *Support Vector Machine (*SVM) dan *Random Forest* (RF) yang telah didapatkan baik *accuracy, precision, senisitivity, dan Fscore.*
10. Membuat *graphic user interface* (GUI) dengan menggunakan metode terbaik yang telah didapatkan.
11. Memberikan kesimpulan dan saran terkait hasil analisis yang telah dilakukan.

Langkah-langkah analisis secara umum digambarkan pada diagram alir pada Gambar 3.2.



Gambar 3.2. Diagram Alir Penelitian

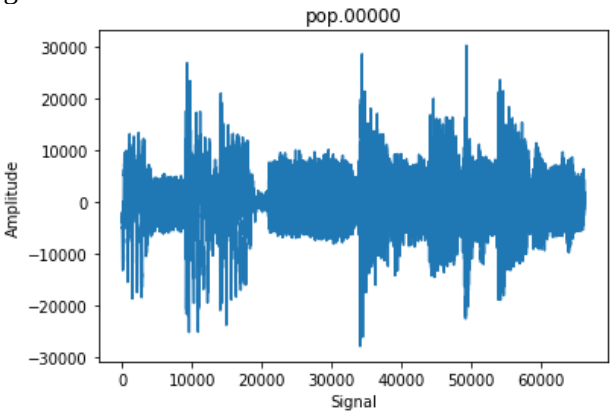
(Halaman ini sengaja dikosongkan)

BAB IV
ANALISIS DAN PEMBAHASAN

Pada penelitian ini dilakukan klasifikasi genre musik berdasarkan nilai *Mel Frequency Cepstrum Coefficient* (MFCC). Penelitian ini membandingkan dua metode yaitu *Support Vector Machine* (SVM) dan *Random Forest* (RF). Kebaikan hasil klasifikasi tersebut didapatkan dari hasil nilai *accuracy*, *precision*, *sensitivity* dan *Fscore* karena data yang digunakan *balanced*.

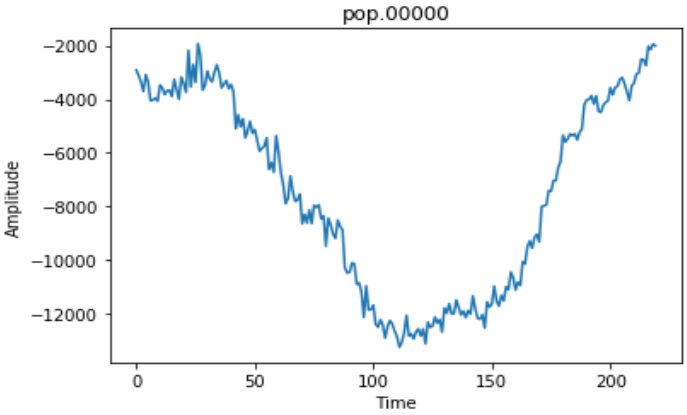
4.1. Praproses Data Audio Musik

Praproses pada data audio musik dilakukan dengan menggunakan salah satu fitur ekstraksi pada *audio mining* berdasarkan frekuensi yaitu MFCC. Sebelum melakukan tahapan praproses perlu melakukan *import* data terlebih dahulu untuk mendapatkan nilai amplitudo pada masing-masing *file* audio. Ilustrasi praproses berikut menggunakan data pop.00000 yang memiliki nilai *sampling frequency* 22500Hz dengan durasi audio yang digunakan sebesar 3 detik.



(a)

Gambar 4.1 (a) *Waveform Audio* pop.00000 3*s* (b) *Waveform Audio* pop.00000 10*ms*



(b)

Gambar 4.1 (a) *Waveform Audio* pop.00000 3*s* (b) *Waveform Audio* pop.00000 10*ms (Lanjutan)*

Gambar 4.1 (a) adalah contoh *waveform* pada data *audio* pop.00000 dengan durasi selama 3 detik, sedangkan untuk lebih jelasnya data tersebut dipotong dengan durasi 10*ms* saja untuk mempermudah secara ilustrasi mengetahui nilai amplitudo seperti pada Gambar 4.1 (b). Gambar 4.1 (b) menunjukkan bahwa dalam 10*ms* file audio dengan *sampel rate* sebesar 22050 Hz terdapat 220 sinyal, dan masing-masing sinyal memiliki nilai amplitudo yang digunakan dalam perhitungan. Nilai amplitudo pada data tersebut dapat dilihat pada Tabel 4.1.

Tabel 4.1 Nilai Amplitudo Data pop.00000

Sinyal ke-i	Amplitudo
1	-2907
2	-3121
3	-3367
…	…
66149	1872
66150	-341

Selanjutnya dapat dilakukan tahapan dari MFCC sebagai berikut.

a. *Pre-Emphasis*

Tahapan pertama adalah melakukan *pre-emphasis* dengan menggunakan data amplitudo yang telah diperoleh sebelumnya seperti pada Tabel 4.1, dengan menerapkan persamaan (2.1) maka akan didapatkan nilai amplitudo setelah tahapan *pre-emphasis* seperti pada Tabel 4.2.

Tabel 4.2. Tahapan *Pre-Emphasis* Data pop.00000

Sinyal ke-i	**Amplitudo**	**Perhitungan**	**Amplitudo *Pre-Emphasis***
1	-2907	-2907-0,97(0)	-2907
2	-3121	-3121-0,97(-2907)	-301,21
3	-3367	-3367-0,97(-3121)	-339,63
…	…	…	…
66149	1872	1872-0,97(1416)	498,48
66150	-341	-341-0,97(1872)	-2156,84

Tahapan *pre-emphasis* ini bertujuan untuk menekan sinyal yang dapat memperkuat frekuensi yang tinggi karena frekuensi yang tinggi erat hubungannya dengan *noise*. Tahapan ini berguna dalam menyeimbangkan *spectrum* frekuensi karena frekuensi yang tinggi memiliki *magnitude* lebih kecil dibandingkan dengan frekuensi yang lebih rendah, sehingga variansi dari nilai amplitudo setelah tahap *pre-emphasis* lebih kecil daripada sebelum dilakukan tahapan *pre-emphasis* dan *noise* dalam *audio* dapat berkurang. Ilustrasi proses *pre-emphasis* seperti pada Gambar 2.1.

b. *Framing*

Tahapan *framing* adalah tahapan membagi sinyal kedalam beberapa *frame* yang lebih kecil. Tahapan *framing* dilakukan karena frekuensi dalam sinyal berganti seiring waktu. Ilustrasi tahapan *framing* seperti pada Gambar 2.2. *Frame size* yang digunakan sebesar 25*ms* dan *frame stride* yang digunakan adalah sebesar 10*ms*. Jumlah *frame* yang akan digunakan dihitung menggunakan perhitungan berikut.

$$number\ of\ frames = \frac{|signal\ length - frame\ length|}{frame\ step}$$

dimana:
signal length : jumlah sinyal hasil *pre-emphasis*
frame length : *frame size* x *sample rate*
frame step : *frame stride* x *sample rate*

Nilai dari *signal length*, *frame length* dan *frame step* di substitusikan kedalam perhitungan tersebut untuk mendapatkan nilai *frame* pada lagu yang digunakan, seperti berikut.

$$number\ of\ frames = \frac{|66150 - 551|}{220} = 299$$

Jumlah *frame* yang didapatkan berdasarkan perhitungan diatas sebesar 299 *frame*, sehingga data amplitudo yang ada dari satu *file audio* akan dibagi menjadi 299 *frame* yang berisi 551 sinyal pada setiap *frame*. Tabel 4.3 menjelaskan indeks pada sinyal untuk menjelaskan hasil pembagian tersebut.

Tabel 4.3. Indeks *Framing*

Frame ke-i	**Data ke-1**	**Data ke-2**	**Data ke-3**	**…**	**Data ke-551**
1	1	2	3	…	551
2	221	222	223	…	771
…	…	…	…	…	…
299	65561	65562	65563	…	66111

Nilai dalam *frame* tersebut menunjukkan indeks sinyal dari data audio yang telah dilakukan *pre-emphasis*, dan apabila data sinyal audio tersebut dimasukkan kedalam *frame* tersebut maka akan menjadi seperti pada Tabel 4.4.

Tabel 4.4. *Framing*

Frame ke-i	**Data ke-1**	**Data ke-2**	**Data ke-3**	**…**	**Data ke-551**
1	-2907	-301,21	-439,01	…	-99,52
2	237,79	-515,27	-201,66	…	-1024,11
…	…	…	…	…	…
299	-2624,27	802,92	-3350,71	…	-3812,07

c. *Windowing*

Tahapan selanjutnya adalah melakukan *windowing* dengan menggunakan persamaan (2.2). Contoh perhitungan pada data pertama dan kedua di *frame* 1 adalah sebagai berikut.

$$w[1] = (-2907)0,54 - 0,46\cos\left(\frac{2\pi(1)}{299-1}\right) = -232,56$$

$$w[2] = (237,79)0,54 - 0,46\cos\left(\frac{2\pi(2)}{299-1}\right) = 19,0232$$

Perhitungan tersebut dilakukan pada setiap nilai amplitudo pada semua *frame* hingga didapatkan hasil *window* seperti pada Tabel 4.5.

Tabel 4.5. Hasil *Window* pada MFCC

Frame ke-i	**Data ke-1**	**Data ke-2**	**Data ke-3**	**…**	**Data ke-551**
1	-232,56	-24,105841	-27,211176	…	-7,9616
2	19,0232	-41,237067	54,947139	…	-81,9288
…	…	…	…	…	…
299	-209,9416	64,257701	-268,459091	…	-304,9656

Tahapan *windowing* bertujuan untuk mengatasai *spectral leakage* yaitu kondisi ketika sinyal yang baru (hasil dari *framing*) memiliki frekuensi yang berbeda dengan sinyal aslinya dikarenakan rendahnya *sampel rate*.

d. *Fourier-Transform* dan *Power Spectrum*

Setelah dilakukan tahapan *window*, selanjutnya menghitung nilai *power spectrum* dengan terlebih dahulu menghitung nilai FFT menggunakan persamaan (2.3) pada data pertama di *frame* pertama setelah dilakukan tahapan *window*.

$$\left|FFT\left(x_0\right)\right| = \left|\sum_{n=0}^{511} x_n e^{-\frac{i2\pi(0)n}{512}}\right|$$

$$\left|FFT\left(x_0\right)\right| = \left|-232,56e^{-\frac{i2\pi(0)(0)}{512}} + (-24,105841)e^{-\frac{i2\pi(0)(0)}{512}} + ... + x_{511}e^{-\frac{i2\pi(0)(0)}{512}}\right|$$

$$\left|FFT\left(x_0\right)\right| = 24730,010845$$

Setelah mendapatkan nilai FFT dilanjutkan dengan menghitung *power spectrum* dengan menggunakan persamaan (2.4) sebagai berikut.

$$P = \frac{\left|FFT\left(x_0\right)\right|^2}{N} = \frac{\left|24730,010845\right|^2}{512} = 1194479$$

Perhitungan *power spectrum* dilakukan pada masing-masing *frame*. Nilai *power spectrum* secara keseluruhan ditampilkan pada Tabel 4.6.

Tabel 4.6. Nilai *Power Spectrum*

Frame ke-i	**Data ke-1**	**Data ke-2**	**Data ke-3**	**…**	**Data ke-256**
1	1194479	4105771	4657025	…	3217,484
2	1772407	558162,6	225242,3	…	1303055
…	…	…	…	…	…
299	35,55298	59,08854	1858,843	…	2396813

e. *Filter Banks*

Tahapan selanjutnya adalah *filter banks*. Tahapan ini diawali dengan mengkonversi nilai frekuensi tertinggi menjadi satuan *mel* dengan persamaan sebagai berikut.

$$m = 2595\log_{10}\left(1+\frac{\left(\frac{22050}{2}\right)}{700}\right) = 3176,318435512582$$

Selanjutnya satuan *mel* terendah menjadi batas bawah dan satuan *mel* tertinggi menjadi batas atas untuk *filter* pada tahapan *filter banks*. Nilai filter yang diguanakan sebesar 40, sehingga langkah selanjutnya adalah membagi 40 nilai sama besar dengan batas bawah dan batas atas adalah nilai satuan *mel* terkecil dan nilai

satuan *mel* tertinggi (selain kedua nilai tersebut) dan selanjutnya dikonversikan menjadi satuan *Hz*. Setelah mendapatkan rentang satuan *Hz* dilanjutkan menghitung nilai pada *f(m)* yang berupa fungsi berdasarkan persamaan berikut

$$f_{(m)} = \frac{(512+1)\times \mathbf{f}}{22050}$$

Hasil perhitungan tersebut akan menghasilkan suatu *array* di mana *f(m)* adalah nilai dari fungsi tersebut pada *array* ke-*m*. Kemudian *array* tersebut digunakan untuk persamaan (2.8). Selanjutnya koefisien *filter banks* yang dihasilkan seperti pada Tabel 4.7.

Tabel 4.7. Hasil *Filter Banks*

***Frame* ke-i**	**Data ke-1**	**Data ke-2**	**Data ke-3**	**…**	**Data ke-40**
1	132,26789	113,3621	99,24711	…	123,20460
2	114,93521	107,0530	104,4885	…	143,39792
…	…	…	…	…	…
299	35,43007	65,38485	82,451	…	167,97854

f. *MFCC*

Tahapan terakhir dalam menghitung MFCC adalah menerapkan DCT sesuai pada persamaan 2.9 dan 2.10 sebagai berikut.

$$y(1) = f(k)\times 2\sum_{n=0}^{11} x_n \cos\left(\pi k\left(\frac{2n+1}{2(12)}\right)\right) = -53,763952$$

Sehingga diperoleh hasil seperti pada Tabel 4.8.

Tabel 4.8. MFCC

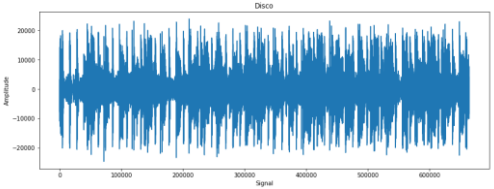
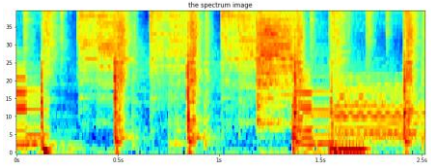
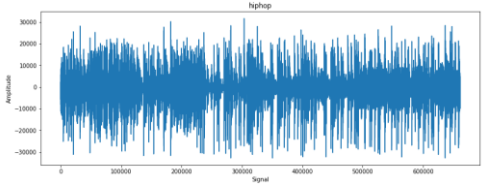
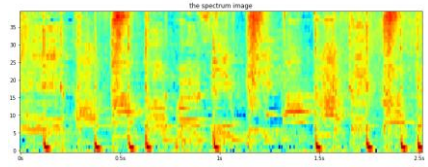
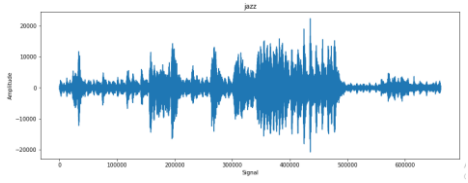
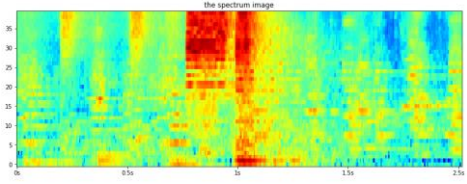
Frame ke-i	**MFCC ke-1**	**MFCC ke-2**	**MFCC ke-3**	**…**	**MFCC ke-12**
1	-53,7639	50,41832	18,83068	…	6,644582
2	-106,737	49,47155	36,81391	…	0,217695
…	…	…	…	…	…
299	-119,320	-28,5045	-52,572	…	-0,176945

4.2. Deksripsi Data Hasil *Pre-Processing*

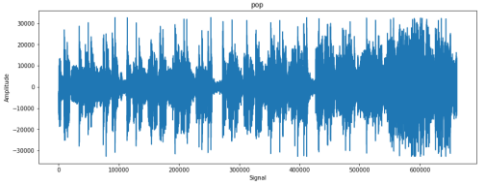
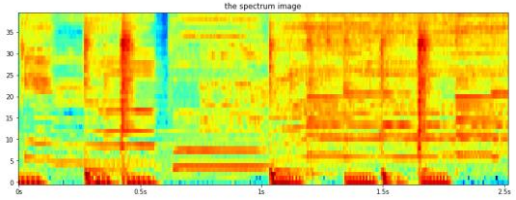
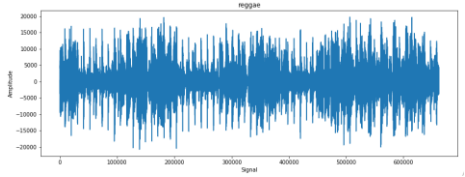
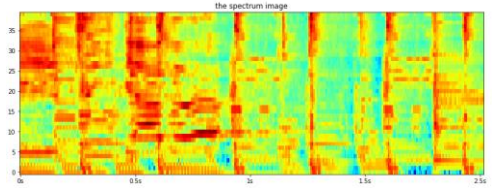
Data *Mel Frequency Cepstrum Coefficient* (MFCC) yang didapatkan dari hasil *pre-processing* adalah 12 koefisien *Mel Frequency Cepstrum Coefficient* (MFCC), dimana masing-masing koefisien *Mel Frequency Cepstrum Coefficient* (MFCC) menjelaskan hal yang berbeda. Koefisien pertama dalam *Mel Frequency Cepstrum Coefficient* (MFCC) atau disebut sebagai MFCC1 merepresentasikan rata-rata *power* dalam *spectrum.* Koefisien kedua atau disebut sebagai MFCC2 pada *Mel Frequency Cepstrum Coefficient* (MFCC) merepresentasikan *spectral centroid*, koefisien ketiga atau MFCC3 hingga ketujuh atau MFCC7 merepresentasikan *spectral centroid* yang lebih detail dan koefisien kedelapan hingga koefisien kedua belas merepresentasikan bentuk dari *spectrum*. Hasil *pre-processing* tersebut kemudian dibuat *spectogram* MFCC. Bentuk *spectogram* dari salah satu audio pada masing-masing kelas Y_1 *(disco),* Y_2 *(hiphop),* Y_4 (*pop*), dan Y_5 (*reggae*) yang digunakan terdapat pada Tabel 4.9.

Hasil *spectrogram* yang dapat dilihat pada Tabel 4.9 tersebut memperlihatkan bahwa *spectrogram* yang tergambarkan antara *genre* musik pada kelas Y_1 *(disco),* Y_2 *(hiphop),* Y_4 (*pop*), dan Y_5 (*reggae*) tidak jauh berbeda yaitu terdapat warna gelap yang menggambarkan semakin keras suatu suara pada setiap beberapa durasi *ms*, sedangkan pada Y_3 (*jazz*) yang digunakan sebagai pembeda benar-benar menggambarkan *spectrogram* yang berbeda diantara yang lain yaitu warna gelap pada MFCC cenderung lebih memusat di beberapa durasi audio. Oleh sebab itu hasil fitur ekstrksi dengan menggunakan ilia MFCC jauh leih baik, karena mampu menangkap sinyal dan membedakan berdasarkan frekuensi yang ada. Warna gelap dalam spectrogram memperlihatkan bahwa semakin keras suara dan semakin tinggi frekuensi dalam lagu tersebut.

Tabel 4.9. *Spectogram* Salah Satu Contoh Audio Setiap Kelas

Genre	***Waveform Audio***	***Spectogram***
Disco		
Hiphop		
Jazz		

Tabel 4.9. *Spectogram* Salah Satu Contoh Audio Setiap Kelas (*Lanjutan*)

Genre	***Waveform Audio***	***Spectogram***
Pop		
Reggae		

4.3. Klasifikasi Genre Musik

Setelah didapatkan data *Mel Frequency Cepstrum Coefficient* (MFCC) pada masing-masing *file audio* yang digunakan, selanjutnya dilakukan klasifikasi *genre* musik dengan menggunakan metode *Support Vector Machine* (*SVM*) dan *Random Forest* (*RF*). Berikut pembahasan lebih lanjut untuk masing-masing metode yang digunakan pada penelitian ini.

4.3.1. Klasifikasi dengan Metode *Support Vector Machine*

Metode klasifikasi pertama yang akan digunakan pada penelitian ini untuk mengklasifikasikan *genre* musik adalah *Support Vector Machine* (SVM). Pada saat analisis menggunakan metode ini, digunakan prinsip *grid search* dalam mencari kombinasi parameter yang akan digunakan pada model. Prinsip *grid search* yang digunakan mengacu pada penelitian yang telah dilakukan oleh Hsu, Chang, dan Lin (2008), yaitu mencobakan kombinasi parameter dengan pertambahan yang bersifat eksponensial. Huang, dll (2007) memberikan rentang nilai *C* yang digunakan ketika *trial and error* adalah 10^{-2} hingga 10^{4}, namun rentang nilai *C* yang digunakan pada penelitian ini hanya 10^{-2} hingga 10^{2}. Tabel 4.10 menjelaskan hasil penentuan performa terbaik pada setiap kernel yang digunakan dengan menggunakan nilai *accuracy*, *sensitivity*, *precision* dan *Fscore* pada data *training* dan data *testing*.

Tabel 4.10. Rangkuman Metode SVM Setiap Kernel

Kernel		*Linear*	*RBF*
Gamma		-	2^{-15}
C		10^{2}	10^{-1}
Accuracy	*Training*	0,7105	**0,7490**
	Testing	0,7087	**0,7468**
Sensitivity	*Training*	0,7106	**0,7489**
	Testing	0,7086	**0,7468**
Precision	*Training*	0,7105	**0,7482**
	Testing	0,7086	**0,7460**
Fscore	*Training*	0,7105	**0,7486**
	Testing	0,7087	**0,7464**

Diketahui dari Tabel 4.10 bahwa hasil yang didapat dengan menggunakan fungsi kernel RBF merupakan hasil yang terbaik karena memiliki nilai *accuracy, sensitivity*, *precision* dan *Fscore* yang paling besar pada data *training* maupun *testing*. Sehingga didapat bahwa model SVM yang terbaik adalah model dengan menggunakan fungsi kernel RBF dengan parameter *gamma* 2^{-15} dan *C* 10^{-1}. Rincian *accuracy*, *sensitivity*, *precision* dan *Fscore* yang didapat dari model SVM terbaik untuk setiap *K* dapat dilihat pada Tabel 4.11.

Tabel 4.11 menggambarkan bahwa dengan menggunakan kernel RBF pada metode SVM dan mengaplikasikan *K-Fold Cros Validation* (KCV) pada pembagian data *training* dan *testing* dengan nilai *K* dari 2 hingga 10, menunjukkan bahwa dengan menggunakan nilai *K* sebesar 10 memberikan performa yang baik pada data *training* dengan nilai *accuracy, sensitivity, precision* dan *Fscore* berturut-turut sebesar 0,7490; 0,7489; 0,7482 dan 0,7486 sedangkan pada data nilai K sebesar 7 memberikan performa terbaik pada data *testing* dengan nilai *accuracy*, *sensitivity*, *precision* dan *Fscore* berturut-turut 0,7477; 0,7478; 0,7471 dan 0,7475. Oleh karena itu, model SVM terbaik apabila menggunakan kernel RBF dengan partisi data *training* dan *testing* menggunakan KCV dengan nilai *K* sebesar 7. Fungsi *hyperplane* yang didapat dari model pada *fold* ke-4 pada pembagian *training testing* dengan nilai *K* =7, sehingga nilai M pada model *training* sebanyak 12815 yang merupakan jumlah data *training* pada *fold* ke-4 pada pembagian data *training testing* menggunakan KCV dengan nilai *K* sebesar 7 adalah sebagai berikut.

$$f(\mathbf{x}) = \sum_{m=1}^{12815} a_m y_m K\left(\mathbf{X}_{m_1} \bullet \mathbf{X}_{m_2}\right) + b_j$$

dimana fungsi kernel yang digunakan adalah fungsi kernel *radial basis* (RBF) dengan parameter *gamma* sebesar 2^{-15}, dengan rumus

$$K\left(\mathbf{x}, \mathbf{x}_i\right) = \exp\left(-\gamma \left\|\mathbf{x} - \mathbf{x}_i\right\|^2\right)$$

$$K\left(\mathbf{x}, \mathbf{x}_i\right) = \exp\left(-2^{-15} \left\|\mathbf{x} - \mathbf{x}_i\right\|^2\right)$$

sehingga fungsi *hyperplane* yang didapat menjadi

$$f(\mathbf{x})_{\mathrm{j}} = \sum_{m=1}^{12815} a_m y_m \exp\left(-2^{-15}\left\|\mathbf{x}-\mathbf{x}_i\right\|^2\right) + b_j$$

dengan b_j merupakan nilai *bias* pada masing-masing model SVM, α_m merupakan matriks koefisien dari *support vector* dan y_m adalah kelas *support vector* dimana $\alpha_m\, y_m$ berukuran (4;12815), $K\left(\mathbf{x},\mathbf{x}_i\right)$ adalah matriks *support vector* yang berukuran (12815:12). Jika didapat nilai $f(x) \geq 0$ maka *frame audio* tersebut akan dikategorikan memiliki kelas 'positif', namun jika nilai dari $f(x) \leq 0$ maka *frame audio* tersebut akan dikategorikan memiliki kelas 'negatif'. Nilai dari vektor $\boldsymbol{x}$ di substitusi sebanyak *j* model yang terbentuk dan penentuan kelas akhir dengan menggunakan persamaan berikut.

$$\hat{y} = \arg\max_{K=1,\ldots,K} \left(f(\mathbf{x})_j\right)$$

Tabel 4.11 Performa Metode SVM setiap *K*

K	*Accuracy*		*Sensitivity*		*Precision*		*Fscore*	
	Training	***Testing***	***Training***	***Testing***	***Training***	***Testing***	***Training***	***Testing***
2	0,7297	0,7256	0,7295	0,7258	0,7293	0,7256	0,7294	0,7257
3	0,7401	0,7363	0,7401	0,7363	0,7396	0,7358	0,7398	0,7361
4	0,7435	0,7404	0,7434	0,7405	0,7429	0,7399	0,7432	0,7402
5	0,7452	0,7424	0,7452	0,7452	0,7445	0,7420	0,7448	0,7422
6	0,7467	0,7445	0,7467	0,7445	0,7460	0,7438	0,7464	0,7442
7	0,7450	**0,7477**	0,7451	**0,7478**	0,7443	**0,7471**	0,7447	**0,7475**
8	0,7484	0,7456	0,7483	0,7458	0,7478	0,7451	0,7480	0,7455
9	0,7486	0,7453	0,7486	0,7454	0,7479	0,7448	0,7482	0,7451
10	**0,7490**	0,7468	**0,7489**	0,7468	**0,7482**	0,7460	**0,7486**	0,7464
Mean	0.7440	0.7416	0.7440	0.7420	0.7434	0.7411	0.7437	0.7414

4.3.2. Klasifikasi dengan Metode *Random Forest*

Metode klasifikasi yang digunakan selanjutnya adalah metode klasifikasi *Random Forest*. Sebelum dilakukan analisis dengan menggunakan metode *random forest*, perlu dilakukan analisis CART karena metode CART sendiri merupakan metode yang mendasari dari analisis *random forest*. Pada analisis CART digunakan data sebanyak 14.095 *frame* audio dengan pembagian data *training* dan *testing* menggunakan KCV dengan nilai *K* dari 2 hingga 10.

Analisis CART dimulai dengani pembentukan pohon klasifikasi yang maksimal. Berikut merupakan penjelasan untuk masing-masing tahapan analisis klasifikasi CART dengan menggunakan kombinasi data *training* dan *testing* menggunakan KCV dengan nilai *K*=2 pada *fold* 1. Serangkaian tahapan analisis CART berikut juga dilakukan untuk pembagian data *training* dan *testing* yang lain yaitu dengan menggunakan nilai *K* dari 2 hingga 10 pada masing-masing *fold*.

Tahapan awal dalam membentuk pohon klasifikasi adalah dengan menentukan variabel pemilah dan nilai variabel (*threshold*). Variabel pemilah dan *threshold* dipilih dari beberapa kemungkinan pemilah dari masing-masing variabel. Perhitungan banyaknya kemungkinan pemilah ditampilkan pada Tabel 4.12.

Tabel 4.12 Perhitungan Kemungkinan Jumlah Pemilah dari Setiap Variabel

Variabel	Nama Variabel	Banyaknya Kategori (Nilai Amatan Sampel)	Kemungkinan Pemilah
X_1	MFCC1	7475	7475-1=7474
X_2	MFCC2	7475	7475-1=7474
X_3	MFCC3	7475	7475-1=7474
X_4	MFCC4	7475	7475-1=7474
X_5	MFCC5	7475	7475-1=7474
X_6	MFCC6	7475	7475-1=7474
X_7	MFCC7	7475	7475-1=7474

Tabel 4.12 Perhitungan Kemungkinan Jumlah Pemilah dari Setiap Variabel (*Lanjutan*)

Variabel	Nama Variabel	Banyaknya Kategori (Nilai Amatan Sampel)	Kemungkinan Pemilah
X_8	MFCC8	7475	7475-1=7474
X_9	MFCC9	7475	7475-1=7474
X_{10}	MFCC10	7475	7475-1=7474
X_{11}	MFCC11	7475	7475-1=7474
X_{12}	MFCC12	7475	7475-1=7474

Dari berbagai kemungkinan pemilah dari tiap variabel, selanjutnya dihitung indeks gini yang merupakan ukuran keheterogenan simpul. Indeks gini lebih sering digunakan karena alasan kesederhanaan dalam proses perhitungan. Cara kerja indeks gini adalah melakukan pemilihan simpul dengan berfokus pada masing-masing simpul kanan atau kiri. Hasil perhitungan indeks gini kemudian digunakan untuk menentukan *goodness of split* dari masing-masing pemilah. Pemilah yang terpilih adalah variabel pemilah dan nilai variabel (*threshold*) yang memiliki nilai *goodness of split* tertinggi. Contoh perhitungan indeks gini pada variabel MFCC dengan menggunakan salah satu *threshold* seperti pada Tabel 4.13.

Tabel 4.13 Ilustrasi Pemilahan pada Sampel MFCC1

MFCC1	***Genre* Musik**					Total
	Disco	***Hiphop***	***Jazz***	***Pop***	***Reggae***	
< 0,10542	1457	1329	316	1317	786	5205
≥ 0,01542	52	209	1160	150	699	2270

Selanjutnya melakukan perhitungan untuk nilai indeks gini pada masing-masing simpul kanan dan kiri sebagai berikut.

$$imp(t_L) = 2 \times \left(\frac{1457}{5205} \times \frac{1329}{5205} \times \frac{316}{5205} \times \frac{1317}{5205} \times \frac{786}{5205} \right) = 0,000332$$

$$imp(t_R) = 2 \times \left(\frac{52}{2270} \times \frac{209}{2270} \times \frac{1160}{2270} \times \frac{150}{2270} \times \frac{699}{2270} \right) = 0,0000438608$$

Kemudian menentukan kriteria *goodness of split* untuk evaluasi pemilahan yang telah dilakukan oleh pemilah *s* pada simpul *t*. karena hanya ada satu kemungkinan pemilahan, maka untuk simpul MFCC1 hanya ada satu kriteria *goodness of split*.

$$\phi(s,t) = 0,000639532 - \left(\frac{5205}{7475} \right) \times 0,000331594 - \left(\frac{2270}{7475} \right) \times 0,0000438608$$

$$\phi(s,t) = 0,000395317$$

Simpul *t* dikatakan sebagai simpul terminal jika tidak terdapat penuruan heterogenitas atau dengan kata lain hanya terdapat satu kelas pada simpul anak.

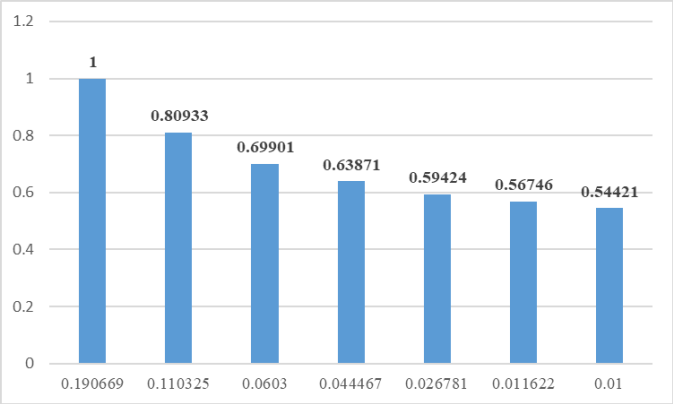
Besarnya kontribusi variabel sebagai pemilah baik pemilah utama maupun pengganti pada pohon klasifikasi maksimal yang terbentuk ditunjukkan melalui suatu angka skor yang ditampilkan pada Tabel 4.14.

Tabel 4.14 Nilai *Importance Variable*

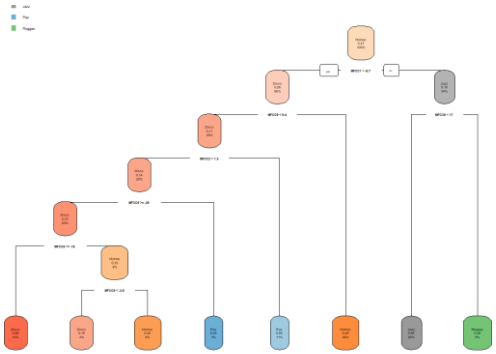
Variabel	Skor Variabel
MFCC1	100
MFCC9	56.7855
MFCC5	55.19072
MFCC2	53.56934
MFCC4	51.0028
MFCC7	44.81933
MFCC11	44.09287
MFCC8	42.75049
MFCC6	41.51806
MFCC3	40.72539
MFCC12	30.47991
MFCC10	30.16131

Tabel 4.14 menunjukkan bahwa semua variabel menjadi pembangun pohon klasifikasi. Akan tetapi, berdasarkan skor yang dihasilkan diketahui bahwa variabel MFCC1 meruapakan variabel yang terpenting dan menjadi pemilah utama dalam mengklasifikasikan musik kedalam beberapa *genre* musik. Hasil dari pohon klasifikasi maksimal yang berukuran relatif besar tidak dapat ditampilkan dalam laporan ini, mengingat pohon klasifikasi maksimal tersebut sangat besar.

Langkah selanjutnya adalah melakukan pemangkasan pada pohon. Nilai yang digunakan adalah *complexity parameter*. Pemangkasan pohon tersebut bertujuan untuk menghindari adanya *overfitting* yang diakibatkan oleh jumlah pemilahan yang terlalu banyak. Gambar 4.2 menunjukkan nilai *complexity parameter* dan besar *error*, dimana dari Gambar 4.2 menunjukkan bahwa parameter *complexity parameter* terbaik adalah sebesar 0,01 karena memiliki error terkecil. Parameter tersebut didapatkan pada pembagian *nsplit* sebesar 7. Selanjutnya, dibentuk pohon keputusan dengan menggunakan nilai *complexity parameter* optimum yaitu 0,01 dengan *nsplit* sebesar 7 seperti pada Gambar 4.3. Gambar 4.3 merupakan pohon klasifikasi optimum yang menjelaskan bahwa variabel MFCC1 menjadi variabel pertama yang dapat memprediksi sebuah audio kedalam beberapa *genre* musik.



Gambar 4.2 Nilai *Complexity Parameter* beserta Error



Gambar 4.3 CART MFCC *Fold* 1

Setelah melakukan analisis dengan menggunakan CART, dapat dilanjutkan dengan melakukan analisis dengan menggunakan *Random Forest*. Dalam penelitian ini parameter kontrol yang ditentukan adalah jumlah pohon yang dibentuk akan dicobakan pada kombinasi 50, 100, 500 dan 1000 pohon melalui proses *grid search* dimana akan didapatkan satu nilai jumlah pohon terbaik yang akan dilanjutkan dalam proses *Random Foreest* yaitu dengan jumlah pohon sebesar 1000. Sehingga, jumlah pohon yang digunakan dalam penelitian ini hanya 1000.

K-Fold Cross Validation (*KCV*) diaplikasikan untuk membagi data *training* dan data *testing* pada metode *Random Forest* dengan nilai *K* dari 2 hingga 10 dan didapatkan hasil kebaikan pada setiap nilai *K* seperti pada Tabel 4.15. Tabel 4.15 memperlihatkan bahwa dengan mengaplikasikan *KCV* pada pembagian data *training* dan *testing* dengan nilai *K* dari 2-10, menunjukkan bahwa dengan menggunakan *K* sebesar 9 memberikan performa yang baik pada data *training* maupun data *testing* dengan nilai *accuracy*, *sensitivity*, *precision* dan *Fscore* berturut-turut sebesar 0,9969, 0,9969, 0,9970 dan 0,9970 sedangkan pada data testing sebesar 0,8883; 0,8882; 0,8883 dan 0,8882. Oleh karena itu, model *Random Forest* terbaik apabila menggunakan *KCV* dengan nilai *K* sebesar 9.

Tabel 4.15 Performa Metode *Random Forest* setiap *K*

K	***Accuracy***		***Sensitivity***		***Precision***		***Fscore***	
	Training	***Testing***	***Training***	***Testing***	***Training***	***Testing***	***Training***	***Testing***
2	0,9961	0,8637	0,9961	0,8638	0,9961	0,8634	0,9961	0,8636
3	0,9965	0,8756	0,9965	0,8756	0,9965	0,8754	0,9965	0,8755
4	0,9967	0,8811	0,9967	0,8811	0,9967	0,8810	0,9967	0,8810
5	0,9969	0,8847	0,9969	0,8847	0,9969	0,8846	0,9969	0,8847
6	0,9968	0,8858	0,9968	0,8859	0,9968	0,8858	0,9968	0,8859
7	0,9968	0,8864	0,9968	0,8865	0,9968	0,8864	0,9968	0,8865
8	0,9968	0,8882	0,9968	0,8882	0,9968	0,8881	0,9968	0,8881
9	**0,9969**	**0,8883**	**0,9969**	**0,8882**	**0,9970**	**0,8883**	**0,9970**	**0,8882**
10	0,9968	0,8881	0,9968	0,8882	0,9969	0,8882	0,9968	0,8882
Mean	0.9967	0.8824	0.9967	0.8825	0.9967	0.8824	0.9967	0.8824

4.4. Perbandingan Performa Klasifikasi Antar Metode

Setelah dilakukan klasifikasi dengan menggunakan dua metode klasifikasi yaitu *Support Vector Machine* (SVM) dan *Random Forest* (RF) dan didapatkan juga nilai-nilai *accuracy*, *sensitivity*, *precision* dan *Fscore* pada masing-masing metode yang digunakan, maka akan dilakukan perbandingan kebaikan metode berdasarkan nilai-nilai tersebut untuk memilih metode mana yang terbaik yang nantinya akan digunakan pada pembuatan *Graphic User Interface* (GUI). Rangkuman hasil performa untuk setiap metode yang digunakan dapat dilihat pada Tabel 4.16.

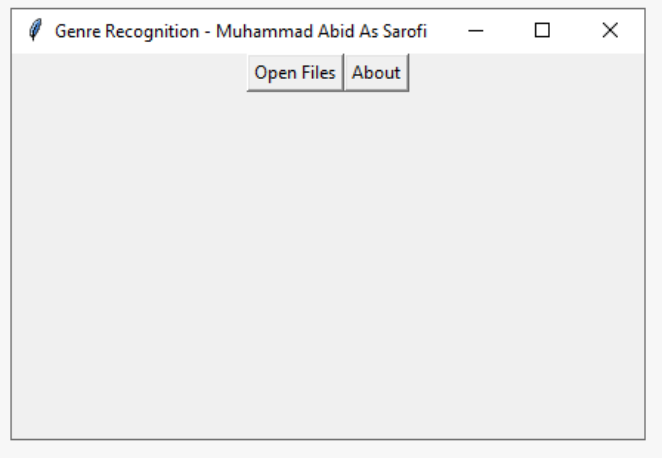
Tabel 4.16. Perbandingan Performa Metode SVM dan RF

Performansi	Data	Metode	
		SVM	RF
Accuracy	*Training*	0,7490	**0,9969**
	Testing	0,7468	**0,8883**
Sensitivity	*Training*	0,7489	**0,9969**
	Testing	0,7468	**0,8882**
Precision	*Training*	0,7482	**0,9970**
	Testing	0,7460	**0,8883**
F-score	*Training*	0,7486	**0,9970**
	Testing	0,7464	**0,8882**

Tabel 4.16 menunjukkan bahwa, apabila dilihat dari empat nilai performansi kebaikan klasifikasi (*accuracy*, *sensitivity*, *precision* dan *Fscore*) dapat disimpulkan bahwa metode *Random Forest* mampu mengklasifikasikan data tersebut lebih baik daripada dengan menggunakan metode *Support Vector Machine* (SVM) dengan nilai *accuracy*, *sensitivity*, *precision* dan *Fscore* pada data *training* berturut-turtu sebesar 0,9969; 0,9969; 0,9970; dan 0,9970 sedangkan pada data *testing* berturut-turut sebesar 0,8883; 0,8882; 0,8883; dan 0,8882.

4.5. *Graphic User Interface*

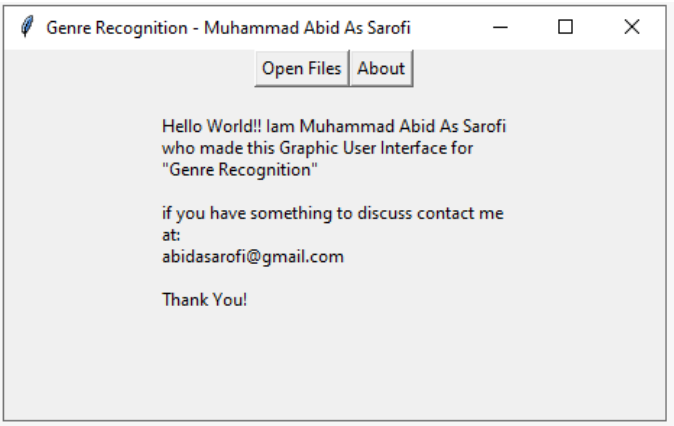
Graphic user interface (GUI) adalah sebuah sistem yang dapat membuat para pengguna atau user dapat berinteraksi dengan suatu perangkat komputer yang digunakan oleh pengguna tersebut. GUI pada penelitian ini digunakan untuk mendeteksi *genre* dari sebuah *file audio* musik. *File audio* musik yang digunakan dalam penelitian ini berformat *.wav* saja. Tampilan awal dari GUI pada penelitian ini seperti pada Gambar 4.4.



Gambar 4.4 Tampilan Awal *Graphic User Interface*

Gambar 4.4 merupakan tampilan awal dalam GUI pada penelitian ini. Terlihat pada bagian judul terdapat tulisan "*Genre Recognition* – Muhammad Abid As Sarofi", dimana maksud dari judul tersebut adalah GUI yang dibuat dikhususkan untuk mendeteksi *genre* dari sebuah musik yang dibuat oleh penulis yaitu Muhammad Abid As Sarofi. Terdapat tombol bertuliskan "*Open Files*" dengan fungsi yaitu membuat *file explorer* untuk memilih lagu yang akan dideteksi *genre*nya dan "About" yang merupakan keterangan dari GUI.

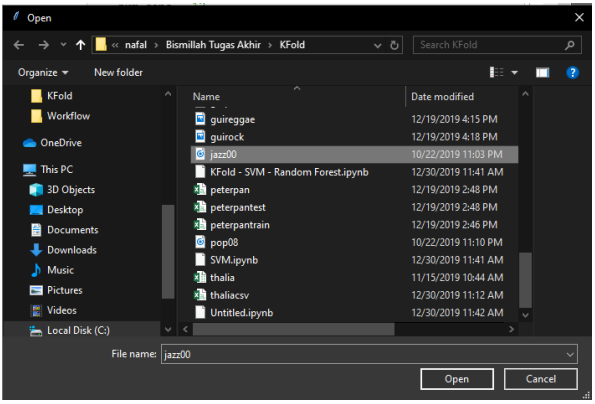
Ilustrasi penggunaan tombol "*About*" seperti pada Gambar 4.5.



Gambar 4.5 Tampilan "*About*" GUI

Gambar 4.5 merupakan hasil output apabila tombol "*About*" ditekan. Keterangan pada Gambar 4.5 menunjukkan pencipta GUI dan tujuan GUI tersebut juga kontak dari pembuat GUI.

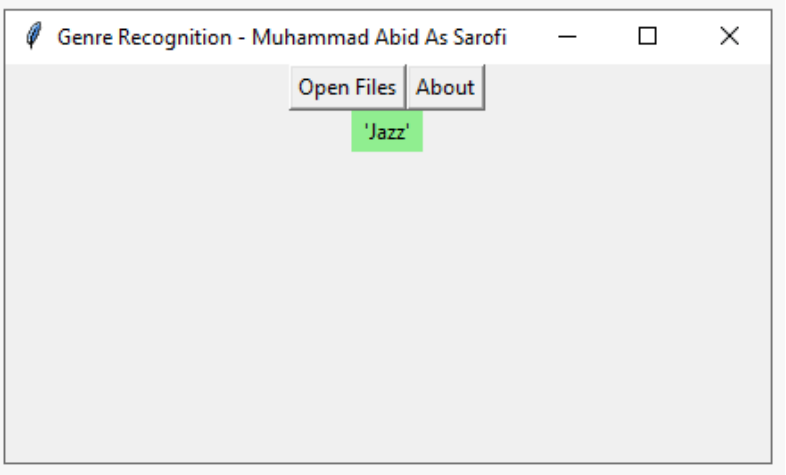
Ilustrasi penggunaan tombol "*Open Files*" seperti pada Gambar 4.6.



Gambar 4.6 Tampilan "*Open Files*" GUI

Gambar 4.6 menunjukkan tampilan pada saat "*Open Files*" ditekan yang akan membawa *user* menuju *file explorer* untuk meng*import* lagu yang akan dideteksi *genre*nya. Contoh yang digunakan adalah dataset dari GTZAN dengan judul "jazz00" yang

akan dideteksi *genre* musiknya. Setelah diklik tombol "*Open*" di *file explorer* seperti pada Gambar 4.6 maka akan muncul hasil seperti Gambar 4.7.



Gambar 4.7 Tampilan Hasil Deteksi *Genre* Musik GUI

Gambar 4.7 merupakan tampilan hasil deteksi *genre* dari musik yang ingin diketahui genrenya. Terdapat tulisan 'Jazz' pada GUI tersebut yang menunjukan bahwa *genre* musik dari *file audio* yang digunakan adalah *genre* musik Jazz.

BAB V
KESIMPULAN DAN SARAN

5.1. Kesimpulan

Kesimpulan yang diperoleh dari penelitian ini adalah sebagai berikut.

1. Metode SVM terbaik yang mampu mengklasifikasikan data audio dengan baik dengan kernel RBF dan K=7, sedangkan metode RF pada nilai K=9.
2. Metode yang memiliki kinerja terbaik adalah metode *Random Forest* dengan jumlah pohon sebesar 1000 dengan menggunakan nilai *K* sebesar 9 pada *K-Fold Cross Validation* (*KCV*) dengan nilai performansi klasifikasi pada data *training* maupun data *testing* dengan nilai *accuracy*, *sensitivity*, *precision* dan *Fscore* berturut-turut sebesar 0,9969, 0,9969, 0,9970 dan 0,9970 sedangkan pada data testing sebesar 0,8883; 0,8882; 0,8883 dan 0,8882.
3. GUI telah dibuat dengan input adalah data audio musik dan outputnya adalah genre musik yang berhasil diklasifikasikan.

5.2. Saran

Berdasarkan kesimpulan yang diperoleh, dapat dirumuskan saran sebagai pertimbangan penelitian selanjutnya adalah sebagai berikut.

1. Menggunakan fitur ekstraksi yang lain seperti berdasarkan *time domain* yaitu *energy*, *zero-cros-sing rate* (ZCR) dan *entropy of energy* ataupun *frequency domain* yaitu *spectral centroid and spead*, *spectral entropy*, *spectral flux*, *spectral rolloff* dan *chroma vector* agar mendapatkan hasil yang maksimal.
2. Menggunakan metode klasifikasi yang lain agar didapatkan hasil yang lebih baik daripada menggunakan metode yang digunakan dalam penelitian ini, terutama dalam kasus *overfitting* pada metode *random forest*.
3. Menambahkan jumlah *file* audio musik yang digunakan agar hasilnya lebih baik.

(Halaman ini sengaja dikosongkan)

DAFTAR PUSTAKA

Al, Y. (2019). Jenis-Jenis Musik. (https://www.eduspensa.id/jenis-jenis-musik/, diakses pada tanggal 20 Oktober 2019).

APJII. (2016). Infografis: Penetrasi dan Perilaku Pengguna Internet Indonesia, 1-34.

Banoe, P. (2003). *Kamus Musik.* Yogyakarta: Kanisius.

Breiman, L. (2001). Random Forests. Machine Learning, 45, 5-32.

Breiman, L., Friedman, J.H., Olshen, R.A. & Stone C.J. (1993). *Classification and Regression Tress*. New York: Chapman Hall.

Cen, L., Wu, F., Yu, Z.L., & Hu, F. (2016). *A Real-Time Speech Emotion Recognition System and its Application in Online Learning*. Emotions, Technology, Design, and Learning, 27-46.

Chatman, L. (2018). *Reggae Rising: Hip-Hop's Roots in Reggae Music*. (https://www.nwfolklife.org/reggae-rising-hip-hops-roots-in-reggae-music, diakses pada tanggal 2 November 2019).

Dailysocial.id. (2018). *YouTube adalah Platform Favorit Responden Indonesia Nikmati Musik Streaming*. (https://dailysocial.id/post/survei-dailysocial-youtube-adalah-platform-favorit-responden-indonesia-nikmati-musik-streaming, diakses pada tanggal 2 September 2019).

Dhanalaksmi, P., Palanivel, S., & Ramalingam, V. (2009). Classification of Audio Signal Using SVM and RBFNN. *Expert Systems with Applications*.

Fansuri, M.R. (2011). *Klasifikasi Genre Musik Menggunakan Learning Vector Quantization (LVQ)*. Departemen Ilmu Komputer, Institut Pertanian Bogor.

Fayek, H. (2016). *Speech Processing for Machine Learning: Filter banks, Mel-Frequency Cepstral Coefficients (MFCCs) and What's In-Between*. (https://haythamfayek.com/2016/04/21/speech-processing-for-machine-learning.html , diakses pada tanggal 25 September 2019).

Geneur, R., Poggi, J., M. & Malot, C., T. (2009). Variable Selection using Random Forests. France: Laboratoire de Mathematiques, Universite Paris-Sud 11, Bat 425, 91405 Orsay.

Giannakopoulos, T., & Pikrakis, A. (2014). Audio Features. *Introduction to Audio Analysis*, 59–103.

Gokgoz, E., & Subasi, A. (2015). Comparison of Decision Tree Algirthms for EMG Signal Classification Using DWT. *Biomedical Signal Processing and Control*, 18, 138-144.

Han, J., Kamber, M., & Pei, J. (2012). *Data Mining Concepts and Techniques (3rd ed.)*. USA: Morgan Kaufmann.

Hardjana, S. (2003). *Corat-Coret Musik Kontemporer Dulu dan Kini*. Jakarta: Ford Foundation dan Masyarakat Seni Pertunjukan Indonesia.

Harenda, F.R. (2018). *Perancangan Gedung Concert Hall di Yogyakarta dengan Pendekatan Arsitektur Neo-Vernakular*. Universitas Gadjah Mada.

Hsu, C.W., Chang, C.C., & Lin, C.J. (2003). A Practicl Guide to Support Vector Classification, 1-16

Huang, C.M., Lee, Y.J., Lin, D.K., & Huang, S.Y. (2007). Model Selenction for Support Vector Machines Via Uniform Design. *Computational Statistics & Data Analysis*, 52(1), 335-346

Jamalus. (1988). *Panduan Pengajaran Buku Pengajar Musik Melalui Pengalaman Musik.* Proyek Pengembangan Lembaga Pendidikan. Jakarta.

Joox. (2015). Joox FAQ. (https://www.joox.com/en/faq.html, diakses pada tanggal 25 September 2019).

Kamus Besar Bahasa Indonesia. (https://kbbi.kemdikbud.go.id/entri/genre, diakses pada tanggal 25 September 2019).

Kaonang, G. (2016). Spotify *Catatkan Pertumbuhan Positif di Indonesia dan Hadirkan Fitur Discovery Weekly*. (https://dailysocial.id/post/spotify-catatkan-pertumbuhan-positif-di-indonesia-dan-hadirkan-fitur-discover-weekly, diakses pada tanggal 1 September 2019).

Kulyukin, V., Mukherjee, S., & Amlathe, P. (2018). Toward Audio Beehive Monitoring: Deep Learning vs. Standard Machine Learning in Classifying Beehive Audio Samples. *Applied Sciences*.

Nanni, L., Costa, Y.M.G., Lumini, A., Kim, M.Y., & Baek, S.R. (2016) Combining Visual and Acoustic Features for Music Genre Classification. *Expert System with Applications*.

Pakpahan, R. (2017). *"Analisa Fenomena Hoax diberbagai Media Sosial dan Cara Menanggulangi Hoax"* dalam: Konferensi Nasional Ilmu Sosial dan Teknologi (KNIST), pp 479-484.

Panchwagh, M.M., & Katkar, V.D. (2016) Music Genre Classification Using Data Mining Algorithm. *Conference on Advances in Signal Processing*.

Putra, D., & Resmawan, A. (2011). Verifikasi Biometrika Suara Menggunakan Metode MFCC dan DTW. *Lontar Komputer*.

Sartono, B. & Syafitri, U.D. (2010). *Metode Pohon Gabungan: Solusi Pilihan untuk Mengatasi Kelemahan Pohon Regresi dan Klasifikasi Tunggal, Forum Statistika dan Komputasi, Vol. 15, No.1, ISSN: 0853-8115*.

Shuiping, W., Zhenming, T., & Shiqiang, L. (2011). Design and Implementation of an Audio Classification System Based on

SVM. *Advanced in Control Engineering and Information Science.*

Sokolova, M., & Lapalme, G. (2009). A systematic analysis of performance measures for classification tasks. *Information Processing and Management*, 45(2), 427-437.

Souli, S., & Lachiri, Z. (2018). Audio Sounds Classification Using Scattering Features and Support Vector Machines for Medical Surveillance. *Applied Acoustics*.

Spotify. (2018). *Spotify Support.* (https://support.spotify.com, diakses pada tanggal 1 September 2019).

Sungkono, J. (2013). *Resampling Bootstrap Pada R, Magistra No. 84, Th. XXV, ISSN: 0215-9511*.

Thambi, S., Sreekumar, K.T., Santhosh, K.C., & Reghu, R.P.C. (2014). Random Forest Algorithm for Improving the Performance of Speech/Non-speech Detection. *First International Conference on Computational Systems and Communications.*

Tzanetakis, G., & Cook, P. (2002). Musical Genre Classification of Audio Signals. *IEEE Transactions on Speech and Audio Processing.*

Zheng, H., Chen, L., Han, X., Zhao, X. & Ma, Y. (2009). *Classification and Regression Tree (CART) for Analysis of Soybean Yield Variability Among Fields in Norheast China: The Importance of Phosporus Application Rates under Drought Conditions, Agriculture, Ecosystems and Environment 132, 98-105*.

LAMPIRAN

Lampiran 1. Syntax *Pre-processing*

```
SYNTAX PRE-PROCESSING
import numpy
import scipy.io.wavfile
from scipy.fftpack import dct
#import data
sample_rate, signal =
scipy.io.wavfile.read('disco09.wav')
signal = signal[0:int(3 * sample_rate)]
#pre-emphasis
pre_emphasis = 0.97
emphasized_signal = numpy.append(signal[0],
signal[1:] - pre_emphasis * signal[:-1])
#framing
frame_size = 0.025
frame_stride = 0.01
frame_length, frame_step = frame_size * sample_rate,
frame_stride * sample_rate
signal_length = len(emphasized_signal)
frame_length = int(round(frame_length))
frame_step = int(round(frame_step))
num_frames =
int(numpy.ceil(float(numpy.abs(signal_length -
frame_length)) / frame_step))
pad_signal_length = num_frames * frame_step +
frame_length
z = numpy.zeros((pad_signal_length - signal_length))
pad_signal = numpy.append(emphasized_signal, z)
indices = numpy.tile(numpy.arange(0, frame_length),
(num_frames, 1)) + numpy.tile(numpy.arange(0,
num_frames * frame_step, frame_step), (frame_length,
1)).T
frames = pad_signal[indices.astype(numpy.int32,
copy=False)]
#windowing
frames *= numpy.hamming(frame_length)
#FFT and Power Spectrum
NFFT = 512
mag_frames = numpy.absolute(numpy.fft.rfft(frames,
NFFT))  # Magnitude of the FFT
pow_frames = ((1.0 / NFFT) * ((mag_frames) ** 2))
#Filter Banks
nfilt = 40
low_freq_mel = 0
high_freq_mel = (2595 * numpy.log10(1 + (sample_rate
/ 2) / 700))  # Convert Hz to Mel
```

Lampiran 1. Syntax *Pre-processing (Lanjutan)*

```
mel_points = numpy.linspace(low_freq_mel,
high_freq_mel, nfilt + 2)  # Equally spaced in Mel
scale
hz_points = (700 * (10**(mel_points / 2595) - 1))  #
Convert Mel to Hz
bin = numpy.floor((NFFT + 1) * hz_points /
sample_rate)

fbank = numpy.zeros((nfilt, int(numpy.floor(NFFT / 2
+ 1))))
for m in range(1, nfilt + 1):
    f_m_minus = int(bin[m - 1])   # left
    f_m = int(bin[m])             # center
    f_m_plus = int(bin[m + 1])    # right

    for k in range(f_m_minus, f_m):
        fbank[m - 1, k] = (k - bin[m - 1]) / (bin[m]
- bin[m - 1])
    for k in range(f_m, f_m_plus):
        fbank[m - 1, k] = (bin[m + 1] - k) / (bin[m +
1] - bin[m])
filter_banks = numpy.dot(pow_frames, fbank.T)
filter_banks = numpy.where(filter_banks == 0,
numpy.finfo(float).eps, filter_banks)  # Numerical
Stability
filter_banks = 20 * numpy.log10(filter_banks)  # dB
#MFCC
num_ceps = 12
mfcc = dct(filter_banks, type=2, axis=1,
norm='ortho')[:, 1 : (num_ceps + 1)] # Keep 2-13
```

Lampiran 2. Syntax *Grid Search* SVM Kernel *Linear*

```
# defining parameter range
param_grid = {'C':[10**-2,10**-1,1],
                    'kernel': ['linear']}
grid = GridSearchCV(SVC(), param_grid, refit = True,
verbose = 1)
# fitting the model for grid search
grid.fit(X, y)
# defining parameter range
param_grid = {'C':[1,2,10,100],
                    'kernel': ['linear']}
grid = GridSearchCV(SVC(), param_grid, refit = True,
verbose = 1)
# fitting the model for grid search
grid.fit(X, y)
```

Lampiran 3. Hasil *Grid Search* SVM Kernel *Linear*

```
{'C': 1, 'kernel': 'linear'}
SVC(C=1, cache_size=200, class_weight=None, coef0=0.0,
  decision_function_shape='ovr', degree=3,
gamma='auto', kernel='linear',
  max_iter=-1, probability=False, random_state=None,
shrinking=True,
  tol=0.001, verbose=False)
{'C': 100, 'kernel': 'linear'}
SVC(C=100, cache_size=200, class_weight=None,
coef0=0.0,
  decision_function_shape='ovr', degree=3,
gamma='auto', kernel='linear',
  max_iter=-1, probability=False,
random_state=None, shrinking=True,
  tol=0.001, verbose=False)
```

Lampiran 4. Syntax *Grid Search* SVM Kernel RBF

```
# defining parameter range
param_grid2 = {'C':[10**-2,10**-1,10**0,10,10**2],
                    'gamma': [2**-15,2**-14,2**-
13,2**-12,2**-11,2**-10,2**-9,2**-8,2**-7,2**-6,2**-
5,2**-4,2**-3,2**-2,2**-1,2**0,2,2**2,2**3],
                    'kernel': ['rbf']}

grid2 = GridSearchCV(SVC(), param_grid2, refit =
True, verbose = 1)
# fitting the model for grid search
grid2.fit(X_train, y_train)
```

Lampiran 5. Hasil *Grid Search* SVM Kernel RBF

```
#HASIL
{'C': 0.1, 'gamma': 3.0517578125e-05, 'kernel': 'rbf'}
SVC(C=0.1, cache_size=200, class_weight=None,
coef0=0.0,
    decision_function_shape='ovr', degree=3,
gamma=3.0517578125e-05,
    kernel='rbf', max_iter=-1, probability=False,
random_state=None,
    shrinking=True, tol=0.001, verbose=False)
```

Lampiran 6. Syntax Klasifikasi SVM Kernel RBF

```
import numpy as np
import pandas as pd
from sklearn.utils import shuffle
from sklearn.model_selection import train_test_split
from sklearn.metrics import
classification_report,confusion_matrix
from sklearn.model_selection import KFold
from sklearn.metrics import confusion_matrix,
accuracy_score, precision_score, recall_score
from sklearn import svm
from sklearn.svm import SVC
svc = svm.SVC(kernel='rbf',C=0.1,gamma=2**-15)
svm=svc.fit(X,y)
y_pred=svm.predict(X)
kf = KFold(n_splits=10,random_state=0, shuffle=True)
kf.get_n_splits(X)
print(kf)
for train_index, test_index in kf.split(X):
    print("TRAIN:", train_index, "TEST:", test_index)
    X_train, X_test = X.iloc[train_index],
X.iloc[test_index]
    y_train, y_test = y.iloc[train_index],
y.iloc[test_index]
    from sklearn import svm
    from sklearn.svm import SVC
    svc = svm.SVC(kernel='rbf',C=10**-1,gamma=2**-15)
    svm=sv
    svm=svc.fit(X_train,y_train)
    y_pred=svm.predict(X_test)
    y_predt=svm.predict(X_train)
    conf_matrixknn=confusion_matrix(y_test, y_pred)
    print(conf_matrixknn)
    akurasitest=accuracy_score(y_test,y_pred)
    presisitest=precision_score(y_test,
y_pred,average='micro')
    recallstest=recall_score(y_test,
y_pred,average='micro')
    akurasitrain=accuracy_score(y_train,y_predt)
    presisitrain=precision_score(y_train,
y_predt,average='micro')
    recallstrain=recall_score(y_train,
y_predt,average='micro')
    print("Akurasi Test:",akurasitest)
    print("Presisi Test:",presisitest)
    print("Sensitivity Test:",recallstest)
    print("Akurasi Train:",akurasitrain)
    print("Presisi Train:",presisitrain)
    print("Sensitivity Train:",recallstrain)
```

Lampiran 7. Syntax Klasifikasi SVM Kernel Linear

```
import numpy as np
import pandas as pd
from sklearn.utils import shuffle
from sklearn.model_selection import train_test_split
from sklearn.metrics import
classification_report,confusion_matrix
from sklearn.model_selection import KFold
from sklearn.metrics import confusion_matrix,
accuracy_score, precision_score, recall_score
from sklearn import svm
from sklearn.svm import SVC
svc = svm.SVC(kernel='linear',C=10)
svm=svc.fit(X,y)
y_pred=svm.predict(X)
kf = KFold(n_splits=10,random_state=0, shuffle=True)
kf.get_n_splits(X)
print(kf)
for train_index, test_index in kf.split(X):
    print("TRAIN:", train_index, "TEST:", test_index)
    X_train, X_test = X.iloc[train_index],
X.iloc[test_index]
    y_train, y_test = y.iloc[train_index],
y.iloc[test_index]
    from sklearn import svm
    from sklearn.svm import SVC
    svc = svm.SVC(kernel='linear',C=100)
    svm=sv
    svm=svc.fit(X_train,y_train)
    y_pred=svm.predict(X_test)
    y_predt=svm.predict(X_train)
    conf_matrixknn=confusion_matrix(y_test, y_pred)
    print(conf_matrixknn)
    akurasitest=accuracy_score(y_test,y_pred)
    presisitest=precision_score(y_test,
y_pred,average='micro')
    recallstest=recall_score(y_test,
y_pred,average='micro')
    akurasitrain=accuracy_score(y_train,y_predt)
    presisitrain=precision_score(y_train,
y_predt,average='micro')
    recallstrain=recall_score(y_train,
y_predt,average='micro')
    print("Akurasi Test:",akurasitest)
    print("Presisi Test:",presisitest)
    print("Sensitivity Test:",recallstest)
    print("Akurasi Train:",akurasitrain)
    print("Presisi Train:",presisitrain)
    print("Sensitivity Train:",recallstrain)
```

Lampiran 8. Syntax *Grid Search* RF

```
from sklearn.ensemble import RandomForestClassifier
rf=RandomForestClassifier()
rf.fit(X_train,y_train)
rf.score(X_test,y_test)
%pylab inline
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import GridSearchCV,
train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
def GridSearch(x, y, model, parameters):
    clf = GridSearchCV(model, parameters,
scoring='accuracy', n_jobs=-1, cv=5, verbose=1)
    clf.fit(x, y)
    print("Best Score: "+str(clf.best_score_))
    print("Best Params: "+str(clf.best_params_))
    return (clf)
ListParams = {
    'n_estimators': [50, 100, 500, 1000],
    'max_depth':[1, 5, 10, 15, 20, 25, 30],
    'min_samples_leaf' : [1, 2, 4, 6, 8, 10],
    'max_features': [0.1, 'sqrt', 'log2', None]
}

BestRF = GridSearch(X_train, y_train,
RandomForestClassifier(random_state=123), ListParams
```

Lampiran 9. Hasil *Grid Search* RF

```
Fitting 5 folds for each of 672 candidates, totalling
3360 fits
[Parallel(n_jobs=-1)]: Using backend LokyBackend with
4 concurrent workers.
[Parallel(n_jobs=-1)]: Done  42 tasks      | elapsed:
41.7s
[Parallel(n_jobs=-1)]: Done 192 tasks      | elapsed:
3.5min
[Parallel(n_jobs=-1)]: Done 442 tasks      | elapsed:
11.4min
[Parallel(n_jobs=-1)]: Done 792 tasks      | elapsed:
24.0min
[Parallel(n_jobs=-1)]: Done 1242 tasks      | elapsed:
54.0min
```

Lampiran 9. Hasil *Grid Search* RF (*Lanjutan*)

```
[Parallel(n_jobs=-1)]: Done 1792 tasks      | elapsed:
102.4min
[Parallel(n_jobs=-1)]: Done 2442 tasks      | elapsed:
181.9min
[Parallel(n_jobs=-1)]: Done 3192 tasks      | elapsed:
250.0min
[Parallel(n_jobs=-1)]: Done 3360 out of 3360 |
elapsed: 281.6min finished
Best Score: 0.6339799331103679
Best Params: {'max_depth': 30, 'max_features': 'sqrt',
'min_samples_leaf': 1, 'n_estimators': 1000}
```

Lampiran 10. Syntax Klasifikasi RF

```
kf = KFold(n_splits=10,random_state=0, shuffle=True)
kf.get_n_splits(X)
print(kf)
for train_index, test_index in kf.split(X):
    print("TRAIN:", train_index, "TEST:", test_index)
    X_train, X_test = X.iloc[train_index],
X.iloc[test_index]
    y_train, y_test = y.iloc[train_index],
y.iloc[test_index]
    RF =
RandomForestClassifier(random_state=0,max_depth=30,ma
x_features='sqrt',min_samples_leaf=1,n_estimators=100
0)
    RF.fit(X_train, y_train)
    y_pred=RF.predict(X_test)
    y_predt=RF.predict(X_train)
    conf_matrixknn=confusion_matrix(y_test, y_pred)
    print(conf_matrixknn)
    akurasitest=accuracy_score(y_test,y_pred)
    presisitest=precision_score(y_test,
y_pred,average='micro')
    recallstest=recall_score(y_test,
y_pred,average='micro')
    akurasitrain=accuracy_score(y_train,y_predt)
    presisitrain=precision_score(y_train,
y_predt,average='micro')
    recallstrain=recall_score(y_train,
y_predt,average='micro')
    print("Akurasi Test:",akurasitest)
    print("Presisi Test:",presisitest)
    print("Sensitivity Test:",recallstest)
    print("Akurasi Train:",akurasitrain)
    print("Presisi Train:",presisitrain)
    print("Sensitivity Train:",recallstrain)
```

Lampiran 11. Data Hasil *Pre-processing*

Nomor	Lagu	MFCC1	MFCC2	…	MFCC12	Kelas
1	1	11,5190672	-76,625046	…	-15,620164	Disco
2	1	17,3081414	-85,634124	…	-20,201353	Disco
3	1	17,6441334	-87,065976	…	-12,530621	Disco
4	1	20,9220411	-80,435329	…	-14,534859	Disco
5	1	28,0236141	-75,045555	…	-11,203806	Disco
6	1	-37,21895	-35,33105	…	-9,0193215	Disco
7	1	-56,367698	-28,978139	…	-5,1516343	Disco
8	1	-44,730966	-7,422376	…	-8,3283323	Disco
…	…	…	…	…	…	…
14943	50	10,2769374	14,6150905	…	-9,7516181	Reggae
14944	50	53,1473185	15,9678331	…	-5,84705	Reggae
14945	50	20,4889262	16,2990488	…	-15,918448	Reggae
14946	50	-40,86834	-10,797586	…	-12,463389	Reggae
14947	50	-28,493735	-15,858251	…	-3,035142	Reggae
14948	50	-15,967948	-4,1391284	…	-17,594136	Reggae
14949	50	-18,813528	-13,123803	…	-26,619518	Reggae
14950	50	-6,8256469	-17,501574	…	-22,575368	Reggae

Lampiran 12. Parameter α_m y_m SVM

Nomor	1	2	3	4	5	…	12814	12815
1	0	0	0	0	0	…	0	0
2	0,1	0,1	0,1	0,1	0,1	…	-0,1	-0,1
3	0,1	0,1	0,1	0,1	0,1	…	0	0
4	0	0,1	0,1	0	0	…	0	0

Lampiran 13. Parameter $K\left(\mathbf{x},\mathbf{x}_i\right)$ SVM

Nomor	**1**	**2**	**…**	**11**	**12**
1	11,519067	-76,625046	…	-8,5004792	-15,620164
2	17,308141	-85,634124	…	-7,1223411	-20,201353
3	17,644133	-87,065976	…	-12,906453	-12,530621
4	20,922041	-80,435329	…	-6,0962009	-14,534859
5	28,023614	-75,045555	…	-7,6477249	-11,203806
6	-37,21895	-35,33105	…	-11,496749	-9,0193215
…	…	…	…	…	…
12811	20,48893	16,29905	…	6,5594571	-15,918448
12812	-40,8683	-10,7976	…	10,444489	-12,463389
12813	-28,4937	-15,8583	…	17,914785	-3,035142
12814	-15,9679	-4,13913	…	21,084761	-17,594136
12815	-18,8135	-13,1238	…	15,102895	-26,619518

Lampiran 14. Parameter b_j SVM

Nomor	***b***
1	-1,250197963
2	-0,478185837
3	-2,334074985
4	-0,834021641
5	0,765035668
6	0,36185484
7	0,285940267
8	-0,878858198
9	-0,523462239
10	-0,069512364

Lampiran 15. Surat Keterangan Pengambilan Data

SURAT PERNYATAAN

Saya yang bertanda tangan di bawah ini, mahasiswa Departemen Statistika FMKSD ITS:

Nama : Muhammad Abid As Sarofi

NRP : 06211640000082

menyatakan bahwa data yang digunakan dalam Tugas Akhir/ ~~Thesis~~ ini merupakan data sekunder yang diambil dari ~~penelitian~~ / buku/ ~~Tugas Akhir~~/ ~~Thesis~~/ publikasi lainnya yaitu:

Sumber : *Music Analysis Retrieval and Synthesis for Audio Signals* (http://marsyas.info/)

Keterangan : GTZAN *dataset* berupa *file audio* musik

Surat Pernyataan ini dibuat dengan sebenarnya. Apabila terdapat pemalsuan data maka saya siap menerima sanksi sesuai aturan yang berlaku.

Mengetahui
Pembimbing Tugas Akhir

Irhamah, S.Si., M.Si., P.hD.
NIP. 19780406 200112 2 002

Surabaya, 6 Januari 2020

Muhammad Abid As Sarofi
NRP. 06211640000082

*(coret yang tidak perlu)

BIODATA PENULIS

Penulis dengan nama lengkap Muhammad Abid As Sarofi dilahirkan di Gresik pada tanggal 20 Juni 1998. Penulis menempuh pendidikan formal di SDN 1 Pangkah Wetan, SMPN 1 Ujungpangkah, dan SMAN 1 Gresik. Penulis diterima sebagai Mahasiswa Departemen Statistika ITS melalui jalur SBMPTN pada tahun 2016. Pada tahun kedua perkuliahan, penulis mulai mengikuti organisasi dan kepanitiaan di dalam ITS sebagai staff Tim Penelitian dan Pengembangan (LITBANG) HIMASTA-ITS dan staff HRD SCC HIMASTA-ITS. Di tahun ketiga, penulis diamanahi menjadi Ketua Tim LITBANG, Manager HRD SCC HIMASTA-ITS, staff Departemen Eksternal BEM FMKSD ITS hingga Ketua Divisi Komunikasi dan Informasi Departemen Eksternal BEM FMKSD ITS periode selanjutnya. Penulis juga menjadi *student presenter* di 15th IMT-GT *International Conference on Mathematics Statistics, and their Applications* (ICMSA) 2019 di IPB juga aktif dalam mengikuti perlombaan seperti KSN, Jambore Statistika, Dokter Data dan ISCO. Penulis juga telah mengikuti beberapa kegiatan seperti pembicara dalam beberapa kegiatan di ITS seperti OKKBK, Workshop Prestasi, dan SOSDEV, *Intern* di PT. Telkom Surabaya, *Data Analyst* di Poltracking Indonesia, *Data Surveyor* hingga Asisten Dosen di Departemen Statistika maupun Departemen Sains Aktuaria. Segala kritik dan saran serta diskusi lebih lanjut mengenai Tugas Akhir ini, dapat menghubungi penulis melalui email abidasarofi@gmail.com.