TUGAS AKHIR – KI141502

RANCANG BANGUN APLIKASI PENDETEKSI SUARA TANGISAN BAYI

WELLY SETIAWAN LIMANTORO
NRP 5112 100 017

Dosen Pembimbing I
Dr.Eng. Chastine Fatichah, S.Kom, M.Kom.

Dosen Pembimbing II
Umi Laili Yuhana, S.Kom, M.Sc.

JURUSAN TEKNIK INFORMATIKA
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya 2016

[Halaman ini sengaja dikosongkan]

UNDERGRADUATE THESES – KI141502

APPLICATION DEVELOPMENT FOR DETECTING THE SOUND OF CRYING INFANTS

WELLY SETIAWAN LIMANTORO
NRP 5112 100 017

First Supervisor
Dr.Eng. Chastine Fatichah, S.Kom., M.Kom.

Second Supervisor
Umi Laili Yuhana, S.Kom., M.Sc.

DEPARTMENT OF INFORMATICS
Faculty of Information Technology
Institut Teknologi Sepuluh Nopember
Surabaya 2016

[Halaman ini sengaja dikosongkan]

LEMBAR PENGESAHAN

RANCANG BANGUN APLIKASI PENDETEKSI SUARA TANGISAN BAYI

TUGAS AKHIR

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

Oleh:
WELLY SETIAWAN LIMANTORO
NRP. 5112 100 017

Disetujui oleh Pembimbing Tugas Akhir:

1. Dr.Eng. Chastine Fatichah, S.Kom., M.Kom.
   NIP. 197512202001122001
   .....
   (Pembimbing I)

2. Umi Laili Yuhana, S.Kom., M.Sc.
   NIP. 197906262005012002
   .....
   (Pembimbing II)

SURABAYA
JUNI, 2016

[Halaman ini sengaja dikosongkan]

RANCANG BANGUN APLIKASI PENDETEKSI SUARA TANGISAN BAYI

Nama Mahasiswa	**:**	**WELLY SETIAWAN LIMANTORO**
NRP	**:**	**5112100017**
Jurusan	**:**	**Teknik Informatika FTIF-ITS**
Dosen Pembimbing 1	**:**	**Dr.Eng. Chastine Fatichah, S.Kom., M.Kom.**
Dosen Pembimbing 2	**:**	**Umi Laili Yuhana, S.Kom., M.Sc.**

Abstrak

Suara tangisan bayi merupakan sebuah tanda dari bayi yang mengalami suatu masalah. Suara tangisan tersebut dapat digunakan untuk mengidentifikasi masalah pada bayi, seperti kelaparan, kesakitan, rasa kantuk, rasa tidak nyaman, kedingingan atau kepanasan, dan lain-lain. Namun, tidak semua orang dapat mengenali arti tangis bayi tersebut.

Beberapa penelitian tentang deteksi suara tangis bayi sudah dilakukan oleh beberapa peneliti, namun saat ini masih belum ada penelitian yang membuat sebuah aplikasi pendeteksi suara tangis bayi berbasis web. Pada Tugas Akhir ini, sebuah aplikasi dibuat untuk membantu pengguna mengenali suara tangis bayi berbasis Dunstan Baby Language. Aplikasi dikembangkan dengan bahasa pemrograman R versi 3.3.1 dan package tuneR untuk ekstraksi fitur Mel-Frequency Cepstrum Coefficient (MFCC). Metode yang diterapkan pada aplikasi ini adalah ekstraksi fitur suara tangis bayi dengan algoritma MFCC, normalisasi hasil ekstraksi fitur, dan klasifikasi K-nearest Neighbor.

Dari berbagai pengujian yang dilakukan, dapat disimpulkan bahwa akurasi rata-rata terbaik sebesar 75,95% dapat dicapai ketika menggunakan parameter wintime pada ekstraksi fitur MFCC sebesar 0,08 detik, proporsi data latih 85% dan data uji 15% dari setiap kelas, normalisasi ekstraksi fitur dengan Standard Deviation Normalization, dan klasifikasi

K-nearest Neighbor dengan k=1. Hasil ini lebih baik jika dibandingkan dengan metode klasifikasi lain seperti Naive Bayes, Neural Network, maupun Support Vector Machine. Penggunaan proporsi data latih berdasarkan persentase data masing-masing jenis tangis bayi menghasilkan akurasi klasifikasi yang lebih baik jika dibandingkan dengan penggunaan jumlah data latih seimbang dari masing-masing jenis tangis bayi.

Pada pengujian aplikasi dengan seluruh data, akurasi rata-rata yang sebesar 96,57% dapat dicapai ketika menggunakan parameter wintime pada ekstraksi fitur MFCC sebesar 0,08 detik, proporsi data latih 85% setiap kelas, normalisasi ekstraksi fitur dengan Standard Deviation Normalization, dan klasifikasi K-nearest Neighbor dengan k=1. Dari pengujian tersebut, dapat disimpulkan bahwa aplikasi telah berjalan dengan baik saat mengklasifikasi seluruh data tangis bayi.

Kata kunci: Mel-Frequency Cepstrum Coefficient, K-nearest Neighbor, Suara Tangis Bayi, Dunstan Baby Language

APPLICATION DEVELOPMENT FOR DETECTING THE SOUND OF CRYING INFANTS

Student's Name	**:**	**WELLY SETIAWAN LIMANTORO**
Student's ID	**:**	**5112100017**
Department	**:**	**Department of Informatics FTIF-ITS**
First Advisor	**:**	**Dr.Eng. Chastine Fatichah, S.Kom., M.Kom.**
Second Advisor	**:**	**Umi Laili Yuhana, S.Kom., M.Sc.**

Abstract

An infant crying sound is a sign of an infant who has experienced a problem. The sound of crying can be used to identify problems of infant, such as hunger, pain, drowsiness, discomfort, cold or heat, and the others. However, not everyone can recognize the meaning infant's cry.

Several studies of the sound of crying infant detection had been done by some researchers, but currently there is still no research that develop an infant cry detection applications based on web. In this final project, an application is developed to help users identify the sound of crying infant based on Dunstan Baby Language. Application is developed with R programming language version 3.3.1 and tuneR package for Mel-Frequency Cepstrum Coefficient (MFCC) feature extraction. The method applied in this application are voice feature extraction algorithm of crying infant with MFCC, normalization of feature extraction result, and K-nearest neighbor classification.

From the various tests performed, it can be concluded that highest average accuracy 75,95% can be obtained by using parameters consist of wintime=0,08 seconds in MFCC feature extraction, 85% the proportion of training data and test data 15% of any class, feature extraction normalization by Standard Deviation Normalization, and k=1 K-nearest Neighbor

classification. This result is better than the other classification methods such as Naive Bayes, Neural Networks, and Support Vector Machine. The use of training data proportion from each type crying infants data based on the percentage resulted in better classification accuracy when compared with the use of training data from each type crying infants data based on balanced amount.

In application testing test by using all data, average accuracy 96,57% can obtained by using parameters consist of wintime=0,08 seconds in MFCC feature extraction, 85% of training data proportion of any class, feature extraction normalization by Standard Deviation Normalization, and k=1 K-nearest Neighbor classification. From the test, it can be concluded that the application has been running well when classifying all infant cry data.

Keywords: Mel-Frequency Cepstrum Coefficient, K-nearest Neighbor, Infant Crying Sound, Dunstan Baby Language

DAFTAR ISI

LEMBAR PENGESAHAN v
Abstrak vii
Abstract ix
KATA PENGANTAR xi
DAFTAR ISI xiii
DAFTAR GAMBAR xvii
DAFTAR TABEL xix
DAFTAR KODE SUMBER xxv
BAB I PENDAHULUAN 1
1.1 Latar Belakang 1
1.2 Rumusan Masalah 3
1.3 Tujuan 3
1.4 Manfaat 3
1.5 Batasan Masalah 4
1.6 Metodologi 5
1.7 Sistematika Penulisan Laporan Tugas Akhir 6
BAB II TINJAUAN PUSTAKA 9
2.1 *Dunstan Baby Language* 9
2.2 *Mel-Frequency Cepstral Coefficients* 10
2.2.1 *Pre-Emphasis* 11
2.2.2 *Frame Blocking* 11
2.2.3 *Windowing* 12
2.2.4 *Fast Fourier Transform* 13
2.2.5 *Mel Frequency Warping* 13
2.2.6 *Discrete Cosine Transform* 14
2.2.7 *Cepstral Liftering* 14
2.3 Normalisasi 15
2.3.1 *Standard Deviation Normalization* 15
2.3.2 *Min-Max Normalization* 15
2.3.3 *Decimal Normalization* 16
2.4 *K-Nearest Neighbor* 16
2.5 *Naive Bayes* 17
2.6 *Neural Network* 18

2.7 *Support Vector Machine* 21
2.8 tuneR 22
2.9 caret 23
2.10 e1071 23
2.11 neuralnet 23
2.12 Penelitian terkait Deteksi Suara Tangis Bayi 23
BAB III PERANCANGAN SISTEM 27
3.1 Desain Metode Secara Umum 27
3.2 Perancangan Data 28
3.2.1 Data Masukan 29
3.2.2 Data Proses 31
3.2.3 Data Keluaran 34
3.3 Perancangan Uji Coba Pemilihan Metode Terbaik 35
3.4 Perancangan Uji Coba dengan Data Latih Seimbang pada Tiap Kelas 37
3.5 Perancangan Uji Coba dengan Metode Klasifikasi Lain 38
3.6 Perancangan Sistem 39
3.6.1 Arsitektur Perangkat Lunak 39
3.6.2 Deskripsi Umum Kebutuhan 40
3.6.3 Kebutuhan Fungsional 41
3.6.4 Aliran Informasi 42
3.6.5 Deskripsi Proses 45
BAB IV IMPLEMENTASI 49
4.1 Lingkungan Implementasi 49
4.1.1. Lingkungan Implementasi Perangkat Keras 49
4.1.2. Lingkungan Implementasi Perangkat Lunak 49
4.2 Implementasi Pemilihan Metode Terbaik 49
4.2.1 Implementasi Proses Ekstraksi Fitur MFCC 50
4.2.2 Implementasi Pembagian proporsi data latih dan data Uji 51
4.2.3 Implementasi Proses Normalisasi 53
4.2.4 Implementasi Proses Klasifikasi *K-nearest Neighbor* 53
4.2.5 Implementasi Perhitungan Akurasi 54

4.3 Implementasi Uji Coba dengan Metode Klasifikasi lain. 55
4.3.1 Implementasi Proses Klasifikasi *Naive Bayes* 55
4.3.2 Implementasi Proses Klasifikasi *Neural Network* 56
4.3.3 Implementasi Proses Klasifikasi *Support Vector Machine* 57
4.4 Implementasi Perancangan Sistem 58
4.4.1 Implementasi Pembentukan Data Latih 58
4.4.2 Implementasi Menggungah Data Uji 61
4.4.3 Implementasi Ekstraksi Fitur Data Uji 61
4.4.4 Implementasi Klasifikasi KNN 63
4.4.5 Implementasi GUI 64
BAB V UJI COBA DAN EVALUASI 67
5.1 Lingkungan Uji Coba 67
5.2 Data Uji Coba 67
5.3 Skenario Uji Coba Pemilihan Metode 68
5.3.1 Skenario 1 70
5.3.2 Skenario 2 71
5.3.3 Skenario 3 72
5.3.4 Skenario 4 73
5.3.5 Skenario 5 74
5.3.6 Skenario 6 75
5.3.7 Skenario 7 76
5.3.8 Skenario 8 77
5.3.9 Skenario 9 78
5.3.10 Skenario 10 79
5.3.11 Skenario 11 80
5.3.12 Skenario 12 81
5.3.13 Skenario 13 82
5.3.14 Skenario 14 83
5.3.15 Skenario 15 84
5.4 Skenario Uji Coba dengan Data Latih Seimbang pada Tiap Kelas 85
5.5 Skenario Uji Coba dengan Metode Klasifikasi Lain 86

5.6 Skenario Uji Coba Aplikasi Pendeteksi Suara Tangis Bayi 87
5.7 Analisis Hasil Uji Coba Pemilihan Metode 88
5.8 Analisis Hasil Uji Coba dengan Data Latih Seimbang pada Tiap Kelas 90
5.9 Analisis Hasil Uji Coba dengan Metode Klasifikasi Lain 91
5.10 Analisis Hasil Uji Coba Aplikasi 91
BAB VI KESIMPULAN DAN SARAN 93
6.1 Kesimpulan Uji Coba Pemilihan Metode 93
6.2 Kesimpulan Uji Coba dengan Data Latih Seimbang 93
6.3 Kesimpulan Uji Coba dengan Metode Klasifikasi Lain 94
6.4 Kesimpulan Pembuatan Aplikasi 94
6.5 Saran 94
DAFTAR PUSTAKA 95
LAMPIRAN 99
7.1 Hasil Uji Klasifikasi *K-nearest Neighbor* Data Latih 90% Data Uji 10% dengan berbagai skenario 99
7.2 Hasil Uji Klasifikasi *K-nearest Neighbor* Data Latih 85% Data Uji 15% dengan berbagai skenario 103
7.3 Hasil Uji Klasifikasi *K-nearest Neighbor* Data Latih 80% Data Uji 20% dengan berbagai skenario 107
7.4 Hasil Uji Klasifikasi *K-nearest Neighbor* Data Latih 75% Data Uji 25% dengan berbagai skenario 110
7.5 Hasil Uji Klasifikasi *K-nearest Neighbor* Data Latih 70% Data Uji 30% dengan berbagai skenario 115
7.6 Hasil Uji Coba Pemilihan Metode Terbaik 118
7.7 Hasil Uji Coba dengan Data Latih Seimbang pada Tiap Kelas 119
7.8 Hasil Uji Coba dengan Metode Klasifikasi Lain 123
7.9 Hasil Uji Coba Aplikasi Pendeteksi Suara Tangis Bayi 125
BIODATA PENULIS 131

DAFTAR GAMBAR

Gambar 2.1 Diagram Alur Ekstraksi Fitur MFCC 10
Gambar 2.2 Ilustrasi *Frame Blocking* [8]..... 11
Gambar 2.3 Ilustrasi *Backpropagation Neural Network*..... 19
Gambar 2.4 Contoh Klasifikasi dengan Metode *One-againts-one* 22
Gambar 3.1 Diagram Proses Klasifikasi Suara Tangis Bayi .. 28
Gambar 3.2 Hasil *Plot* Audio Salah Satu Tangis Bayi Jenis "Eairh"..... 29
Gambar 3.3 Hasil *Plot* Audio Salah Satu Tangis Bayi Jenis "Eh" 30
Gambar 3.4 Hasil *Plot* Audio Salah Satu Tangis Bayi Jenis "Heh"..... 30
Gambar 3.5 Hasil *Plot* Audio Salah Satu Tangis Bayi Jenis "Neh"..... 30
Gambar 3.6 Hasil *Plot* Audio Salah Satu Tangis Bayi Jenis "Owh"..... 31
Gambar 3.7 Alur Ekstraksi Fitur dan Normalisasi 33
Gambar 3.8 Alur Klasifikasi Data..... 34
Gambar 3.9 Diagram Alur Uji Coba macam-macam Klasifikasi dan Perhitungan Akurasi klasifikasi..... 36
Gambar 3.10 Diagram Arsietektur Perangkat Lunak “Aplikasi Pendeteksi Suara Tangisan Bayi” 39
Gambar 3.11 DFD Level 0 Aplikasi Pendeteksi Suara Tangis Bayi 42
Gambar 3.12 DFD Level 1 Aplikasi Pendeteksi Suara Tangis Bayi 43
Gambar 3.13 DFD Level 2 Proses 1 "extractAudio" 44
Gambar 4.1 GUI Aplikasi Pendeteksi Tangis Bayi..... 65
Gambar 4.2 GUI Aplikasi Pendeteksi Tangis Bayi saat Menerima *Input* 65
Gambar 5.1 Grafik Rata-rata Hasil Uji Data Latih 90% Data Uji 10% *Decimal Normalization* 70

Gambar 5.2 Rata-rata Hasil Uji Data Latih 90% Data Uji 10% *Min-Max Normalization* 71
Gambar 5.3 Grafik Rata-rata Hasil Uji Data Latih 90% Data Uji 10% *Standard Deviation Normalization* 72
Gambar 5.4 Grafik Rata-rata Hasil Uji Data Latih 85% Data Uji 15% *Decimal Normalization* 73
Gambar 5.5 Grafik Rata-rata Hasil Uji Data Latih 85% Data Uji 15% *Min-Max Normalization.* 74
Gambar 5.6 Grafik Rata-rata Hasil Uji Data Latih 85% Data Uji 15% *Standard Deviation Normalization* 75
Gambar 5.7 Grafik Rata-rata Hasil Uji Data Latih 80% Data Uji 20% *Decimal Normalization* 76
Gambar 5.8 Grafik Rata-rata Hasil Uji Data Latih 80% Data Uji 20% *Min-Max Normalization*.....77
Gambar 5.9 Rata-rata Grafik Hasil Uji Data Latih 80% Data Uji 20% *Standard Deviation Normalization* 78
Gambar 5.10 Grafik Hasil Uji Data Latih 75% Data Uji 25% *Decimal Normalization* 79
Gambar 5.11 Grafik Rata-rata Hasil Uji Data Latih 75% Data Uji 25% *Min-Max Normalization*.....80
Gambar 5.12 Grafik Rata-rata Hasil Uji Data Latih 75% Data Uji 25% *Standard Deviation Normalization* 81
Gambar 5.13 Grafik Rata-rata Hasil Uji Data Latih 70% Data Uji 30% *Decimal Normalization* 82
Gambar 5.14 Grafik Rata-rata Hasil Uji Data Latih 70% Data Uji 30% *Min-Max Normalization*.....83
Gambar 5.15 Grafik Rata-rata Hasil Uji Data Latih 70% Data Uji 30% *Standard Deviation Normalization* 84

DAFTAR TABEL

Tabel 2.1 Contoh SVM 6 Biner dengan Metode *One-againts-one* 22
Tabel 3.1 Tabel Jumlah Dataset Suara Tangis Bayi 31
Tabel 4.1 Pembagian Proporsi Data Latih 52
Tabel 4.2 Pembagian Proporsi Data Uji 52
Tabel 5.1 Spesifikasi Lingkungan Uji Coba 67
Tabel 5.2 Rata-rata Hasil Uji Data Latih 90% Data Uji 10% *Decimal Normalization* 70
Tabel 5.3 Rata-rata Hasil Uji Data Latih 90% Data Uji 10% *Min-Max Normalization* 71
Tabel 5.4 Rata-rata Hasil Uji Data Latih 90% Data Uji 10% *Standard Deviation Normalization* 72
Tabel 5.5 Rata-rata Hasil Uji Data Latih 85% Data Uji 15% *Decimal Normalization* 73
Tabel 5.6 Rata-rata Hasil Uji Data Latih 85% Data Uji 15% *Min-Max Normalization* 74
Tabel 5.7 Rata-rata Hasil Uji Data Latih 85% Data Uji 15% *Standard Deviation Normalization* 75
Tabel 5.8 Rata-rata Hasil Uji Data Latih 80% Data Uji 20% *Decimal Normalization* 76
Tabel 5.9 Rata-rata Hasil Uji Data Latih 80% Data Uji 20% *Min-Max Normalization* 77
Tabel 5.10 Rata-rata Hasil Uji Data Latih 80% Data Uji 20% *Standard Deviation Normalization* 78
Tabel 5.11 Rata-rata Hasil Uji Data Latih 75% Data Uji 25% *Decimal Normalization* 79
Tabel 5.12 Rata-rata Hasil Uji Data Latih 75% Data Uji 25% *Min-Max Normalization* 80
Tabel 5.13 Rata-rata Hasil Uji Data Latih 75% Data Uji 25% *Standard Deviation Normalization* 81
Tabel 5.14 Rata-rata Hasil Uji Data Latih 70% Data Uji 30% *Decimal Normalization* 82

Tabel 5.15 Rata-rata Hasil Uji Data Latih 70% Data Uji 30% *Min-Max Normalization* 83
Tabel 5.16 Rata-rata Hasil Uji Data Latih 70% Data Uji 30% *Standard Deviation Normalization* 84
Tabel 5.17 Tabel Pembagian Data Latih Seimbang pada Tiap Kelas 85
Tabel 5.18 Perbedaan Hasil Klasifikasi KNN dengan Variasi Nilai *k* pada Data Latih Seimbang 85
Tabel 5.19 Perbandaingan Akurasi Metode Klasifikasi 86
Tabel 5.20 *Confusion Matrix* Uji Coba Aplikasi Pendeteksi Tangis Bayi 87
Tabel 5.21 Akurasi Tiap Kelas Aplikasi Pendeteksi Tangis Bayi 87
Tabel 5.22 *Confusion Matrix* Uji Coba Pemilihan Metode Terbaik dengan Parameter *wintime*=0,08 detik, Data Latih 85% Data Uji 15%, Nilai *k*=1 pada *K-nearest Neighbor*, dan Normalisasi Ekstraksi Fitur dengan *Standard Deviation Normalization* 88
Tabel 5.23 Akurasi Rata-rata Penggunaan Kombinasi *wintime* pada 15 skenario percobaan 89
Tabel 7.1 Hasil Uji Data Latih 90% Data Uji 10% *wintime* 0,02 *Decimal Normalization* 99
Tabel 7.2 Hasil Uji Data Latih 90% Data Uji 10% *wintime* 0,02 *Min-Max Normalization* 99
Tabel 7.3 Hasil Uji Data Latih 90% Data Uji 10% *wintime* 0,02 *Standard Deviation Normalization* 99
Tabel 7.4 Hasil Uji Data Latih 90% Data Uji 10% *wintime* 0,04 *Decimal Normaliation* 100
Tabel 7.5 Hasil Uji Data Latih 90% Data Uji 10% *wintime* 0,04 *Min-Max Normalization* 100
Tabel 7.6 Hasil Uji Data Latih 90% Data Uji 10% *wintime* 0,04 *Standard Deviation Normalization* 100
Tabel 7.7 Hasil Uji Data Latih 90% Data Uji 10% *wintime* 0,06 *Decimal Normalization* 101

Tabel 7.8 Hasil Uji Data Latih 90% Data Uji 10% *wintime* 0,06 *Min-Max Normalization* 101
Tabel 7.9 Hasil Uji Data Latih 90% Data Uji 10% *wintime* 0,06 *Standard Deviation Normalization* 101
Tabel 7.10 Hasil Uji Data Latih 90% Data Uji 10% *wintime* 0,08 *Decimal Normalization* 102
Tabel 7.11 Hasil Uji Data Latih 90% Data Uji 10% *wintime* 0,08 *Min-Max Normalization* 102
Tabel 7.12 Hasil Uji Data Latih 90% Data Uji 10% *wintime* 0,08 *Standard Deviation Normalization* 102
Tabel 7.13 Hasil Uji Data Latih 85% Data Uji 15% *wintime* 0,02 *Decimal Normalization* 103
Tabel 7.14 Hasil Uji Data Latih 85% Data Uji 15% *wintime* 0,02 *Min-Max Normalization* 103
Tabel 7.15 Hasil Uji Data Latih 85% Data Uji 15% *wintime* 0,02 *Standard Deviation Normalization* 103
Tabel 7.16 Hasil Uji Data Latih 85% Data Uji 15% *wintime* 0,04 *Decimal Normalization* 104
Tabel 7.17 Hasil Uji Data Latih 85% Data Uji 15% *wintime* 0,04 *Min-Max Normalization* 104
Tabel 7.18 Hasil Uji Data Latih 85% Data Uji 15% *wintime* 0,04 *Standard Deviation Normalization* 104
Tabel 7.19 Hasil Uji Data Latih 85% Data Uji 15% *wintime* 0,06 *Decimal Normalization* 105
Tabel 7.20 Hasil Uji Data Latih 85% Data Uji 15% *wintime* 0,06 *Min-Max Normalization* 105
Tabel 7.21 Hasil Uji Data Latih 85% Data Uji 15% *wintime* 0,06 *Standard Deviation Normalization* 105
Tabel 7.22 Hasil Uji Data Latih 85% Data Uji 15% *wintime* 0,08 *Decimal Normalization* 106
Tabel 7.23 Hasil Uji Data Latih 85% Data Uji 15% *wintime* 0,08 *Min-Max Normalization* 106
Tabel 7.24 Hasil Uji Data Latih 85% Data Uji 15% *wintime* 0,08 *Standard Deviation Normalization* 106

Tabel 7.25 Hasil Uji Data Latih 80% Data Uji 20% *wintime* 0,02 *Decimal Normalization* 107
Tabel 7.26 Hasil Uji Data Latih 80% Data Uji 20% *wintime* 0,02 *Min-Max Normalization* 107
Tabel 7.27 Hasil Uji Data Latih 80% Data Uji 20% *wintime* 0,02 *Standard Deviation Normalization* 107
Tabel 7.28 Hasil Uji Data Latih 80% Data Uji 20% *wintime* 0,04 *Decimal Normalization* 108
Tabel 7.29 Hasil Uji Data Latih 80% Data Uji 20% *wintime* 0,04 *Min-Max Normalization* 108
Tabel 7.30 Hasil Uji Data Latih 80% Data Uji 20% *wintime* 0,04 *Standard Deviation Normalization* 108
Tabel 7.31 Hasil Uji Data Latih 80% Data Uji 20% *wintime* 0,06 *Decimal Normalization* 109
Tabel 7.32 Hasil Uji Data Latih 80% Data Uji 20% *wintime* 0,06 *Min-Max Normalization* 109
Tabel 7.33 Hasil Uji Data Latih 80% Data Uji 20% *wintime* 0,06 *Standard Deviation Normalization* 109
Tabel 7.34 Hasil Uji Data Latih 80% Data Uji 20% *wintime* 0,08 *Decimal Normalization* 110
Tabel 7.35 Hasil Uji Data Latih 80% Data Uji 20% *wintime* 0,08 *Min-Max Normalization* 110
Tabel 7.36 Hasil Uji Data Latih 80% Data Uji 20% *wintime* 0,08 *Standard Deviation Normalization* 110
Tabel 7.37 Hasil Uji Data Latih 75% Data Uji 25% *wintime* 0,02 *Decimal Normalization* 111
Tabel 7.38 Hasil Uji Data Latih 75% Data Uji 25% *wintime* 0,02 *Min-Max Normalization* 111
Tabel 7.39 Hasil Uji Data Latih 75% Data Uji 15% *wintime* 0,02 *Standard Deviation Normalization* 111
Tabel 7.40 Hasil Uji Data Latih 75% Data Uji 25% *wintime* 0,04 *Decimal Normalization* 112
Tabel 7.41 Hasil Uji Data Latih 75% Data Uji 15% *wintime* 0,04 *Min-Max Normalization* 112

Tabel 7.42 Hasil Uji Data Latih 75% Data Uji 15% *wintime* 0,04 *Standard Deviation Normalization* 112
Tabel 7.43 Hasil Uji Data Latih 75% Data Uji 25% *wintime* 0,06 *Decimal Normalization* 113
Tabel 7.44 Hasil Uji Data Latih 75% Data Uji 25% *wintime* 0,06 *Min-Max Normalization* 113
Tabel 7.45 Hasil Uji Data Latih 75% Data Uji 25% *wintime* 0,06 *Standard Deviation Normalization* 113
Tabel 7.46 Hasil Uji Data Latih 75% Data Uji 25% *wintime* 0,08 *Decimal Normalization* 114
Tabel 7.47 Hasil Uji Data Latih 75% Data Uji 25% *wintime* 0,08 *Min-Max Normalization* 114
Tabel 7.48 Hasil Uji Data Latih 75% Data Uji 25% *wintime* 0,08 *Standard Deviation Normalization* 114
Tabel 7.49 Hasil Uji Data Latih 70% Data Uji 30% *wintime* 0,02 *Decimal Normalization* 115
Tabel 7.50 Hasil Uji Data Latih 70% Data Uji 30% *wintime* 0,02 *Min-Max Normalization* 115
Tabel 7.51 Hasil Uji Data Latih 70% Data Uji 30% *wintime* 0,02 *Standard Deviation Normalization* 115
Tabel 7.52 Hasil Uji Data Latih 70% Data Uji 30% *wintime* 0,04 *Decimal Normalization* 116
Tabel 7.53 Hasil Uji Data Latih 70% Data Uji 30% *wintime* 0,04 *Min-Max Normalization* 116
Tabel 7.54 Hasil Uji Data Latih 70% Data Uji 30% *wintime* 0,04 *Standard Deviation Normalization* 116
Tabel 7.55 Hasil Uji Data Latih 70% Data Uji 30% *wintime* 0,06 *Decimal Normalization* 117
Tabel 7.56 Hasil Uji Data Latih 70% Data Uji 30% *wintime* 0,06 *Min-Max Normalization* 117
Tabel 7.57 Hasil Uji Data Latih 70% Data Uji 30% *wintime* 0,06 *Standard Deviation Normalization* 117
Tabel 7.58 Hasil Uji Data Latih 70% Data Uji 30% *wintime* 0,08 *Decimal Normalization* 118

Tabel 7.59 Hasil Uji Data Latih 70% Data Uji 30% *wintime* 0,08 *Min-Max Normalization* 118
Tabel 7.60 Hasil Uji Data Latih 70% Data Uji 30% *wintime* 0,08 *Standard Deviation Normalization* 118
Tabel 7.61 Hasil Uji Coba Pemilihan Metode Terbaik 118
Tabel 7.62 Hasil Uji Coba KNN dengan Data Latih Seimbang 119
Tabel 7.63 *Confusion Matrix* Hasil Uji Data Latih Seimbang dengan KNN *k*=1 122
Tabel 7.64 *Confusion Matrix* Hasil Uji Data Latih Seimbang dengan KNN *k*=2.....122
Tabel 7.65 *Confusion Matrix* Hasil Uji Data Latih Seimbang dengan KNN *k*=3.....122
Tabel 7.66 *Confusion Matrix* Hasil Uji Data Latih Seimbang dengan KNN k=4 123
Tabel 7.67 Confusion Matrix Hasil Uji Data Latih Seimbang dengan KNN k=5 123
Tabel 7.68 Hasil Klasifikasi dengan metode *Naive Bayes*, *Neural Network*, dan *Support Vector Machine* 123
Tabel 7.69 *Confusion Matrix* Hasil Uji Klasifikasi *Naive Bayes* 124
Tabel 7.70 *Confusion Matrix* Hasil Uji Neural Network 124
Tabel 7.71 *Confusion Matrix* Hasil Uji *Support Vector Machine* 125
Tabel 7.72 Hasil Klasifikasi Aplikasi Pendeteksi Suara Tangis Bayi 125

DAFTAR KODE SUMBER

Kode Sumber 4.1 Implementasi Ekstraksi Fitur pada Pemiihan Metode Terbaik 51
Kode Sumber 4.2 Implementasi Pembagian Data Latih dan Data Uji pada Pemilihan Metode Terbaik 52
Kode Sumber 4.3 Implementasi *Standard Deviation Normalization* pada Pemiihan Metode Terbaik..... 53
Kode Sumber 4.4 Implementasi *Decimal Normalization* pada Pemiihan Metode Terbaik 53
Kode Sumber 4.5 Implementasi *Min-Max Normalization* pada Pemiihan Metode Terbaik 53
Kode Sumber 4.6 Implementasi *K-nearest Neighbor* Pada Pemiihan Metode Terbaik 54
Kode Sumber 4.7 Implementasi Perhitungan Akurasi pada Pemilihan Metode Terbaik 55
Kode Sumber 4.8 Implementasi Klasifikasi *Naive Bayes* 56
Kode Sumber 4.9 Implementasi Klasifikasi *Neural Network* 57
Kode Sumber 4.10 Implementasi Klasifikasi *Support Vector Machine* 58
Kode Sumber 4.11 Implementasi Pembentukan Data Latih pada Pembuatan Aplikasi 61
Kode Sumber 4.12 Implementasi Mengunggah Data Uji pada Aplikasi 61
Kode Sumber 4.13 Implementasi Ekstraksi Fitur Data Uji 62
Kode Sumber 4.14 Implementasi KNN pada Aplikasi 64
Kode Sumber 4.15 Implementasi GUI Aplikasi 65

[Halaman ini sengaja dikosongkan]

BAB I
PENDAHULUAN

1.1 Latar Belakang

Suara tangisan bayi merupakan sebuah tanda dari bayi yang mengalami suatu masalah. Suara tangisan tersebut dapat digunakan untuk mengidentifikasi masalah pada bayi, seperti kelaparan, kesakitan, rasa kantuk, rasa tidak nyaman, kedingingan atau kepanasan, dan lain-lain. Dari suara tangis itu, seseorang yang telah terbiasa mengasuh bayi dapat mengerti apa maksud tangisan tersebut. Namun, tidak semua orang terbiasa mengasuh bayi, terutama bagi pasangan muda yang baru memiliki anak pertama. Pasangan muda bisa bertanya pada orang tua atau kerabat mereka bagaimana cara menenangkan bayi yang sedang menangis, namun orang tua dari pasangan tersebut belum tentu bisa memahami arti tangisan tersebut sehingga bayi kerap kali masih tetap menangis meskipun sudah dihibur atau ditenangkan dengan berbagai cara.

Dari permasalahan tentang tangis bayi, ada beberapa penelitian yang mengangkat tema bagaimana cara untuk mendeteksi masalah pada tangis bayi. Beberapa penelitian tersebut ialah "*Application of Neuro-Fuzy Approaches to Recognition and Classification of Infant Cry*" [1], "Identifikasi Arti Tangis Bayi Versi *Dunstan Baby Language* Menggunakan Jarak Terpendek dari Jarak *Mahalanobis*" [2], dan "*Normal and Hypoacoustic Infant Cry Signal Classification Using Time-Frequency Analysis and General Regression Neural Network*" [3]. Penelitian [1] dan [2] menggunakan *Dunstan Baby Language* sebagai acuan untuk mengidentifikasi jenis tangis bayi, sementara penelitian [3] fokus pada identifikasi suara tangis bayi yang normal atau tuli.

Pada penelitian [1], metode yang digunakan ialah *Perceptual Linear Prediction* untuk ekstraksi fitur, *Neuro-Fuzzy* untuk pengenalan suara, dan *K-nearest Neighbor* untuk mengklasifikasi masing-masing jenis tangis bayi. Hasil dari

penelitian ini adalah *Neuro-Fuzzy* merupakan algoritma yang terbaik untuk pengenalan jenis tangis bayi jika dibandingkan dengan metode lainnya yang diuji pada penelitian [1].

Pada penelitian [2], metode yang digunakan ialah *Mel-Frequency Cepstrum Coefficients* untuk ekstraksi fitur, pemodelan *codebook* dengan *K-Means clustering* untuk pecocokan fitur, dan perhitungan jarak *Mahalanobis* untuk untuk identifikasi jenis tangis bayi. Penelitian ini menghasilkan akurasi terbaik sebesar 83% untuk identifikasi 5 jenis tangis bayi berdasarkan *Dunstan Baby Language*.

Pada penelitian [3], metode yang digunakan ialah *Short-time Fourier Transform* untuk ekstrasi fitur dan *General Regression Neural Network* (GRNN) untuk mengklasifikasi suara tangis bayi yang normal atau tuli. Hasil dari penelitian tersebut ialah GRNN merupakan *classifier* terbaik jika dibandingkan dengan metode yang klasifikasi yang diuji pada penelitian [3].

Berdasarkan tiga penelitian tersebut, dapat disimpulkan bahwa penelitian tentang arti tangis bayi terus berkembang, baik dari segi metode ekstraksi fitur maupun metode identifikasi. Namun, saat ini masih belum ada penelitian yang mengembangkan aplikasi pendeteksi tangis bayi berbasis web agar dapat digunakan oleh banyak orang untuk mengetahui arti suara tangis bayi.

Maka dari itu, untuk mengenal arti tangis bayi berbasis *Dunstan Baby Language*, penulis menerapkan metode ekstraksi fitur *Mel-Frequency Cepstrum Coefficient*s (MFCC) dan metode klasifikasi *K-nearest Neighbor* pada aplikasi pendeteksi suara tangisan bayi berbasis web. Metode ekstraksi fitur MFCC dipilih karena telah digunakan oleh banyak peneliti untuk menganalisa pola kecocokan suara dan tersedia dalam bentuk *library* pada beberapa bahasa pemrograman populer, sedangkan metode klasifikasi *K-nearest Neighbor* dipilih karena waktu yang diperlukan untuk komputasi klasifikasi pada data uji yang cukup singkat dengan komposisi data latih yang cukup banyak.

Pembuatan aplikasi berbasis web ini dipilih karena mempertimbangkan kemudahan penggunaan aplikasi di manapun dan kapanpun. Selain itu, pengguna tidak perlu mengalokasikan ruang pada media penyimpanan untuk pemasangan aplikasi, baik pada komputer atau perangkat bergerak lainnya, seperti *smartphone* atau tablet. Dengan adanya aplikasi tersebut, pengguna diharapkan dapat mengetahui arti dari suara tangisan bayi yang bersangkutan sehingga dapat memberikan penanganan yang tepat terhadap suara tangisan tersebut.

1.2 Rumusan Masalah

Rumusan masalah yang diangkat dalam Tugas Akhir ini dapat dipaparkan sebagai berikut:

1. Bagaimana cara mengetahui arti suara tangis bayi secara tepat dan efisien.
2. Bagaimana membuat aplikasi yang menerapkan metode klasifikasi suara tangis bayi berbasis web.

1.3 Tujuan

Tujuan dari Tugas Akhir ini dapat dijabarkan sebagai berikut:

1. Membuat aplikasi yang dapat mengklasifikasikan suara tangis bayi menjadi 4 kategori berdasarkan *Dunstan Baby Language*.
2. Memberikan *output* jenis tangisan bayi kepada pengguna sehingga pengguna dapat memberikan tindakan penanganan yang tepat pada tangisan tersebut.

1.4 Manfaat

Tugas Akhir ini dibuat untuk memberikan beberapa manfaat sebagai berikut:

1. Memberikan *output* berupa jenis tangisan bayi kepada pengguna.
2. Mempermudah pengguna, terutama pasangan muda yang baru memiliki anak pertama untuk mengetahui makna

tangisan bayi sehingga dapat memberikan tindakan penanganan yang tepat pada bayi.

1.5 Batasan Masalah

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

1. Aplikasi akan dikembangkan pada *platform* web dengan menggunakan bahasa pemrograman R dan *package* Shiny.
2. Aplikasi hanya dapat mengklasifikasikan suara tangis bayi pada usia 0-3 bulan. Jika suara tangis bayi yang diujikan lebih dari usia 3 bulan, maka hasil klasifikasinya tidak dapat akurat karena dataset pada penelitian ini dibatasi pada bayi berusia maksimal 3 bulan.
3. Aplikasi hanya dapat mengklasifikasikan rekaman suara tangis bayi dengan spesifikasi sebagai berikut:
    a. *Bitrate*: 16 bit
    b. *Channel*: mono
    c. *Frekuensi*: 11205 Hz
    d. Durasi rekaman: maksimal 10 detik.
4. Aplikasi hanya dapat mengklasifikasikan suara tangis bayi menjadi 4 kategori berdasarkan *Dunstan Baby Language*, yaitu:
    a. "Eairh" atau "Eh" yang berarti bayi ingin buang air besar, buang angin, atau bersendawa.
    b. "Heh" yang berarti bayi merasa tidak nyaman, saatnya untuk mengganti popok.
    c. "Neh" yang berarti bayi merasa lapar.
    d. "Owh" yang berarti bayi mulai mengantuk.
5. Dataset yang digunakan dalam penelitian ini diekstrak dari 139 rekaman suara tangis bayi dengan algoritma *MFCC* yang dinormalisasi dengan algoritma *Decimal Normalization*, *Min-Max Normalization*, atau *Standard Deviation Normalization*.
6. Metode klasifikasi yang digunakan adalah *K-nearest Neighbor*.

1.6 Metodologi

Tahapan-tahapan yang dilakukan dalam pengerjaan Tugas Akhir ini adalah sebagai berikut:

1. Penyusunan proposal Tugas Akhir.

   Tahap awal untuk memulai pengerjaan Tugas Akhir adalah penyusunan proposal Tugas Akhir. Pada proposal Tugas Akhir tersebut berisi rencana Tugas Akhir yang akan dikerjakan sebagai syarat untuk menyelesaikan studi dan meraih gelar Strata-1 Teknik Informatika. Pada proposal tersebut dijelaskan secara garis besar tentang tahapan anilisis data suara yang akan dilakukan, diantaranya ekstraksi fitur, normalisasi data, dan terakhir klasifikasi.

2. Studi literatur

   Tahap ini merupakan tahap pengumpulan informasi yang diperlukan untuk pengerjaan Tugas Akhir sekaligus mempelajarinya. Beberapa literatur yang perlu dipelajari lebih dalam lagi untuk proses pengolahan data suara khususnya *Dunstan Baby Language*, ekstraksi fitur suara tangis bayi dengan algoritma *MFCC (Mel-frequency cepstral coefficients)*, normalisasi data ekstraksi fitur dengan *Decimal Normalization*, *Min-Max Normalization*, dan *Standard Deviation Normalization*, serta proses klasifikasi menggunakan metode *K-nearest Neighbors*, *Naive Bayes*, *Resilient Backpropagation Neural Network*, dan *Support Vector Machine*.

3. Analisis dan desain perangkat lunak

   Pada tahap ini akan dilakukan analisis dan desain perancangan aplikasi sesuai dengan tujuan yang dijabarkan, kemudian disesuaikan dengan metode yang tepat, hal ini dimaksudkan agar nantinya ketika diimplementasikan ke dalam aplikasi dapat berjalan sesuai yang diharapkan.

4. Implementasi perangkat lunak

   Perangkat lunak ini akan dibangun dengan menggunakan bahasa pemrograman R, *package* tuneR, *package* caret, dan *package* Shiny dengan kakas bantu Rstudio.

5. Pengujian dan evaluasi

   Aplikasi akan diuji setelah selesai diimplementasikan menggunakan skenario yang sudah dipersiapkan. Pengujian dan evaluasi akan dilakukan dengan melihat kesesuaian dengan perencanaan. Dengan melakukan pengujian dan evaluasi dimaksudkan juga untuk mengevaluasi jalannya program, mencari masalah yang mungkin timbul dan mengadakan perbaikan jika terdapat kesalahan.

6. Penyusunan buku Tugas Akhir.

   Pada tahap ini dilakukan penyusunan laporan yang menjelaskan dasar teori dan metode yang digunakan dalam Tugas Akhir ini serta hasil dari implementasi aplikasi perangkat lunak yang telah dibuat.

1.7 Sistematika Penulisan Laporan Tugas Akhir

Buku Tugas Akhir ini merupakan laporan secara lengkap mengenai Tugas Akhir yang telah dikerjakan baik dari sisi teori, rancangan, maupun implementasi sehingga memudahkan bagi pembaca dan juga pihak yang ingin mengembangkan lebih lanjut. Sistematika penulisan buku Tugas Akhir secara garis besar antara lain:

Bab I Pendahuluan

Bab ini berisi penjelasan latar belakang, rumusan masalah, batasan masalah dan tujuan pembuatan Tugas Akhir. Selain itu, metodologi pengerjaan dan

sistematika penulisan laporan Tugas Akhir juga terdapat di dalamnya.

Bab II Tinjauan Pustaka

Bab ini berisi penjelasan secara detail mengenai dasar-dasar penunjang dan teori-teori yang digunakan untuk mendukung pembuatan Tugas Akhir ini.

Bab III Perancangan Perangkat Lunak

Bab ini berisi penjelasan tentang rancangan dari sistem yang akan dibangun. Rancangan ini digambarkan dalam bentuk diagram dan dituliskan dalam bentuk *pseudocode* (bila diperlukan).

Bab IV Implementasi

Bab ini berisi penjelasan implementasi dari rancangan yang telah dibuat pada bab sebelumnya. Implementasi disajikan dalam bentuk *code* secara keseluruhan disertai dengan penjelasannya.

Bab V Uji Coba Dan Evaluasi

Bab ini berisi penjelasan mengenai data hasil percobaan dan pembahasan mengenai hasil percobaan yang telah dilakukan.

Bab VI Kesimpulan Dan Saran

Bab ini merupakan bab terakhir yang menyampaikan kesimpulan dari hasil uji coba yang dilakukan dan saran untuk pengembangan perangkat lunak ke depannya.

[Halaman ini sengaja dikosongkan]

BAB II
TINJAUAN PUSTAKA

Dalam bab ini akan dijelaskan mengenai teori-teori yang merupakan dasar dari pembangunan sistem. Selain itu, terdapat penjelasan yang menunjang pengerjaan Tugas Akhir ini sehingga dapat memberikan gambaran secara umum sistem yang akan dibangun.

2.1 *Dunstan Baby Language*

Dunstan Baby Language [4] adalah gagasan tentang arti tangisan bayi yang dicetuskan oleh Priclilla Dunstan pada tahun 2006. Menurut Pricilla, terdapat 5 jenis tangisan bayi universal (berlaku pada seluruh bayi di dunia) yang memiliki arti masing-masing, yaitu:

- ***Neh (Aku lapar)*** - Seorang bayi menggunakan refleks suara "Neh" untuk berkomunikasi bahwa ia sedang kelaparan. Suara yang dihasilkan berasal dari refleks menghisap dan lidah didorong di atap mulut.
- ***Owh (Aku mengantuk)*** - Seorang bayi menggunakan refleks suara "Owh" untuk berkomunikasi bahwa ia lelah. Suara yang dihasilkan terdegar seperti orang yang sedang menguap.
- ***Heh (Aku merasa tidak nyaman)*** - Seorang bayi menggunakan refleks suara "Heh" untuk berkomunikasi bahwa ia merasa tertekan, tidak nyaman, atau membutuhkan popok baru. Suara yang dihasilkan ialah respon pada masalah kulit bayi, biasanya rasa kepanasan (disertai keringat) atau rasa gatal.
- ***Eairh (Ada angin di perutku)*** - Seorang bayi menggunakan refleks suara "Eairh" untuk berkomunikasi bahwa ia ingin buang angin atau masalah perut. Suara tersebut dihasilkan ketika bayi tak dapat bersendawa, kemudian angin tersebut masuk ke

perut dan menekan otot-otot perut agar angin tersebut dapat dikeluarkan. Seringkali, suara ini juga mengindikasaikan bayi ingin buang air besar yang disertai dengan gerakan menekuk lutut ke tubuhnya.

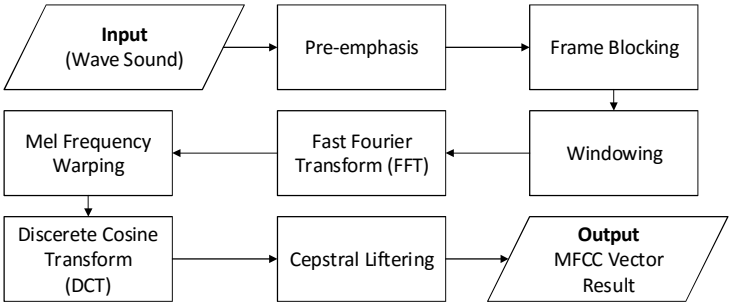
- ***Eh (Aku ingin bersendawa)*** - Seorang bayi menggunakan suara refleks "Eh" untuk berkomunikasi bahwa ia ingin bersendawa. Suara tersebut dihasilkan ketika angin tidak dapat keluar dan terperangkap di sekitar dada dan disertai dengan refleks untuk mengeluarkan angin tersebut melalui mulut.

2.2 *Mel-Frequency Cepstral Coefficients*

Mel-Frequency Cepstral Coefficients (MFCC) [5] adalah sebuah metode ekstraksi fitur suara yang merubah sinyal suara menjadi vektor. Keunggulan dari MFCC ialah sebagai berikut:

- Dapat menangkap karakter suara untuk mengetauhi pola pada suara tertentu.
- Memberikan *output* berupa vektor dengan ukuran data yang kecil namun tidak menghilangkan karakteristik dari suara yang diekstrak.
- Cara kerja algoritma MFCC mirip dengan cara kerja pendengaran manusia dalam memberikan persepsi terhadap suara yang didengarkan.

Alur ekstraksi fitur MFCC terdapat pada Gambar 2.1. Penjelasan untuk tiap-tiap tahap pada MFCC akan dibahas pada subbab 2.2.1-2.2.7.



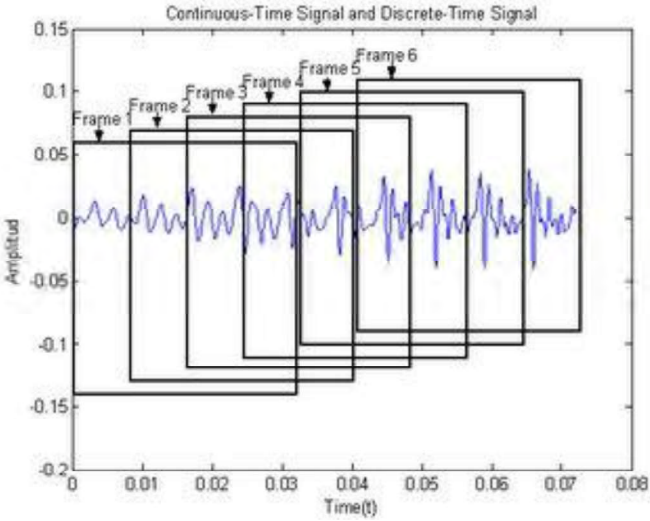
Gambar 2.1 Diagram Alur Ekstraksi Fitur MFCC

2.2.1 *Pre-Emphasis*

Pre-Emphasis [6] digunakan untuk mempertahankan frekuensi-frekuensi tinggi pada sebuah spektrum yang umumnya tereduksi saat proses produksi suara. Pada tahap ini, *file* audio yang telah dibaca sebagai data sinyal frekuensi dalam tipe data *array* sepanjang N detik akan ditransformasi menjadi sinyal frekuensi baru dengan persamaan (2.1) [7]. Persamaan *Pre-Emphasis* tertulis pada (2.1), di mana y_n adalah sinyal hasil *Pre-Emphasis* dan x_n adalah sinyal sebelum *Pre-Emphasis*. Misalkan nilai dari $\alpha = 0{,}97$, maka 97% dari setiap sampel dianggap berasal dari sampel sebelumnya.

$$y_n = x_n - \alpha \times x_{n-1}\ , 0.9 \leq \alpha \leq 1.0 \qquad \textbf{(2.1)}$$

2.2.2 *Frame Blocking*



Gambar 2.2 Ilustrasi *Frame Blocking* [8]

Frame Blocking [6] digunakan untuk membuat segmentasi pada sinyal frekuensi sebesar 20-30 milidetik (dikenal sebagai *frame*). Sinyal suara dibagi menjadi *N* sampel *frame* dan *frame* yang berdekatan dipisahkan oleh *M* (*M* < *N*) [5]. Pada umumnya nilai *N*=256 dan *M*=100. Panjang *frame*

akan menentukan keberhasilan analisa spektral. Ukuran *frame* harus sepanjang mungkin untuk menunjukkan resolusi frekuensi yang baik, namun juga harus cukup pendek untuk menunjukkan resolusi waktu yang baik [6].

Proses *frame blocking* ini akan dilakukan secara terus-menerus sampai seluruh sinyal dapat diproses. Pada umumnya, proses ini dilakukan secara *overlapping* pada setiap *frame*-nya. Panjangnya o*verlap* yang umum digunakan ialah 30% hingga 50% dari panjang *frame*. Dengan adanya *overlapping*, hilangnya karakteristik suara pada perbatasan perpotongan setiap *frame* dapat dihindari. Ilustrasi *frame blocking* dapat dilihat pada Gambar 2.2.

Sebagai contoh, jika audio yang telah melewati proses *Pre-Emphasis* memiliki *sample rate* sebesar 16000 Hz, maka segmentasi selama 20 milidetik dilakukan setiap 320 *frame*. Hasil tersebut diperoleh dari perkalian *sample rate* dengan durasi segmentasi. Kemudian, jika *overlap* yang dilakukan sebesar 50%, maka segmen pertama ialah dengan *frame* urutan 1-320, diikuti dengan segmen kedua dengan *frame* urutan 160-480 dan seterusnya [7].

2.2.3 *Windowing*

Windowing [6] digunakan untuk mengurangi terjadinya efek *aliasing* pada sinyal yang diproses setelah proses *framing*. *Aliasing* adalah sinyal baru yang frekuensinya berbeda dengan frekuensi sinyal aslinya akibat dari rendahnya *sampling rate* atau proses *frame blocking* yang menyebabkan sinyal terputus. Pada tahap ini, setiap sinyal frekuensi berupa spektrum suara ke-*n* yang telah diproses dari *Frame Blocking* dikalikan dengan nilai *Hamming Window* [7]. Nilai *Hamming Window* didapat dari fungsi *window*. Persamaan *windowing* tertulis pada (2.2), di mana N adalah jumlah sampel pada setiap *frame*, y_n adalah nilai sampel sinyal hasil *windowing*, x_n adalah nilai sampel dari *frame* sinyal ke-*n*, dan w_n adalah fungsi *window*. Fungsi *window* yang sering digunakan pada proses *windowing* ini ialah

Hamming Window (2.3) karena menghasilkan *sidelobe level* yang tidak terlalu tinggi dan *noise* yang tidak terlalu banyak.

$$y_n = x_n \times w_n \,, 0 \leq n \leq N-1 \quad \textbf{(2.2)}$$

$$w_n = 0.54 - 0.46cos\frac{2\pi n}{N-1} \,, 0 \leq n \leq N-1 \quad \textbf{(2.3)}$$

2.2.4 *Fast Fourier Transform*

Fast Fourier Transform [6] (FFT) digunakan mengubah tiap *frame* frekuensi pada sample N dari domain waktu menjadi domain frekuensi dan untuk menganalisa *spectral properties* berdasarkan sinyal yang dimasukkan. *Spectral properties* ini biasa dikenal sebagai *spectogram*. *Spectogram* berhubungan dengan waktu dan frekuensi. Jika resolusi waktu yang digunakan tinggi, maka frekuensi yang dihasilkan menjadi semakin rendah. Pada tahap ini, sinyal frekuensi dari proses *windowing* akan diproses dengan persamaan (2.4), di mana X_k adalah jumlah frekuensi k pada sinyal, k adalah frekuensi audio sebelum FFT (0 Hz sampai $N-1$ Hz), x_n adalah nilai sinyal pada waktu ke-n, dan N adalah jumlah waktu pada sampel [9].

$$X_k = \sum_{n=0}^{N-1} x_n \times e^{\frac{-2\pi kn}{N}} \,, n = 0 \leq n \leq N-1 \quad \textbf{(2.4)}$$

2.2.5 *Mel Frequency Warping*

Mel Frequency Warping [6] dilakukan untuk menghasilkan *magnitude spectrum* yang bagus dan memperkecil ukuran fitur terkait. *Mel Frequency Warping* dilakukan dengan menerapkan *filterbank* untuk mengetahui ukuran energi *frequency band* tertentu dalam sinyal suara [6]. *Filterbank* ini dapat diterapkan pada domain frekuensi maupun domain waktu, namun karena digunakan untuk MFCC, maka domain frekuensi yang diterapkan pada penelitian ini.

Pada tahap *Mel Frequency Warping*, sinyal frekuensi yang telah diproses pada tahap *Fast Fourier Transform* akan dirubah ke dalam frekuensi *Mel* dengan persamaan (2.5), di mana f adalah frekuensi sinyal. Kemudian, untuk mengetahui

besarnya energi pada frekuensi *Mel*, dilakukan perhitungan *filterbank* [7]. Persamaan *filterbank* terdapat pada (2.6), di mana N adalah jumlah *magnitude spektrum* (nilai yang sering digunakan ialah 40), $S[j]$ adalah *magnitude spectrum* yang pada frekuensi j, $H_i[j]$ adalah koefisien *filterbank* pada frekuensi j, dan M adalah jumlah *channel* pada *filterbank*.

$$Mel(f) = 2595 \log_{10}(1 + \frac{f}{700}) \quad \textbf{(2.5)}$$

$$y[i] = \sum_{i=1}^{N} S[j]H_i[j] \ , j = 1 \leq i \leq M \quad \textbf{(2.6)}$$

2.2.6 *Discrete Cosine Transform*

Discrete Cosine Transform [7] merupakan proses terakhir dari rangkaian proses MFCC yang digunakan untuk mengonversi *mel spectrum* menjadi domain yang mirip waktu (dikenal sebagai *quefrency*) sehingga spektral lokal dapat terepresentasi dengan baik. Hasil dari konversi ini disebut sebagai *Mel-Frequency Cepstrum Coefficient* yang berbentuk deretan vektor akustik. Pada tahap ini, frekuensi *Mel* ditransformasi dengan persamaan (2.7), di mana N adalah jumlah dari *Mel Frequency Warping* (nilai yang sering digunakan ialah 40), L adalah jumlah dari *Mel-Scale Cepstral Coefficients* (nilai yang sering digunakan ialah 12).

$$C_m = \sum_{k=1}^{N} \cos\left[m\left(k - \frac{1}{2}\right)\frac{\pi}{N}\right] E_k \ , m = 1,2, \dots, L \quad \textbf{(2.7)}$$

2.2.7 *Cepstral Liftering*

Cepstral Liftering [10] digunakan untuk memperhalus hasil dari seluruh proses MFCC agar lebih hasilnya lebih baik saat digunakan pada pattern matching. *Cepstral Liftering* diimplementasikan pada hasil *Discrete Cosine Transform* berupa *cepstral features* dengan menggunakan fungsi *window* pada persamaan (2.8), di mana L adalah jumlah *cepstral coefficients* dan n adalah indeks dari *cepstral coefficients*.

$$w_n = 1 + \frac{L}{2}\sin\left[\frac{\pi n}{L}\right] \quad \textbf{(2.8)}$$

2.3 Normalisasi

Normalisasi adalah sebuah teknik penskalaan data agar distribusi data sesuai dengan standar tertentu. Beberapa teknik normalisasi yang digunakan dalam penelitian ini ialah sebagai berikut.

2.3.1 *Standard Deviation Normalization*

Standard Deviation Normalization [1] adalah metode untuk menormalkan data berdasarkan besarnya nilai standar deviasi dari sebuah kumpulan data. Cara kerja metode ini ialah nilai data yang ke sekian dikurangi dengan nilai rata-rata dari kumpulan data terkait kemudian dibagi dengan nilai standar deviasi dari data terkait.

Pada Tugas Akhir ini, *Standard Deviation Normalization* digunakan untuk menormalisasi hasil ekstraksi fitur sebuah rekaman suara tangis bayi dengan persamaan (2.9), di mana x_i adalah data ekstraksi fitur ke-*i* yang telah dinormalisasi, $data_i$ adalah data ekstraksi fitur ke-*i* sebelum dinormalisasi, $\mu(data)$ adalah rata-rata data ekstraksi fitur, dan $\sigma(data)$ adalah standar deviasi dari data ekstraksi fitur.

$$x_i = \frac{data_i - \mu(data)}{\sigma(data)} \quad \textbf{(2.9)}$$

2.3.2 *Min-Max Normalization*

Min-Max Normalization [1] (2.10) adalah metode untuk menormalkan data berdasarkan besarnya nilai maksimum dikurangi dengan nilai minimum dari sebuah kumpulan data. Cara kerja metode ini ialah nilai data yang ke sekian dikurangi dengan nilai minimum dari kumpulan data terkait kemudian dibagi dengan nilai maksimum yang dikurangi nilai minimum dari data terkait.

Pada Tugas Akhir ini, *Min-Max Normalization* digunakan untuk menormalisasi hasil ekstraksi fitur sebuah

rekaman suara tangis bayi dengan persamaan (2.10), di mana x_i adalah data ekstraksi fitur ke-*i* yang telah dinormalisasi, $data_i$ adalah data ekstraksi fitur ke-*i* sebelum dinormalisasi, $\min(data)$ adalah nilai minimal pada data ekstraksi fitur, dan $\max(data)$ adalah nilai maksimal dari data ekstraksi fitur.

$$x_i = \frac{data_i - \min(data)}{\max(data) - \min(data)} \quad \textbf{(2.10)}$$

2.3.3 *Decimal Normalization*

Decimal Normalization [1] (2.11) adalah metode untuk menormalkan data berdasarkan besarnya nilai 10^j (j adalah bilangan bulat). Cara kerja metode ini ialah data nilai data yang ke sekian dibagi dengan 10^j.

Pada Tugas Akhir ini, *Min-Max Normalization* digunakan untuk menormalisasi hasil ekstraksi fitur sebuah rekaman suara tangis bayi dengan persamaan (2.11), di mana x_i adalah data ekstraksi fitur ke-*i* yang telah dinormalisasi, $data_i$ adalah data ekstraksi fitur ke-*i* sebelum dinormalisasi, dan j adalah bilangan bulat (nilai yang dipakai adalah 3).

$$x_i = \frac{data_i}{10^j} \quad \textbf{(2.11)}$$

2.4 *K-Nearest Neighbor*

K-Nearest Neighbor (KNN) [11] adalah algoritma sederhana yang menyimpan semua kasus yang tersedia dan mengklasifikasikan kasus baru berdasarkan ukuran kesamaan menggunakan fungsi jarak. KNN telah digunakan dalam estimasi statistik dan pengenalan pola pada awal 1970-an sebagai teknik non-parametrik.

Cara kerja algoritma ini ialah sebuah data uji diklasifikasikan menjadi suatu kelas tertentu berdasarkan jumlah tetangga terdekat dengan menggunakan perhitungan jarak *Euclidian*. Saat perhitungan, jarak antar data uji dengan seluruh data latih beserta label kelas disimpan pada vektor, kemudian diurutkan berdasarkan jarak terpendek. Data uji akan

masuk ke kelas tertentu berdasarkan jumlah label terbanyak dari nilai *k* yang ditentukan.

Pada Tugas Akhir ini, *K-nearest Neighbor* digunakan untuk mengklasifikasikan hasil ekstraksi fitur suara tangis bayi dengan menggunakan persamaan (2.12), di mana d_E adalah jarak *Euclidian* dari suatu data uji dengan data latih pada satu kelas tangis bayi, n adalah jumlah fitur yang diekstrak dari suara tangis bayi, x_i adalah fitur ke-*i* dari data uji, dan y_i adalah fitur ke-*i* dari data latih.

$$d_E = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2} \tag{2.12}$$

2.5 *Naive Bayes*

Naive Bayes [12] merupakan sebuah algoritma klasifikasi yang menerapkan metode probabilitas dan statistik yang ditemukan oleh Thomas Bayes. Cara kerja algoritma ini ialah memprediksi peluang data baru berdasarkan berdasarkan hasil pembelajaran dari data yang sebelumnya sudah ada. Tahap-tahap pada algoritma ini sebagai berikut:

- Masukkan data latih, kemudian hitung nilai *mean* (2.13) dan standar deviasi (2.14) yang merupakan parameter numerik dari data tersebut.

$$\mu = \frac{1}{n} \sum_{i=1}^{n} x_i \tag{2.13}$$

$$\sigma = \sqrt{\frac{1}{n-1} \sum_{i=1}^{n} (x_i - \mu)^2} \tag{2.14}$$

Keterangan:
μ = nilai *mean* data
σ = standar deviasi data
n = jumlah data
x_i = nilai data yang ke - i

- Setelah mendapatkan *mean*, dan standar deviasi, dari data latih, tentukan nilai probabilistik data uji dengan menghitung jumlah data yang sesuai pada kategori yang sama dibagi dengan jumlah data pada kategori tersebut. Persamaan yang digunakan ialah *Gaussian Naive Bayes* (2.15), di mana x adalah data uji dan c adalah kelas dari suatu data. Cari nilai probabilitas terbesar, maka ia akan masuk pada kelas tersebut.

$$P(x|c) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \quad \textbf{(2.15)}$$

2.6 *Neural Network*

Neural Network [13] merupakan sebuah metode klasifikasi yang mengadopsi cara berpikir manusia pada pemrosesan sinyal elemen yang diterima, toleransi terhadap kesalahan, dan *paralel processing*. Struktur *Neural Network* terdiri atas banyak *node* yang terhubung secara langsung melalui sebuah jaringan. Jaringan antar satu *node* ke *node* yang lainnya digunakan untuk propagasi aktivasi dari *node* pertama ke *node* selanjutnya. Setiap *node* memiliki bobot numerik yang menentukan kekuatan maupun penanda dari sebuah jaringan. Kumpulan dari beberapa *node* pada satu dimensi disebut *layer*.

Pada umumnya, terdapat tiga layer dalam *Neural Network*, yaitu:

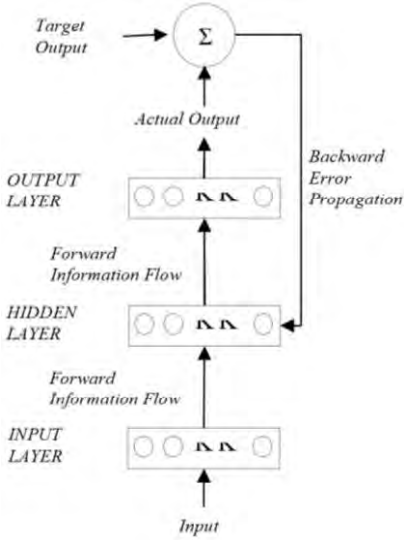
- *Input layer* yang berfungsi untuk menampung data masukan. Banyaknya *node* pada *input layer* tergantung pada banyaknya parameter pada data masukan tersebut.
- *Hidden layer* yang berfungsi untuk melakukan proses pembelajaran pada *Neural Network* dengan menggunakan fungsi aktivasi dan algoritma pembelajaran tertentu. Jumlah hidden layer boleh lebih dari satu, dan jumlah *node* pada masing-masing *hidden layer* dapat diatur sesuai dengan kebutuhan.
- *Output layer* yang berfungsi untuk mengeluarkan hasil pemrosesan data. Untuk klasifikasi *multi-class*, jumlah

node mengikuti jumlah kelas yang dilatih maupun diklasifikasi.

Pada *supervised learning*, algoritma yang digunakan *Neural Network* untuk proses pembelajaran ialah *backpropagation* dengan *backward error (gradien) propagation* [14]. Fungsi aktivasi yang digunakan dalam *backpropagation* ialah fungsi *sigmoid* atau dikenal juga sebagai fungsi *logistic* yang terdapat pada persamaan (2.16).

$$f_1(x) = \frac{1}{1 + e^{-x}} \tag{2.16}$$

Cara kerja *Neural Network* diawali dengan inisialisasi nilai bobot dan bias. Pemilihan metode inisialisasi menentukan kecepatan *Neural Network* untuk mencapai konvergen. Ilustrasi *backpropagation* dapat dilihat pada Gambar 2.3.



Gambar 2.3 Ilustrasi *Backpropagation Neural Network*

Pada pelatihan *Neural Network* dengan *backpropagation*, terdapat tiga tahap, yaitu pelatihan data masukan yang bersifat umpan maju, perhitungan *error*, dan penyesuaian nilai bobot. Secara umum, cara kerja *Neural*

Network dengan *backpropagation* terdiri dari beberapa langkah. Pertama, pola data masukan dan target dimasukkan, kemudian pola data masukan tersebut akan berubah sesuai dengan propagasi pola tersebut ke lapisan-lapisan berikutnya hingga menghasilkan data keluaran. Data keluaran tersebut akan dibandingkan dengan target. Jika hasil perbandingan sama dengan target, maka pembelajaran akan berhenti. Namun, jika berbeda, *Neural Network* akan mengubah nilai bobot yang ada pada hubungan antar *neuron* dengan aturan tertentu agar nilai data keluaran lebih mendekati nilai target.

Proses perubahan nilai bobot dilakukan dengan mempropagasikan kembali nilai korelasi *error* data keluaran *Neural Network* ke lapisan-lapisan sebelumnya (*backpropagation*). Kemudian, pola dari lapisan data masukan akan diproses lagi untuk mengubah nilai bobot hingga memperoleh data keluaran baru. Proses ini dilakukan berulang-ulang hingga nilai yang sama atau minimal sesuai dengan *error* yang diinginkan diperoleh. Proses perubahan nilai bobot ini yang disebut pembelajaran.

Seiring dengan berkembangnya zaman, algoritma *backpropagation* dioptimasi agar lebih cepat saat melakukan pembelajaran. Algoritma tersebut adalah *resilient backpropagation*. Cara kerja algoritma ini ialah menggunakan tanda turunan untuk arah perbaikan bobot-bobot. Perubahan nilai pada setiap bobot ditentukan oleh suatu faktor yang diatur pada parameter yang disebut *delt_inc* dan *delt_dec*. Jika gradien fungsi *eror* berubah tanda dari satu iterasi ke iterasi berikutnya, maka besaran bobot akan berkurang sebesar *delt_dec*. Sebaliknya, jika gradien *error* tidak berubah tanda dari satu iterasi ke iterasi lainnya, maka bobot akan berkurang sebesar *delt_inc*. Jika gradien *error* sama dengan 0 maka perubahan sama dengan perubahan bobot sebelumnya. Di awal iterasi, berubahnya nilai bobot diinisialisasikan dengan parameter *delta0*. Nilai perubahan tidak boleh melebihi batas maksimal dari parameter *deltamax*. Jika perubahan bobot melebihi atas

maksimal tersebut, maka perubahan bobot ditentukan sebesar nilai dari parameter *deltamax*.

2.7 *Support Vector Machine*

Support Vector Machine [15] merupakan sebuah algoritma klasifikasi *supervised* yang membuat suatu model dengan memisahkan kategori antara satu data dengan yang lain yang membuat data tersebut menjadi *non-probabilistic binary linear classifier*. Pada dasarnya, klasifikasi dengan *Support Vector Machine* digunakan pada data yang terdiri dari dua kelas. Namun, penelitian lebih lanjut mengembangkan *Support Vector Machine* agar dapat digunakan untuk menyelesaikan masalah klasifikasi *multi-class*. Salah satu metode tersebut ialah *One-againts-one*. Metode tersebut dibangun sebanyak $\frac{k(k-1)}{2}$ buah model klasifikasi biner dengan k adalah jumlah kelas pada suatu data. Model tersebut akan dilatih pada data dari dua kelas. Untuk melatih data dari kelas ke-i dan kelas ke-j dilakukan pencarian solusi untuk optimasi konstrain pada persamaan (2.17).

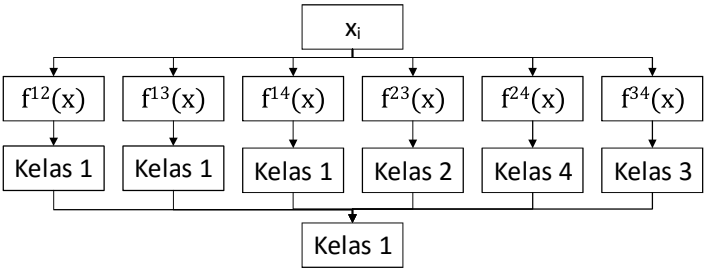
$$
\begin{aligned}
& min_{w^{ij},b^{ij},\xi^{ij}} \frac{1}{2}(w^{ij})^T w^{ij} + C\sum_t \xi_t^{ij} \\
& s.t (w^{ij})^T \phi(x_t) + b^{ij} \geq 1 - \xi_t^{ij} \rightarrow y_t = i, \\
& (w^{ij})^T \phi(x_t) + b^{ij} \geq -1 + \xi_t^{ij} \rightarrow y_t = j, \\
& \xi_t^{ij} \geq 0
\end{aligned}
\tag{2.17}
$$

Terdapat beberapa metode untuk melakukan pengujian setelah $\frac{k(k-1)}{2}$ model klasifikasi selesai dibangun, salah satunya adalah metode voting. Saat data x dimasukkan ke dalam fungsi pelatihan $f(x) = \left(w^{ij}\right)^T \phi(x) + b$ dan hasilnya menyatakan x adalah kelas i, maka suara untuk kelas i bertambah satu. Kelas dari data x akan ditentukan berdasarkan jumlah suara terbanyak. Jika terdapat dua buah kelas yang jumlah suaranya sama, maka indeks dari kelas yang lebih kecil dinyatakan sebagai kelas dari

data. Dengan pendekatan $\frac{k(k-1)}{2}$ buah permasalahan *quadratic programming* ini, masing-masing memiliki $\frac{2n}{k}$ variabel dengan n adalah jumlah data pelatihan. Sebagai contoh, ada sebuah permasalah klasifikasi dengan 4 kelas, maka terdapat 6 buah *Support Vector Machine* biner yang akan digunakan. Ilustrasi dapat dilihat pada Tabel 2.1 dan Gambar 2.4.

Tabel 2.1 Contoh SVM 6 Biner dengan Metode *One-againts-one*

$y_i = 1$	$y_i = -1$	Hipotesis
Kelas 1	Kelas 2	$f^{12}(x) = w^{12}x + b^{12}$
Kelas 1	Kelas 3	$f^{13}(x) = w^{13}x + b^{13}$
Kelas 1	Kelas 4	$f^{14}(x) = w^{14}x + b^{14}$
Kelas 2	Kelas 3	$f^{23}(x) = w^{23}x + b^{23}$
Kelas 2	Kelas 4	$f^{24}(x) = w^{24}x + b^{24}$
Kelas 3	Kelas 4	$f^{34}(x) = w^{34}x + b^{34}$



Gambar 2.4 Contoh Klasifikasi dengan Metode *One-againts-one*

2.8 tuneR

Package tuneR [16] adalah kumpulan pustaka pengolahan audio gratis yang dapat dipasang dan terintegrasi dengan bahasa pemrograman R. Beberapa fungsi yang ada pada *package* tuneR ialah *tools* untuk menganalisa musik, ekstraksi fitur MFCC, *handling* dokumen dengan format ".wav"e, membaca dokumen dengan format mp3, dan beberapa fungsi lain yang di-*port* dari *package* rastamat Matlab. Pada Tugas

Akhir ini, *package* tuneR dipakai untuk membaca *file* audio dengan format .wav dan ekstraksi fitur MFCC.

2.9 caret

Package caret [17] adalah kumpulan pustaka pengolahan data gratis berisi fungsi *misc* untuk pelatihan dan *plotting* klasifikasi dan model regresi yang dapat dipasang dan terintegrasi dengan bahasa pemrograman R. Pada Tugas Akhir ini, *package* caret dipakai untuk menghitung akurasi program saat pengujian data uji secara paralel.

2.10 e1071

Package e1071 [18] adalah kumpulan pustaka pengolahan data gratis berisi berbagai fungsi untuk klasifikasi maupun *clustering* yang dapat dipasang dan terintegrasi dengan bahasa pemrograman R. Pada Tugas Akhir ini, *package* e1071 dipakai untuk mengklasifikasikan data tangis bayi dengan metode *Naive Bayes* dan *Support Vector Machine*.

2.11 neuralnet

Package neuralnet [19] adalah kumpulan pustaka pengolahan data gratis berisi berbagai fungsi untuk melatih *Neural Network* dengan *backpropagation*, *resilient backpropagation* dengan atau tanpa *weight backtracking*, dan lain-lain yang dapat dipasang dan terintegrasi dengan bahasa pemrograman R. Pada Tugas Akhir ini, *package* neuralnet dipakai untuk mengklasifikasikan data tangis bayi dengan metode *Resilient Backpropagation Neural Network* dengan dua *hidden layer* yang berjumlah 7 dan 5 *node* pada masing-masing *hidden layer*.

2.12 Penelitian terkait Deteksi Suara Tangis Bayi

Pada penelitian sebelumnya, terdapat beberapa penelitian terkait deteksi suara tangis bayi, yaitu penelitian dengan judul "Identifikasi Arti Tangis Bayi Versi *Dunstan Baby Language* Menggunakan Jarak Terpendek dari Jarak *Mahalanobis*" [2],

"*Application of Neuro-Fuzy Approaches to Recognition and Classification of Infant Cry*" [1], dan "*Normal and Hypoacoustic Infant Cry Signal Classification Using Time-Frequency Analysis and General Regression Neural Network*" [3]. Secara garis besar, penelitian [2] dan [1] membahas tentang bagaimana menentukan arti tangis bayi berdasarkan *Dunstan Baby Language*, sedangkan penelitian [3] membahas bagaimana membedakan bayi yang normal dengan bayi yang tuli berdasarkan frekuensi suara tangis bayi yang bersangkutan.

Pada penelitian [2], penelitian ditujukan pada suara tangis bayi berusia 0-3 bulan versi *Dunstan Baby Language* menggunakan pemodelan *codebook* dengan *K-means clustering* sebagai metode untuk pencocokan fitur dan ekstraksi fitur *Mel-Frequency Cepstrum Coefficients* untuk ekstraksi ciri suara tangis bayi. Identifikasi suara tangis bayi dilakukan dengan menggunakan jarak terpendek pada jarak *Mahalanobis*. Pengolahan data pada penelitian [2] menggunakan kombinasi panjang *frame* suara tangis bayi, yaitu 25 milidetik, 40 milidetik, dan 60 milidetik, *frame overlap* 0%, 40% dan 60% dengan jumlah *codewords* (jumlah *cluster*) 1 sampai 29. Akurasi terbaik untuk pengenalan lima tipe tangis bayi dengan jarak *Mahalanobis* dapat dicapai sebesar 83% ketika panjang *frame* 275, *overlap frame* 0.25, dan k=17. Suara tangis "Heh" paling mudah dikenali, sementara suara "Owh" yang paling sering salah dan dikenali sebagai "Neh" atau "Eairh".

Pada penelitian [1], penelitian ditujukan pada suara tangis bayi versi *Dunstan Baby Language* dengan menggunakan metode ekstraksi fitur *Perceptual Linear Prediction*, *Neuro-Fuzzy* sebagai pengenalan suara, dan *K-Nearest Neighbor* sebagai *classifier*. Hasil dari penelitian ini ialah *Neuro-Fuzzy* menghasilkan akurasi terbaik sebagai metode untuk pengenalan suara jika dibandingkan dengan beberapa algoritma pengenalan suara lainnya.

Pada penelitian [3], penelitian dilakukan untuk membedakan bayi yang normal dengan bayi yang tuli

berdasarkan suara tangis bayi tersebut. Metode yang digunakan ialah *Short-time Fourier Transform* (STFT) untuk menganalisis sinyal tangis bayi berdasarkan *time-frequency* dan *General Regression Neural Network* (GRNN) sebagai *classifer* untuk membedakan tangis bayi normal atau tuli. Untuk membuktikan kehandalan fitur yang diusulkan, dua model *neural network* seperti *Multi Layer Perceptron* (MLP) dan *Time-Delay Neural Network* (TDNN) dilatih dengan algoritma *conjugate gradient* digunakan sebagai *classifier*. Hasil percobaan pada penelitian [3] menunjukkan bahwa algoritma GRNN memberikan akurasi yang lebih baik jika dibandingkan dengan MLP dan TDNN dan metode yang diusulkan dapat mengklasifikasikan suara tangis bayi normal dan suara tangis bayi tuli dengan baik.

[Halaman ini sengaja dikosongkan]

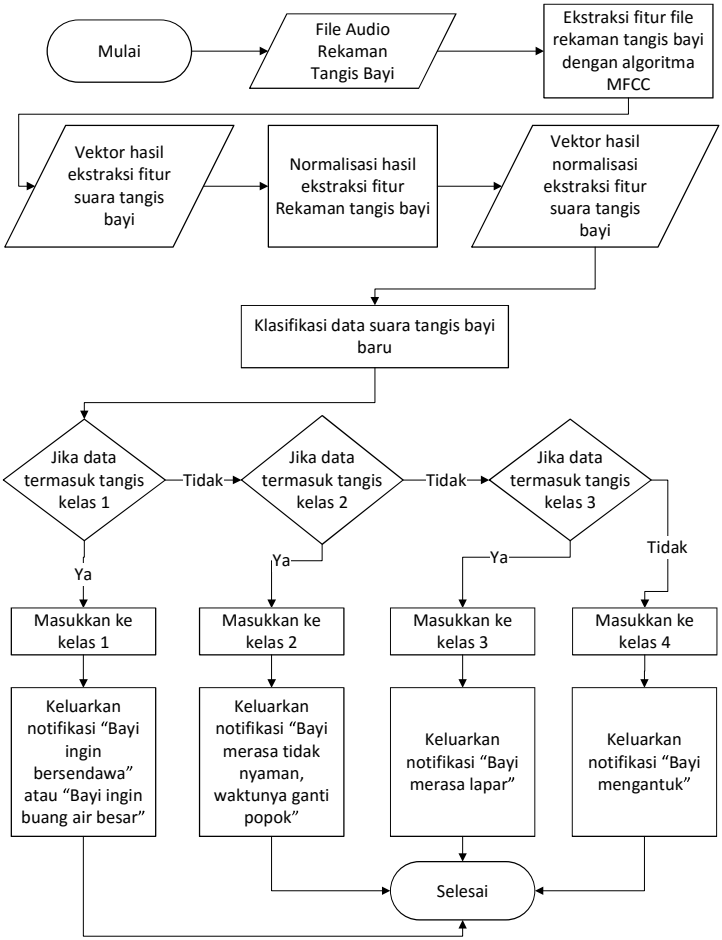
BAB III
PERANCANGAN SISTEM

Pada bab ini akan dijelaskan mengenai perancangan sistem dari perangkat lunak yang akan dibuat. Perancangan sistem pada Tugas Akhir ini meliputi perancangan data, perancangan uji coba, dan perancangan sistem. Selain itu akan dijelaskan juga desain metode secara umum pada sistem.

3.1 Desain Metode Secara Umum

Pada Tugas Akhir ini akan dibangun suatu sistem yang dapat melakukan klasifikasi terhadap rekaman suara tangis bayi pada empat kelas yaitu kelas "Eairh/Eh" (bayi ingin bersendawa, kentut, atau buang air besar), kelas "Heh" (bayi merasa tidak nyawan, waktunya ganti popok), kelas "Neh" (bayi merasa lapar), atau kelas "Owh" (bayi mulai mengantuk), di mana rekaman suara tangis bayi yang dipakai memiliki durasi maksimal sepuluh detik. Pada tahap ini dilakukan beberapa proses meliputi ekstraksi fitur, normalisasi hasil ekstraksi fitur, dan tahap klasifikasi berdasarkan fitur yang telah didapatkan. Kemudian, hasil akhirnya adalah hasil kalisifikasi beserta perhitungan akurasi dari data *testing*. Diagram alur dari proses ini ditunjukan pada Gambar 3.1. Dalam Tugas Akhir ini, terdapat empat *classifier* yaitu *K-nearest Neighbor*, *Naive Bayes*, *Neural Network*, dan *Support Vector Machine*.

Sebelum klasifikasi dilakukan, suara rekaman bayi yang ada terlebih dahulu di ekstraksi sehingga menjadi fitur-fitur yang nantinya akan diolah. Fitur yang diolah ialah frekuensi Mel dari proses ekstraksi fitur MFCC. Fitur yang akan diolah akan dinormalisasi terlebih dahulu untuk menghindari rentang nilai yang terlalu jauh. Setelah dilakukan normalisasi, data akan diklasifikasi. Pada Tugas Akhir ini digunakan klasifikasi *K-nearest Neighbor* dengan menggunakan nilai konstanta $1 \leq k \leq 5$, *Naive Bayes*, *Resilient Backpropagation Neural Network*, dan *Support Vector Machine*.



Gambar 3.1 Diagram Proses Klasifikasi Suara Tangis Bayi

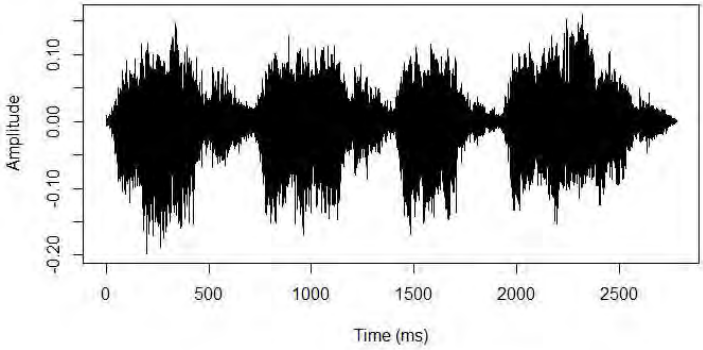
3.2 Perancangan Data

Pada sub bab ini akan dijelaskan mengenai perancangan data yang dibutuhkan untuk membangun perangkat lunak deteksi suara tangis bayi dengan metode klasifikasi *K-nearest Neighbor*. Perancangan data meliputi data masukan, data proses

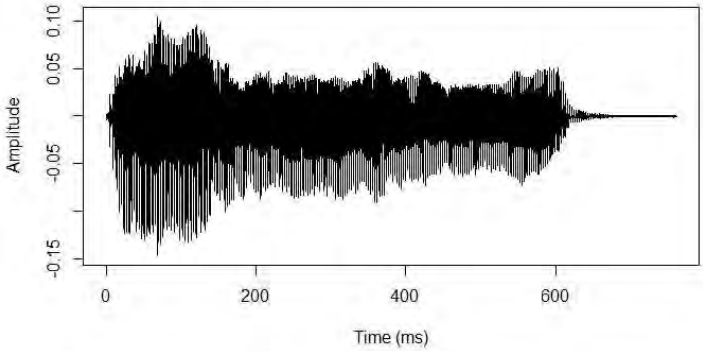
berupa data-data yang dibutuhkan dan dihasilkan selama menjalankan proses eksekusi perangkat lunak, dan data keluaran.

3.2.1 Data Masukan

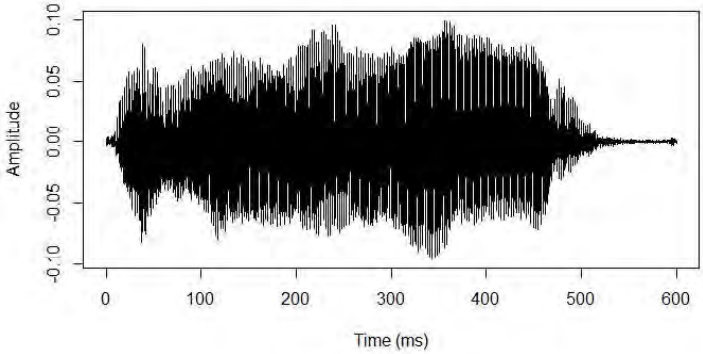
Data masukan adalah data yang akan diolah oleh sistem untuk mendapatkan hasil keluaran yang sudah ditentukan sistem. Pada aplikasi ini, *Dataset* yang digunakan diekstraksi dari kumpulan rekaman suara tangis bayi pada usia 0-3 bulan. Rekaman suara tangis bayi ini didapat dari penelitian [1] berupa *file* video tangis bayi versi *Dunstan Baby Language* dengan nama *file* setiap jenis tangis bayi (Eairh/Eh/Heh/Neh/Owh) yang kemudian dipotong dan dikonversi menjadi *file* audio dengan format ".wav", *sample rate* 11205 Hz, channel *mono*, dan durasi maksimal sepuluh detik. Contoh hasil *plot* masing-masing jenis tangis bayi terdapat pada Gambar 3.2 sampai Gambar 3.6. Setiap audio akan diproses melalui ekstraksi fitur dengan menggunakan metode MFCC yang terdapat pada *package* tuneR berbasis bahasa pemrograman R. Ekstraksi fitur dilakukan untuk membedakan masing-masing jenis tangis bayi yang akan dijadikan acuan untuk memprediksi jenis tangisan dari rekaman suara tangis baru.



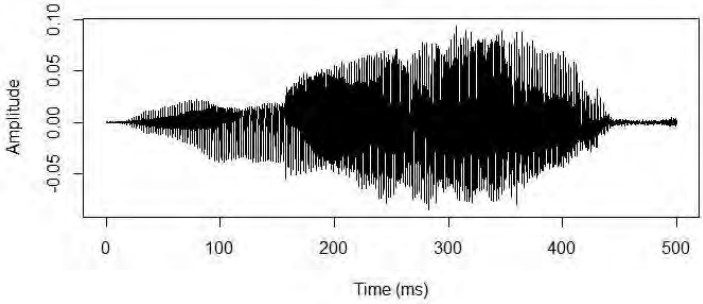
Gambar 3.2 Hasil *Plot* Audio Salah Satu Tangis Bayi Jenis "Eairh"



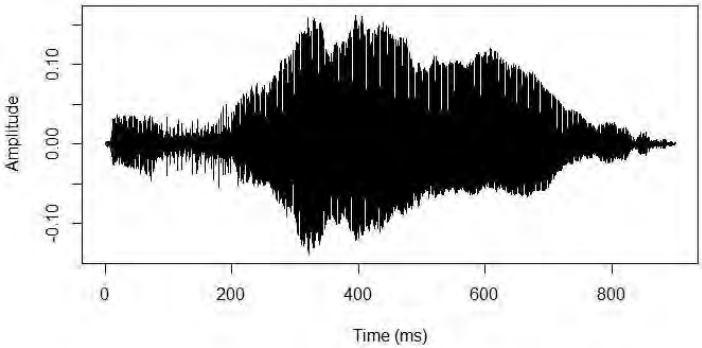
Gambar 3.3 Hasil *Plot* Audio Salah Satu Tangis Bayi Jenis "Eh"



Gambar 3.4 Hasil *Plot* Audio Salah Satu Tangis Bayi Jenis "Heh"



Gambar 3.5 Hasil *Plot* Audio Salah Satu Tangis Bayi Jenis "Neh"



Gambar 3.6 Hasil *Plot* Audio Salah Satu Tangis Bayi Jenis "Owh"

Data masukan terdiri dari 139 suara rekaman tangis bayi yang dibagi menjadi empat kelas, yaitu kelas “Eairh/Eh”, kelas “Heh”, kelas “Neh”, dan kelas “Owh”. Jumlah dataset yang digunakan di setiap kelas dijelaskan pada Tabel 3.1.

Tabel 3.1 Tabel Jumlah Dataset Suara Tangis Bayi

Jenis Tangis	Kelas (angka)	Jumlah rekaman
Eairh/Eh	1	46
Heh	2	23
Neh	3	35
Owh	4	35

3.2.2 Data Proses

Data proses adalah proses pengolahan data dari data masukan. Sebelum keluarnya suatu *output* dari data masukan berupa audio rekaman tangis bayi, maka data masukan tersebut harus diproses melalui tahap ekstraksi fitur, normalisasi, dan klasifikasi hingga akhir nya keluar sebuah *output*.

3.2.2.1 Data pada Ekstraksi Fitur dan Normalisasi Ekstraksi Fitur

Sebelum dilakukan klasifikasi terhadap suatu rekaman suara tangis bayi, proses ekstraksi fitur dilakukan terlebih dahulu pada tiap fitur yang ada, baik data latih mapun data uji.

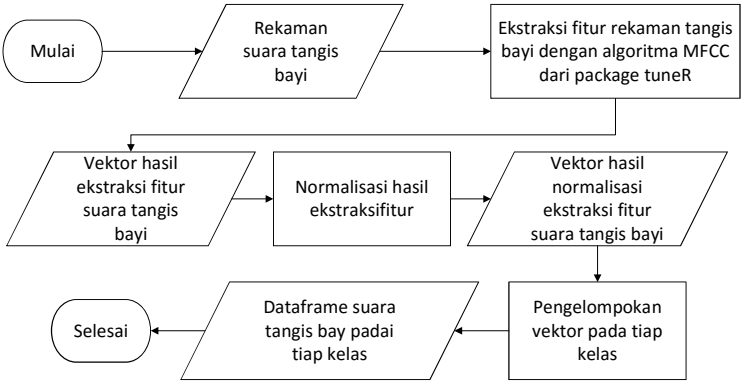
Fitur yang diambil dari tiap rekaman berasal dari *speech signal* suara tangis bayi. Ekstraksi fitur bertujuan untuk mendapatkan informasi-informasi penting pada sebuah rekaman suara tangis bayi, sehingga dapat membedakan suara tangis yang satu dengan suara tangis yang lain.

Pada Tugas Akhir ini, ekstraksi fitur MFCC dilakukan dengan menggunakan *package* tuneR. Dari data masukan berupa suara tangis bayi, ada beberapa parameter yang diperhatikan saat proses ekstraksi fitur dengan algoritma MFCC berdasarkan *package* tuneR, yaitu:

- *Sample rate* (*sr*) dari rekaman suara tangis bayi sebesar 11205 Hz.
- *Window time* (*wintime*) selama 0,02-0,08 detik.
- Jumlah spektral (*nbands*) yang di-*warp* sebanyak 40.
- Jarak antar *window* (*hoptime*). Jika durasi rekaman tangis bayi kurang dari 2 detik, maka nilai *hoptime* sebesar durasi tangis bayi dikurangi 0,1 detik. Jika durasi rekaman bayi lebih dari 2 detik, maka nilai *hoptime* sebesar durasi tangis bayi dikurangi 1 detik.
- Pre-emphasize (*preemph*) sebesar 0,97.
- *Sumpower* = TRUE.
- Nilai eksponensial untuk *liftering* (*lifterexp*) sebesar 0,6.
- Lebar dari *spectral band* (*bwidth*) sebesar 10.
- *Usecmp* = TRUE.
- Jumlah kolom pada vektor hasil ekstraksi fitur (*numcep*) yang dihasilkan sebanyak 10 untuk setiap *hoptime*.

Total fitur yang dihasilkan pada tahap ekstraksi fitur adalah 20 fitur Frekuensi Mel. 10 fitur pertama dihasilkan berdasarkan durasi tangis bayi dikurangi variabel *hoptime*, kemudian 10 fitur berikutnya dihasilkan dari sisa durasi tangis bayi yang sebelumnya telah dipotong variabel *hoptime*. Fitur tersebut akan dimasukkan ke dalam sebuah variabel dengan tipe data vektor yang terdiri dari 1 baris dan 20 kolom yang kemudian

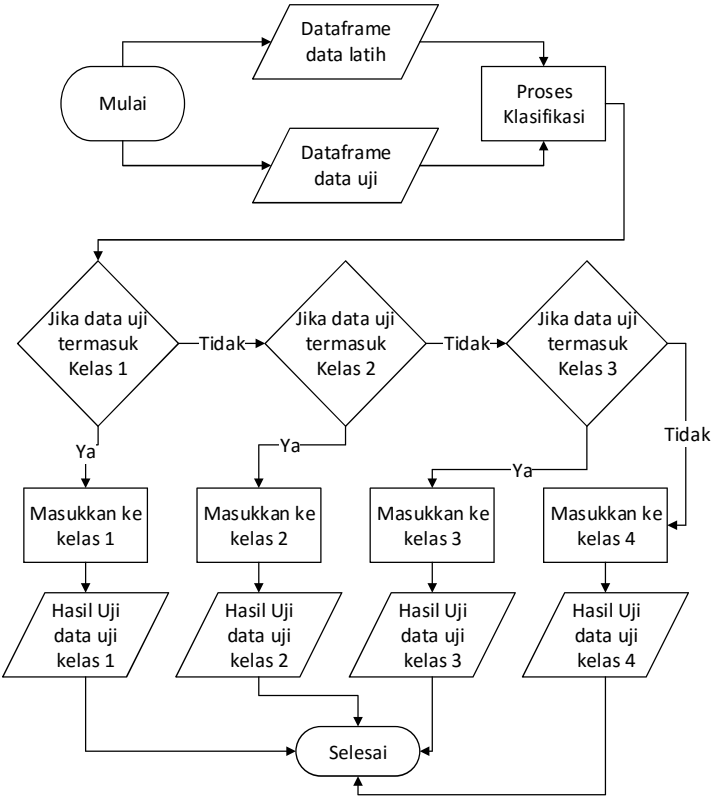
akan dinormalisasi. Setelah hasil ekstraksi fitur dinormalisasi, data tersebut akan dimasukan pada variabel *dataframe* masing-masing kelas dengan jumlah baris sesuai jumlah data tangis bayi pada tiap kelas dan 21 kolom yang terdiri dari 20 kolom hasil normalisasi ekstraksi fitur dan 1 kolom sebagai penanda kelas data terkait. Jumlah *dataframe* yang dihasilkan pada proses ini sebanyak empat. Alur dari proses ekstraksi fitur dijelaskan melalui Gambar 3.7.



Gambar 3.7 Alur Ekstraksi Fitur dan Normalisasi

3.2.2.2 Klasifikasi Data

Dataset berupa *dataframe* yang telah melalui tahap ekstraksi fitur dan normalisasi ekstraksi fitur selanjutnya akan diolah untuk diklasifikasi dengan *K-nearest Neighbor*, *Naive Bayes*, *Neural Network*, dan *Support Vector Machine*. Pada *K-nearest Neighbor*, variasi nilai *k* sebesar 1 sampai 5. Pada *Resilient Backpropagation Neural Network*, fungsi aktivasi yang digunakan adalah fungsi *sigmoid*, *hidden layer* yang digunakan sebanyak dua dengan 7 dan 5 node pada masing-masing layer dan *learning rate* yang digunakan sebesar 0,02. Pada *Support Vector Machine*, *kernel* yang digunakan ialah *polynomial* dan metode klasifikasi *One-againts-one*. Alur klasifikasi akan dijelaskan melalui Gambar 3.8.



Gambar 3.8 Alur Klasifikasi Data

3.2.3 Data Keluaran

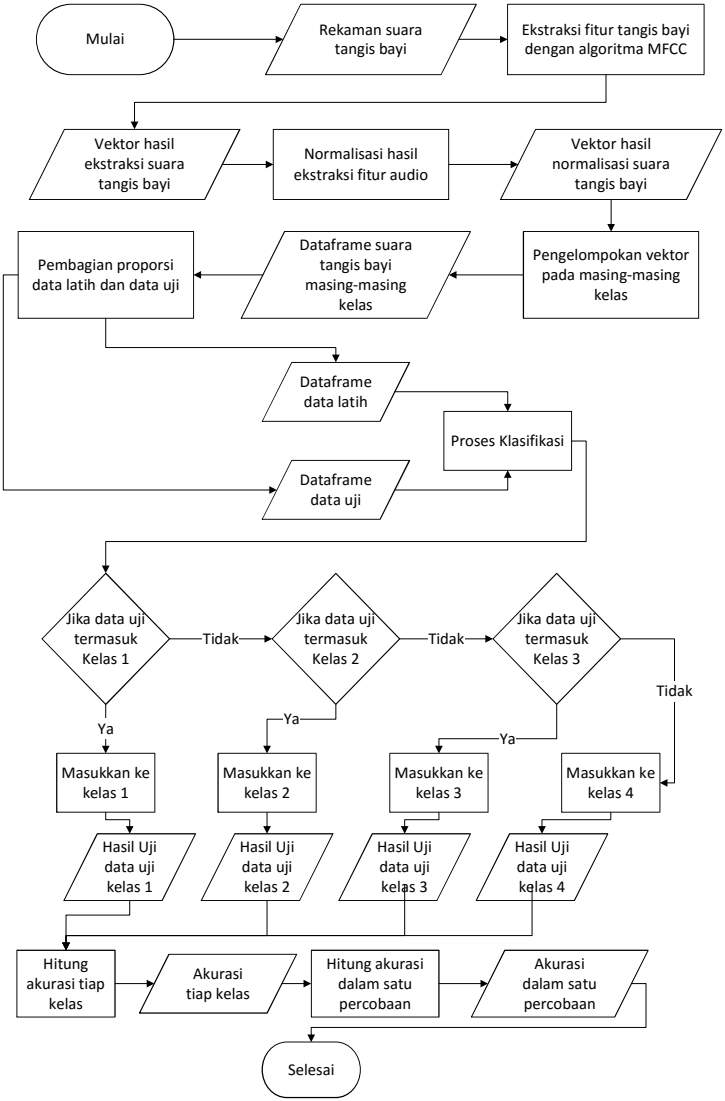
Data keluaran adalah data yang dihasilkan dari pemrosesan data masukan. Pada Tugas Akhir ini, akan ada dua jenis data keluaran, yaitu data keluaran masing-masing pemilihan metode dan data keluaran dari aplikasi yang telah dibangun. Data keluaran yang dihasilkan saat pemilihan metode terbaik berupa akurasi tiap kelas dan akurasi rata-rata dalam satu percobaan. Data keluaran pada aplikasi yang telah dibangun berupa hasil ekstraksi fitur yang sudah dinormalisasi

berupa teks dan hasil klasifikasi jenis suara tangis bayi berupa teks.

3.3 Perancangan Uji Coba Pemilihan Metode Terbaik

Pada sub bab ini akan dijelaskan tentang perancangan uji coba pemilihan metode terbaik pada pembuatan "Aplikasi Pendeteksi Suara Tangis Bayi". Diharapkan pada tahap ini ditemukan akurasi tertinggi dengan kombinasi metode yang diterapkan. Metode dengan hasil terbaik akan dijadikan sebagai metode yang ada pada aplikasi yang akan dibagun.

Dalam percancangan uji coba ini, data masukan berupa rekaman suara tangis bayi dengan format ".wav" diproses melalui ekstraksi fitur MFCC dari *package* tuneR dengan kombinasi parameter *wintime* 0,02-0,08 detik dengan tipe data vektor. Setelah itu, hasil ekstraksi fitur dari data masukan akan dinormalisasi dengan kombinasi metode normalisasi, yaitu *Decimal Normalization*, *Min-Max Normalization*, dan *Standard Deviation Normalization*. Data hasil ekstraksi fitur berupa vektor yang telah dinormalisasi akan dimasukkan pada *dataframe* masing-masing kelas. Sebelum dilakukan klasifikasi, dataset berupa *dataframe* yang berjumlah 139 data akan dipisah sebanyak 70%-90% data latih dan 10%-30% data uji pada masing-masing kelas. Proses klasifikasi *K-nearest Neighbor* dilakukan dengan menggunakan kombinasi nilai *k* sebesar 1-5. Setelah tahap klasifikasi selesai, akurasi akan dihitung untuk mengetahui seberapa besar keakuratan metode *K-nearest Neighbor* untuk melakukan klasifikasi pada suara tangis bayi ini. Akurasi pada uji coba ini ada 2 macam, yaitu akurasi tiap kelas dan akurasi rata-rata dalam satu percobaan. Akurasi tiap kelas dihitung berdasarkan jumlah data tangis yang diklasifikasi dengan benar dibagi jumlah seluruh data tangis bayi pada masing-masing kelas. Akurasi rata-rata dalam satu percobaan dihitung berdasarkan rata-rata akurasi tiap kelas. Alur pemilihan metode terbaik yang akan diterapkan pada aplikasi ini terdapat pada Gambar 3.9.



Gambar 3.9 Diagram Alur Uji Coba macam-macam Klasifikasi dan Perhitungan Akurasi klasifikasi

3.4 Perancangan Uji Coba dengan Data Latih Seimbang pada Tiap Kelas

Pada sub bab ini akan dijelaskan tentang perancangan uji coba data latih seimbang pada pembuatan "Aplikasi Pendeteksi Suara Tangis Bayi". Diharapkan pada tahap ini ditemukan apakah *K-nearest Neighbor* dengan nilai kombinasi nilai k=1 sampai k=5 mampu mengklasifiksikan data uji pada dengan data latih seimbang pada tiap kelas.

Dalam percancangan uji coba ini, data masukan berupa rekaman suara tangis bayi dengan format ".wav" diproses melalui ekstraksi fitur MFCC dari *package* tuneR dengan kombinasi parameter *wintime* 0,08 detik dengan tipe data vektor. Setelah itu, hasil ekstraksi fitur dari data masukan akan dinormalisasi dengan kombinasi metode normalisasi, yaitu *Standard Deviation Normalization*. Data hasil ekstraksi fitur berupa vektor yang telah dinormalisasi akan dimasukkan pada *dataframe* masing-masing kelas. Sebelum dilakukan klasifikasi, dataset berupa *dataframe* yang berjumlah 139 data akan dipisah sebanyak 19 data latih dari masing-masing kelas dan sisanya akan digunakan sebagai data uji. Proses klasifikasi *K-nearest Neighbor* dilakukan dengan menggunakan kombinasi nilai k sebesar 1 sampai 5. Setelah tahap klasifikasi selesai, akurasi akan dihitung untuk mengetahui seberapa besar keakuratan metode *K-nearest Neighbor* untuk melakukan klasifikasi pada suara tangis bayi ini. Akurasi pada uji coba ini ada 2 macam, yaitu akurasi tiap kelas dan akurasi rata-rata dalam satu percobaan. Akurasi tiap kelas dihitung berdasarkan jumlah data tangis yang diklasifikasi dengan benar dibagi jumlah seluruh data tangis bayi pada masing-masing kelas. Akurasi rata-rata dalam satu percobaan dihitung berdasarkan rata-rata akurasi tiap kelas. Alur peranangan uji coba dengan data latih seimbang terdapat pada Gambar 3.9.

3.5 Perancangan Uji Coba dengan Metode Klasifikasi Lain

Pada sub bab ini akan dijelaskan tentang perancangan uji coba data latih seimbang pada pembuatan "Aplikasi Pendeteksi Suara Tangis Bayi". Diharapkan pada tahap ini ditemukan apakah ada algoritma klasifikasi yang mampu memberikan akurasi klasifikasi lebih baik selain *K-nearest Neighbor* dengan *k*=1. Algoritma klasifikasi yang digunakan ialah *Naive Bayes, Neural Network, dan Support Vector Machine*.

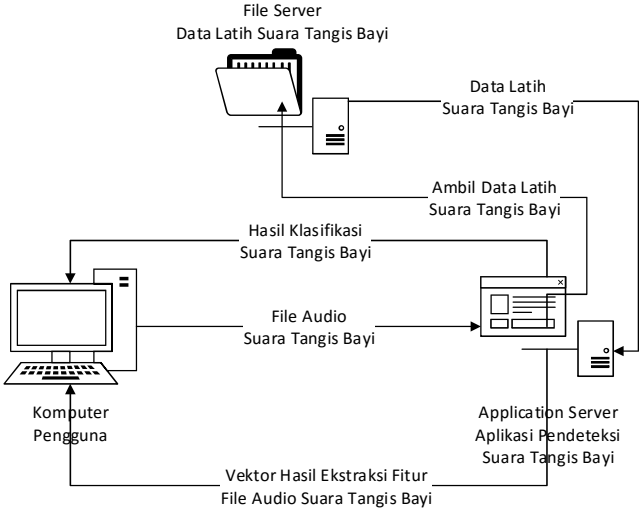
Dalam percancangan uji coba ini, data masukan berupa rekaman suara tangis bayi dengan format ".wav" diproses melalui ekstraksi fitur MFCC dari *package* tuneR dengan kombinasi parameter *wintime* 0,08 detik dengan tipe data vektor. Setelah itu, hasil ekstraksi fitur dari data masukan akan dinormalisasi dengan kombinasi metode normalisasi, yaitu *Standard Deviation Normalization*. Data hasil ekstraksi fitur berupa vektor yang telah dinormalisasi akan dimasukkan pada *dataframe* masing-masing kelas. Sebelum dilakukan klasifikasi, dataset berupa *dataframe* yang berjumlah 139 data akan dipisah sebanyak 85% data latih dan 15% data uji dari masing-masing kelas. Proses klasifikasi dilakukan dengan metode *Naive Bayes*, *Resilient Backpropagation Neural Network*, dan *Support Vector Machine*. Pada *Resilient Backpropagation Neural Network*, fungsi aktivasi yang digunakan adalah fungsi *sigmoid*, *hidden layer* yang digunakan sebanyak dua dengan 7 dan 5 node pada tiap *hidden layer*, dan *learning rate* yang digunakan sebesar 0,02. Pada *Support Vector Machine*, *kernel* yang digunakan ialah *polynomial* dan metode klasifikasi *One-againts-one*. Setelah tahap klasifikasi selesai, akurasi akan dihitung untuk mengetahui seberapa besar keakuratan masing-masing metode dalam melakukan klasifikasi pada suara tangis bayi ini. Akurasi pada uji coba ini ada 2 macam, yaitu akurasi tiap kelas dan akurasi rata-rata dalam satu percobaan. Akurasi tiap kelas dihitung berdasarkan jumlah data tangis yang diklasifikasi dengan benar dibagi

jumlah seluruh data tangis bayi pada masing-masing kelas. Akurasi rata-rata dalam satu percobaan dihitung berdasarkan rata-rata akurasi tiap kelas. Alur peranangan uji coba dengan data latih seimbang terdapat pada Gambar 3.9.

3.6 Perancangan Sistem

Pada sub bab ini akan dijelaskan tentang perancangan sistem aplikasi pendeteksi suara tangis bayi dengan menggunakan metode perancangan secara terstruktur. Dalam perancangan ini, akan dibahas tentang deskripsi umum kebutuhan sistem, kebutuhan fungional dari sistem, aliran informasi, serta deskripsi masing-masing proses yang digunakan pada sistem. Perancangan ini diimplementasikan berdasarkan metode terbaik yang telah didapat dari berbagai skenario uji coba yang telah dilakukan.

3.6.1 Arsitektur Perangkat Lunak



Gambar 3.10 Diagram Arsietektur Perangkat Lunak "Aplikasi Pendeteksi Suara Tangisan Bayi"

Arsitektur perangkat lunak "Aplikasi Pendeteksi Suara Tangisan Bayi" dapat dilihat pada Gambar 3.10. Saat pengguna

mengunggah *file* audio suara tangis bayi, aplikasi pendeteksi suara tangis bayi yang ada pada *application server* akan mengambil data latih dari *file server* data latih suara tangis bayi. Selanjutnya, data latih dan *file* audio yang diunggah oleh pengguna ke *application server* akan diproses melalui proses ekstraksi fitur MFCC, normalisasi hasil ekstraksi fitur dengan *Standard Deviation Normalization*, dan klasifikasi *K-nearest Neighbor*. Keluaran dari aplikasi ini ialah hasil ekstraksi fitur *file* audio suara tangis bayi yang diunggah oleh pengguna dan hasil klasifikasi dari *file* audio tangis bayi berupa teks jenis tangis bayi yang ditampilkan pada layar. Proses ekstraksi fitur MFCC, normalisasi hasil ekstraksi fitur dengan *Standard Deviation Normalization* akan dijelaskan pada sub bab Kebutuhan Fungsional.

3.6.2 Deskripsi Umum Kebutuhan

3.6.2.1 Antarmuka Pemakai

Aplikasi Pendeteksi Suara Tangis Bayi ini menggunakan antarmuka grafis (GUI). Pengguna dapat memasukkan data dan menjalankan aksi melalui *keyboard* dan *mouse.*

3.6.2.2 Antarmuka Perangkat Keras

Aplikasi Pendeteksi Suara Tangis Bayi ini berjalan di komputer *server* Shiny Apps.

3.6.2.3 Antarmuka Perangkat Lunak

Aplikasi Pendeteksi Suara Tangis Bayi ini adalah program yang akan dibangun berbasiskan web yang dikembangkan dengan menggunakan bahasa pemrograman R 3.3.1 dan *package* shiny dengan kakas bantu Rstudio.

3.6.2.4 Antarmuka Komunikasi

Aplikasi Pendeteksi Suara Tangis Bayi ini hanya dapat diakses degan menggunakan *web browser* komputer, *smartphone*, atau tablet yang terhubung dengan jaringan internet.

3.6.3 Kebutuhan Fungsional

Pada sub bab ini akan dijelaskan berbagai kebutuhan yang harus dipenuhi pada perangkat lunak.

F1. Pengguna dapat mengunggah *file* audio rekaman suara tangis bayi dengan format ".wav" untuk mengetahui arti suara tangis bayi.

F2. Sistem hanya dapat menerima *file* audio ".wav" dengan *sample rate* 11205 Hz *mono channel* dengan durasi maksimal 10 detik.

F3. Sistem melakukan ekstraksi fitur MFCC dari *pakage* tuneR pada audio rekaman suara tangis bayi dengan parameter sebagai berikut:
- *wintime*: 0,08 detik.
- *nbands*: 40.
- *hoptime*: Jika durasi rekaman tangis bayi kurang dari 2 detik, maka nilai *hoptime* sebesar durasi tangis bayi dikurangi 0,1 detik. Jika durasi rekaman bayi lebih dari 2 detik, maka nilai *hoptime* sebesar durasi tangis bayi dikurangi 1 detik.
- *preemph*: 0,97.
- *sumpower* = TRUE.
- *lifterexp*: 0,6.
- *bwidth*: 10.
- *usecmp* = TRUE.
- *numcep*: 10.

F4. Sistem melakukan normalisasi *Standard Deviation Normalization* pada hasil ekstraksi fitur.

F5. Sistem melakukan klasifikasi K-nearest Neighbor dengan *k*=1 pada data tangis bayi yang menghasilkan salah satu dari 4 jenis suara tangis bayi, yaitu:
- "Eairh" atau "Eh" yang berarti bayi ingin buang air besar, buang angin, atau bersendawa.
- "Heh" yang berarti bayi merasa tidak nyaman, saatnya untuk mengganti popok.

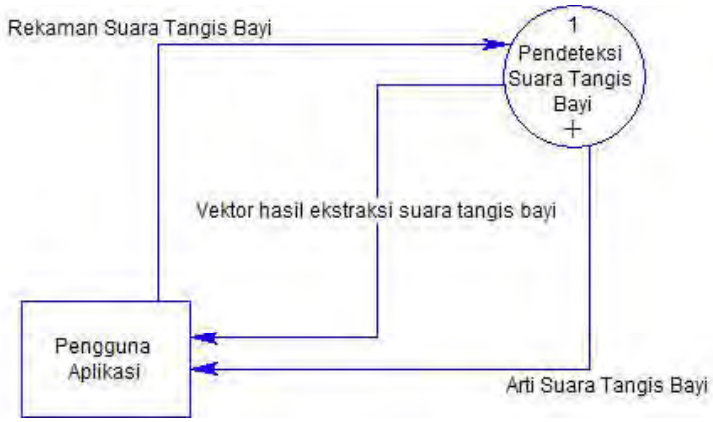
- "Neh" yang berarti bayi merasa lapar.
- "Owh" yang berarti bayi mulai mengantuk

F6. Sistem menampilkan hasil ekstraksi fitur yang telah dinormalisasi dan arti tangis bayi berupa teks pada pengguna berdasarkan audio rekaman suara tangis bayi yang diunggah oleh pengguna.

3.6.4 Aliran Informasi

Pada sub bab ini akan dijelaskan berbagai macam aliran informasi yang dimasukkan, diproses, dan dikeluarkan oleh sistem dengan menggunakan format *Data Flow Diagram*.

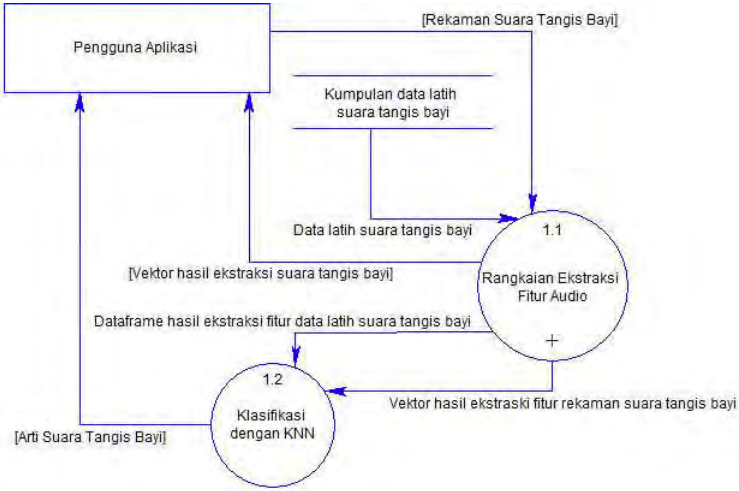
3.6.4.1 DFD Level 0



Gambar 3.11 DFD Level 0 Aplikasi Pendeteksi Suara Tangis Bayi

Pada DFD level 0 yang ditampilkan Gambar 3.11, proses yang dilakukan pada Aplikasi Pendeteksi Tangis Bayi ini adalah pengguna memasukkan rekaman suara tangis bayi dengan format ".wav", kemudian sistem memberikan keluaran berupa vektor hasil ekstraksi suara tangis bayi dan arti suara tangis bayi kepada pengguna.

3.6.4.2 DFD Level 1

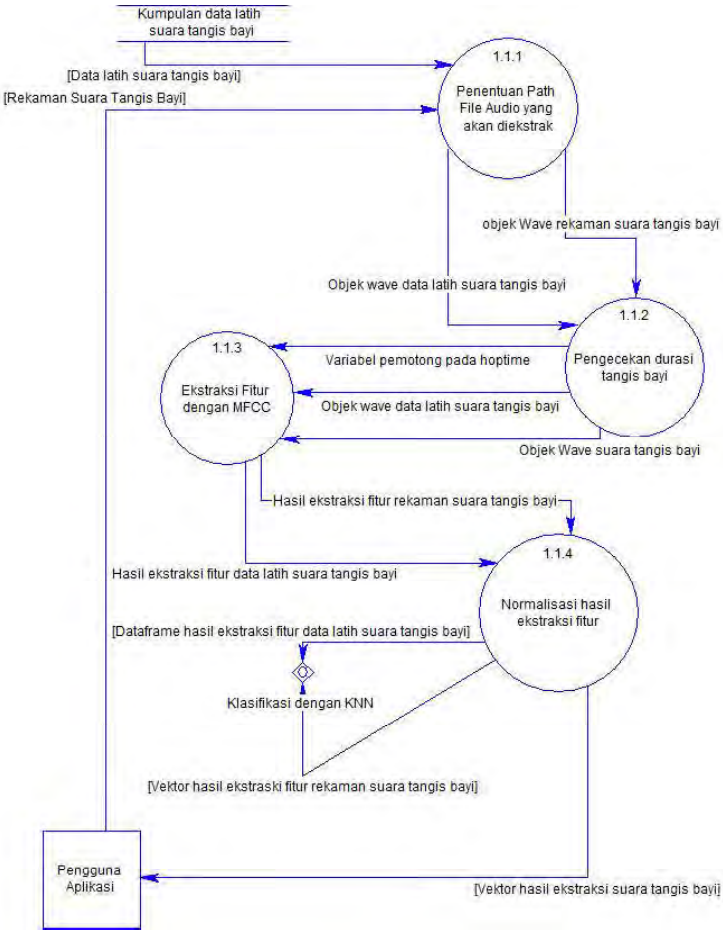


Gambar 3.12 DFD Level 1 Aplikasi Pendeteksi Suara Tangis Bayi

Pada DFD Level 1 yang ditampilkan Gambar 3.12, proses yang terjadi sebagai berikut:

1. Sistem melakukan ekstraksi fitur pada rekaman suara tangis bayi yang dimasukkan oleh pengguna dan kumpulan data latih suara tangis bayi berupa *file* audio format ".wav" melalui proses "Rangkaian Ekstraksi Fitur Audio", kemudian dari proses tersebut dihasilkan dataframe data latih suara tangis bayi dan vektor rekaman suara tangis bayi yang siap digunakan untuk proses klasifikasi.
2. Sistem memberikan keluaran vektor hasil ekstraksi fitur suara tangis bayi pada pengguna berupa teks.
3. Sistem melakukan klasifikasi *K-nearest Neighbor* dengan k=1 pada data uji berdasarkan data latih dengan, kemudian memberikan keluaran arti dari tangis bayi kepada pengguna aplikasi berupa teks.

3.6.4.3 DFD Level 2



Gambar 3.13 DFD Level 2 Proses 1 "extractAudio"

Pada DFD Level 2 yang ditampilkan Gambar 3.13, proses yang berlangsung ialah:

1. Sistem menentukan *path file* audio suara tangis bayi dari pengguna dan kumpulan data latih suara tangis

bayi yang akan diekstrak saat pengguna aplikasi memasukkan rekaman suara tangis bayi,.

2. Sistem melakukan pengecekan durasi *file* audio suara tangis bayi dari pengguna dan kumpulan data latih suara tangis bayi untuk menentukan besarnya nilai para variabel pemotong pada "*hoptime*" yang dipakai saat ekstraksi fitur setelah *file* audio suara tangis bayi dari pengguna dan kumpulan data latih suara tangis bayi dirubah menjadi objek Wave.
3. Setelah sistem melakukan pengecekan durasi audio, objek Wave *file* audio dari pengguna dan kumpulan data latih suara tangis bayi akan diproses melalui ekstraksi fitur MFCC dari *package* tuneR.
4. Sistem melakukan normalisasi hasil ekstraksi fitur *file* audio dari pengguna dan kumpulan data latih suara tangis bayi dengan *Standard Deviation Normalization*.
5. Hasil normalisasi ekstraksi fitur dari *file* audio suara tangis bayi dari pengguna akan ditampilkan oleh sistem berupa teks.
6. Hasil ekstraksi fitur dari *file* audio suara tangis bayi dari pengguna dan kumpulan data latih suara tangis bayi yang sudah dinormalisasai akan diproses oleh sistem melalui klasifikasi *K-nearest Neighbor* dengan k=1.

3.6.5 Deskripsi Proses

3.6.5.1 Proses 1.1-Rangkaian Ekstraksi Fitur Audio

Pengguna aplikasi memasukkan *file* audio suara tangis bayi dengan format ".wav", kemudian sistem melakukan proses "rangkaian ekstraksi fitur audio" pada *file* audio dari pengguna dan kumpulan data latih suara tangis bayi. Setelah sistem melakukan proses "rangkaian proses ekstraksi fitur audio", sistem akan menampilkan hasil ekstraksi fitur *file* audio penggun berupa teks. Hasil ekstraksi fitur *file* audio suara tangis bayi dari pengguna berupa vektor dan kumpulan data latih suara tangis bayi berupada *datafame* akan digunakan pada proses 1.2.

3.6.5.2 Proses 1.2-Klasifikasi dengan KNN

Sistem melakukan klasifikasi dengan *K-nearest Neighbor* berdasarkan *datafame* kumpulan data latih suara tangis bayi dan vektor *file* audio suara tangis bayi dari pengguna yang berasal dari proses 1.1. *Dataframe* kumpulan data latih suara tangis bayi terdiri dari baris sebanyak data latih pada masning-masing kelas dengan 21 kolom yang berisi 20 kolom hasil ekstraksi fitur MFCC yang telah dinormalisasi dengan *Standard Deviation Normalization* dan 1 kolom yang berisi kelas masing-masing data latih. Vektor *file* audio suara tangis bayi dari pengguna terdiri dari baris sebanyak data latih pada masning-masing kelas dengan 21 kolom yang berisi 20 kolom hasil ekstraksi fitur MFCC yang telah dinormalisasi dengan *Standard Deviation Normalization* dan 1 kolom yang berisi kelas masing-masing data latih. Nilai *k* pada *K-nearest Neighbor* sebesar 1. Setelah melaui proses klasifikasi, sistem akan memberikan hasil klasifikasi kepada pengguna berupa teks jenis tangis bayi dari *file* audio terkait.

3.6.5.3 Proses 1.1.1-Penentuan Path File Audio yang akan diekstrak

Sistem menentukan *path file* audio suara tangis bayi dari pengguna dan kumpulan data latih suara tangis bayi yang akan diekstraksi fitur dengan MFCC, kemudian membaca audio tersebut sebagai objek wave dengan fungsi "readWave" dari *package* tuneR agar dapat diproses pada ekstraksi fitur. Objek wave *file* audio dari pengguna dan kumpulan data latih suara tangis bayi akan masuk pada proses 1.1.2.

3.6.5.4 Proses 1.1.2-Pengecekan durasi tangis bayi

Sistem mengecek durasi suara tangis bayi dari objek Wave *file audio* suara tangis bayi dari pengguna dan kumpulan data latih suara tangis bayi, kemudian menentukan nilai variabel pemotong pada *hoptime* yang dipakai pada ekstraksi fitur MFCC dari *package* tuneR. Nilai variabel tersebut adalah 0,1 jika durasi *file* audio dari pengguna atau kumpulan data latih

suara tangis bayi kurang dari 2 detik. Sebaliknya, jika durasi *file* audio dari pengguna atau kumpulan data latih suara tangis bayi lebih dari 2 detik, maka variabel pemotong tersebut bernilai 1. Objek wave *file* audio dari pengguna atau kumpulan data latih suara tangis bayi beserta variabel pemotong pada *hoptime* akan masuk pada proses 1.1.3.

3.6.5.5 Proses 1.1.3-Ekstraksi Fitur dengan MFCC

Sistem melakukan ekstraksi fitur pada objek Wave *file* audio dari pengguna atau kumpulan data latih suara tangis bayi dengan menggunakan algoritma MFCC dengan parameter *wintime* sebesar 0,08. Algoritma MFCC yang dipakai pada sistem ini berasal dari *package* tuneR. Hasil dari ekstraksi fitur ini ialah vektor dengan dimensi 1 baris 20 kolom untuk setiap data, baik data latih maupun data uji. Selanjutnya, hasil ekstraksi fitur kumpulan data latih suara tangis bayi dan hasil ekstraksi fitur *file* audio suara tangis bayi dari penggua akan masuk pada proses 1.1.4.

3.6.5.6 Proses 1.1.4-Normalisasi Hasil Ekstraksi Fitur

Sistem melakukan normalisasi hasil ekstraksi fitur *file* audio dari pengguna dan hasil ekstraksi fitur kumpulan data latih suara tangis bayi. Metode normalisasi yang digunakan adalah *Standard Deviation Normalization*. Hasil dari normalisasi ekstraksi fitur kumpulan data latih suara tangis bayi akan dimasukkan pada *dataframe* masing-masing kelas dengan jumlah baris sebanyak data latih pada masing-masing kelas dan jumlah kolom sebanyak 21 yang terdiri dari 20 kolom nilai hasil ekstraksi fitur dan 1 kolom berisi label jenis tangis bayi masing-masing kelas. Hasil dari normalisasi ekstraksi fitur *file* audio dari pengguna akan dimasukkan pada vektor dengan jumlah baris sebanyak 1 dan jumlah kolom sebanyak 20 yang terdiri dari nilai hasil ekstraksi fitur. Selanjutnya, kedua data tersebut akan masuk pada proses klasifikasi KNN yang ada pada proses 1.2.

[Halaman ini sengaja dikosongkan]

BAB IV
IMPLEMENTASI

Setelah dijelaskan mengenai proses yang dijalankan untuk melakukan klasifikasi rekaman suara tangis bayi pada Bab III, pada Bab IV ini akan dijelaskan mengenai implementasi yang dilakukan pada proses percobaan.

4.1 Lingkungan Implementasi

Dalam mengimplementasikan klasifikasi pada rekaman suara tangis bayi diperlukan beberapa perangkat pendukung sebagai berikut.

4.1.1. Lingkungan Implementasi Perangkat Keras

Spesifikasi perangkat keras yang digunakan pada lingkungan pengembangan aplikasi adalah sebagai sebuah komputer PC atau laptop sebagai komputer pemroses dengan spesifikasi:

Prosesor	:	Intel® Core™ i5 CPU M460 @ 2.53 GHz
RAM	:	8 GB
Tipe Sistem	:	Sistem Operasi 64-bit

4.1.2. Lingkungan Implementasi Perangkat Lunak

Spesifikasi perangkat lunak yang digunakan pada lingkungan pengembangan aplikasi adalah sebagai berikut:

Sistem Operasi	:	Windows Embedded 8.1 Industry Pro 64-bit
IDE	:	Rstudio dengan R versi 3.3.1

4.2 Implementasi Pemilihan Metode Terbaik

Sub bab implementasi pemilihan metode terbaik ini menjelaskan tentang pembangunan sistem uji coba yang menerapkan kombinasi proporsi data latih dan data uji, kombinasi nilai parameter *wintime* pada ekstraksi fitur MFCC dari package tuneR, kombinasi metode normalisasi ekstraksi fitur, dan kombinasi nilai *k* pada klasifikasi *K-nearest Neighbor*. Hasil terbaik berupa akurasi rata-rata dalam satu

percobaan dari penerapan kombinasi metode tersebut akan digunakan pada pembuatan aplikasi

4.2.1 Implementasi Proses Ekstraksi Fitur MFCC

Pada tahap ini dijelaskan implementasi dari proses ekstraksi fitur pada rekaman suara tangis bayi dengan fungsi extractAudio yang ada pada Kode Sumber 4.1. Dalam proses extractAudio, fungsi akan membaca *file* audio berdasarkan *path* yang telah ditentukan. Setelah *file* audio berhasil dibaca, durasi audio akan dicek apakah lebih dari 2 detik atau tidak. Jika durasi audio kurang dari 2 detik, maka variabel pemotong *hoptime* akan bernilai 0,1. Jika durasi audio lebih dari 2 detik, maka variabel pemotong *hoptime* akan bernilai 1. Kemudian, fungsi ekstraksi fitur MFCC bernama "melfcc" dari *package* tuneR dengan kombinasi parameter *wintime* sebesar 0,02 sampai 0,08 digunakan untuk mengekstrak rekaman suara tangis bayi menjadi frekuensi Mel. Setelah proses ekstraksi fitur selesai, hasil dari ekstraksi fitur akan dinormalisasi. Hasil dari fungsi extractAudio ialah sebuah vector yang akan dimasukkan ke *dataframe* untuk diklasifikasi dengan *K-nearest Neighbor*.

```
extractAudio <- function(x, count, Files)
{
  #parsing direktori file yang akan diekstrak
  tmp_path = x
  listFiles = Files
  readPath <- paste(tmp_path,"\\",listFiles, sep = "")

  # cat("\n","iteration = ", count , readPath, "\n")

  #membaca file audio dari variabel readPath
  sndObj <- readWave(readPath)

  #hitung durasi tangis bayi
  left <- sndObj@left / sndObj@samp.rate
  left_length <- length(left)
  totsec <- left_length / sndObj@samp.rate

  # print(totsec)
  #pengecekan durasi tangis bayi
  if (totsec < 2){
    sec <- 0.1
```

```
  } else {
    sec <- 1
  }

  #ekstraksi fitur audio dengan MFCC
  mfcc <- as.vector(melfcc(sndObj, sr=sndObj@samp.rate,
                           wintime = 0.08, nbands = 40,
                           hoptime = totsec-sec,
                           preemph = 0.97, numcep = 10,
                           sumpower = TRUE, lifterexp = 0.6,
                           bwidth = 10, usecmp = TRUE))

  #normalisasi dengan standard deviation normalization
  n <- (mfcc-mean(mfcc))/sd(mfcc)

  #normalisasi dengan decimal normalization
  # n <- mfcc / 10^3

  #normalisasi dengan min-max normalization
  # n <- (mfcc - min(mfcc)) / (max(mfcc) - min(mfcc))

  # print(mfcc)
  return(n)
}
```

Kode Sumber 4.1 Implementasi Ekstraksi Fitur pada Pemiihan Metode Terbaik

4.2.2 Implementasi Pembagian proporsi data latih dan data Uji

Pembagian proporsi data latih dilakukan dengan mengambil data pertama sampai batas yang ditentukan pada masing-masing jenis tangis bayi. Pembagian proporsi data uji dilakukan dengan mengambil sisa data masing-masing jenis tangis bayi dari yang belum ada pada data latih. Pembagian data latih dan data uji diimplementasikan pada Kode Sumber 4.2 dengan mengganti pengambilan baris pada *dataframe* masing-masing kelas secara manual pada setiap perocbaan. Kombinasi proporsi data latih dan data uji yang digunakan pada pemilihan metode terbaik dapat dilihat pada Tabel 4.1 dan Tabel 4.2.

Tabel 4.1 Pembagian Proporsi Data Latih

Jenis Tangis	Total Data	Data Latih 90%	Data Latih 85%	Data Latih 80%	Data Latih 75%	Data Latih 70%
Eairh	25	23	21	20	19	18
Eh	21	19	18	17	16	15
Heh	23	21	20	18	17	16
Neh	35	32	30	28	26	25
Owh	35	32	30	28	26	25

Tabel 4.2 Pembagian Proporsi Data Uji

Jenis Tangis	Total Data	Data Uji 10%	Data Uji 15%	Data Uji 20%	Data Uji 25%	Data Uji 30%
Eairh	25	2	4	5	6	7
Eh	21	2	3	4	5	6
Heh	23	2	3	5	6	7
Neh	35	3	5	7	9	10
Owh	35	3	5	7	9	10

```
# pembagian data latih
train1 <- class1[1:21,]
train2 <- class2[1:18,]
train3 <- class3[1:20,]
train4 <- class4[1:30,]
train5 <- class5[1:30,]

# pembagian data uji
test1 <- class1[22:25,]
test2 <- class2[19:21,]
test3 <- class3[21:23,]
test4 <- class4[31:35,]
test5 <- class5[31:35,]

# data latih
data_train <- rbind(train1, train2, train3, train4, train5)

# data uji
data_test <- rbind(test1, test2, test3, test4, test5)
```

Kode Sumber 4.2 Implementasi Pembagian Data Latih dan Data Uji pada Pemilihan Metode Terbaik

4.2.3 Implementasi Proses Normalisasi

Pada tahap ini akan dijelaskan tentang implementasi metode normalisasi pada hasil ekstraksi fitur MFCC dari *package* tuneR pada data suara tangis bayi.

4.2.3.1 Implementasi Standard Deviation Normalization

Berikut ini adalah implementasi *Standard Deviation Normalization* yang terdapat pada Kode Sumber 4.3.

```
n <- (mfcc-mean(mfcc))/sd(mfcc)
```

Kode Sumber 4.3 Implementasi *Standard Deviation Normalization* pada Pemiihan Metode Terbaik

4.2.3.2 Implementasi Decimal Normalization

Berikut ini adalah implementasi *Decimal Normalization* yang terdapat pada Kode Sumber 4.4.

```
n <- mfcc / 10^3
```

Kode Sumber 4.4 Implementasi *Decimal Normalization* pada Pemiihan Metode Terbaik

4.2.3.3 Implementasi Min-Max Normalization

Berikut ini adalah implementasi *Min-Max Normalization* yang terdapat pada Kode Sumber 4.5.

```
n <- (mfcc - min(mfcc)) / (max(mfcc) - min(mfcc))
```

Kode Sumber 4.5 Implementasi *Min-Max Normalization* pada Pemiihan Metode Terbaik

4.2.4 Implementasi Proses Klasifikasi *K-nearest Neighbor*

Tahap terakhir setelah ekstraksi fitur dan normalisasi yaitu klasifikasi. Pada tahap ekstraksi fitur telah dihasilkan 139 dataset dengan 20 kolom fitur frekuensi Mel. Pada tahap ini, data latih dan data uji yang telah dibagi akan diklasifikasikan dengan *K-nearest Neighbor*. Berikut ini adalah implementasi *K-nearest Neighbor* yang terdapat pada Kode Sumber 4.6.

```
# perulangan kombinasi nilai k pada KNN
for(k in 1:1)
{
  # perulangan KNN
  for(i in 1:nrow(data_test))
  {
    for (j in 1:nrow(data_train))
```

```
    {
      d[j] <- sqrt(sum((data_test[i,1:(ncol(data_test)-2)]-
                 data_train[j,1:(ncol(data_train)-2)])^2))

      c[j] <- data_train$c[j]
    }
    hasil <- data.frame(class = c, distance = d)
    hasil_final <- hasil[order(hasil$distance),]
    C1 <- 0
    C2 <- 0
    C3 <- 0
    C4 <- 0

    for(a in 1:k)
    {
      ifelse(hasil_final[a,1] == 1, C1 <- C1 + 1,
        ifelse(hasil_final[a,1] == 2, C2 <- C2 + 1,
          ifelse(hasil_final[a,1] == 3, C3 <- C3 + 1,
                C4 <- C4 + 1)
             )
      )
    }

    new <- c("C1", "C2", "C3", "C4")[which.max(c(C1, C2, C3,
C4))]

    classified[i] <- ifelse(new == "C1",
                      classified[i] <- 1,
                       ifelse(new == "C2",
                        classified[i] <- 2,
                         ifelse(new == "C3",
                          classified[i] <- 3,
                           classified[i] <- 4)
                                  )
                            )
  }
```

Kode Sumber 4.6 Implementasi *K-nearest Neighbor* Pada Pemiihan Metode Terbaik

4.2.5 Implementasi Perhitungan Akurasi

Akurasi klasifikasi per kelas pada pemilihan metode terbaik dilakukan dengan memasukkan kelas masing-masing data uji dan hasil prediksi ke dalam sebuah variabel *dataframe*, kemudian membuat *confussion matrix* dengan bantuan *package* caret. Selanjutnya, objek “byClass” pada *confussion matrix*

tersebut disimpan dalam format csv untuk dianalisa. Rata-rata akurasi dalam satu percobaan dihitung dengan merata-rata nilai akurasi klasifikasi per kelas pada Microsoft Excel berdasarkan *file* ".csv" masing-masing uji coba. Implementasi perhitungan akurasi ini terdapat pada Kode Sumber 4.7.

```
predfiles <- data.frame(file = data_test$file,
             actual = data_test$c, pred_result = classified)

# confussion matrix percobaan
CM <- confusionMatrix(data = classified, predfiles$actual)

# penentuan path penyimpanan file
namafile_path <- "D:\\Tugas Akhir\\Pengujian\\Hasil Uji\\"

namafile_byClass <- paste(
  namafile_path, "Min-Max Traning70%byJenis Tangis", "k",
  "=", k, wintime=0.02", ".csv", sep = " ")

# simpan file dengan format csv
write.csv2(CM$byJenis Tangis, namafile_byClass, row.names =
FALSE, col.names = FALSE)
```

Kode Sumber 4.7 Implementasi Perhitungan Akurasi pada Pemilihan Metode Terbaik

4.3 Implementasi Uji Coba dengan Metode Klasifikasi lain.

Sub bab ini menjelaskan kode uji coba dengan metode klasifikasi lain pada data latih dan data uji yang telah dibentuk dari ekstraksi fitur. Metode klasifikasi yang digunakan ialah klasifikasi *Naive Bayes*, *Neural Network*, dan *Support Vector Machice*. Ketiga metode klasifikasi tersebut diimplementasikan dengan menggunakan *package* pada bahasa pemerograman R yang akan dijelaskan pada masing-masing sub bab ini. Setelah uji coba pada masing-masing metode klasifikasi dilakukan, akurasi dihitung dengan mengeksekusi Kode Sumber 4.7.

4.3.1 Implementasi Proses Klasifikasi *Naive Bayes*

Berikut ini adalah implementasi klasifikasi *Naive Bayes* dengan menggunakan *package* e1071 yang terdapat pada Kode Sumber 4.8.

```
# proses pelatihan data latih
training <- naiveBayes(x = data_train[,1:20],
                       y = data_train$c, laplace = 3)

# proses klasifikasi dengan Naive Bayes
pr.nb.result <- as.data.frame(
  round(predict(training, data_test[,1:20], type = "raw"))
  )

classified <- rep(0, nrow(data_test))

# Hitung jumlah klasifikasi tiap kelas
for(i in 1:nrow(pr.nb.result))
{
  ifelse (pr.nb.result[i,1] >= 0.5, x <- 1,
          ifelse(pr.nb.result[i,2] >= 0.5, x <- 2,
                 ifelse(pr.nb.result[i,3] >= 0.5, x <- 3,
                        x <- 4)
                 )
          )
  classified[i] <- x
}
```

Kode Sumber 4.8 Implementasi Klasifikasi *Naive Bayes*

4.3.2 Implementasi Proses Klasifikasi *Neural Network*

Berikut ini adalah implementasi klasifikasi *Neural Network* dengan algoritma *resilient backpropagation*, dua *hidden layer* dengan jumlah node 7 dan 5 pada masing-masing *layer*, koefisien *learning rate* sebesar 0,02, dan fungsi aktivasi fungsi logistik yang menggunakan *package* neuralnet dan terdapat pada Kode Sumber 4.9.

```
# data latih
data_train <- rbind(train1, train2, train3, train4, train5)
data_train_nn <- rbind(train1[,1:20], train2[,1:20],
                       train3[,1:20], train4[,1:20],
                       train5[,1:20])

data_train_nn$c1 <- c(rep(1, 21), rep(1, 18), rep(0, 20),
rep(0, 30), rep(0,30))
data_train_nn$c2 <- c(rep(0, 21), rep(0, 18), rep(1, 20),
rep(0, 30), rep(0,30))
data_train_nn$c3 <- c(rep(0, 21), rep(0, 18), rep(0, 20),
rep(1, 30), rep(0,30))
data_train_nn$c4 <- c(rep(0, 21), rep(0, 18), rep(0, 20),
rep(0, 30), rep(1,30))
```

```
# data uji
data_test <- rbind(test1, test2, test3, test4, test5)

# pembentukan target
f <- as.formula(c1 + c2 + c3 + c4 ~
                  V1 + V2 + V3 + V4 + V5 +
                  V6 + V7 + V8 + V9 + V10 +
                  V11 + V12 + V13 + V14 + V15 +
                  V16 + V17 + V18 + V19 + V20)

# proses pelatihan data latih
nn <- neuralnet(f, data = data_train_nn,hidden = c(7,5),
                learningrate = 0.02, threshold = 0.0001,
                linear.output = FALSE, act.fct = "logistic",
                algorithm = "rprop+")

# proses klasifikasi Neural Network
pr.nn <- compute(nn, data_test[,1:20])
pr.nn.result <- as.data.frame(round(pr.nn$net.result, digits
= 0))
classified <- c(rep(0, nrow(pr.nn.result)))

# hitung hasil klasifikasi
for(i in 1:nrow(pr.nn.result))
{
  ifelse (pr.nn.result[i,1] >= 0.5, x <- 1,
          ifelse(pr.nn.result[i,2] >= 0.5, x <- 2,
                 ifelse(pr.nn.result[i,3] >= 0.5, x <- 3,
                        x <- 4)
                 )
          )
  classified[i] <- x
}
```

Kode Sumber 4.9 Implementasi Klasifikasi *Neural Network*

4.3.3 Implementasi Proses Klasifikasi *Support Vector Machine*

Berikut ini adalah implementasi klasifikasi *Support Vector Machine* dengan kernel *polynomial* tanpa *cross validation* yang menggunakan *package* e1071 dan terdapat pada Kode Sumber 4.10.

```
# proses pelatihan data latih
training <- svm(x = data_train[,1:20], y = data_train$c,
                scale  =TRUE, type = "C-classification",
                kernel = "polynomial",cross = 0)
```

```
# proses klasifikasi Support Vector Machine
classified <- predict(training, data_test[,1:20])

table(classified[classified == data_test$c])
```

Kode Sumber 4.10 Implementasi Klasifikasi *Support Vector Machine*

4.4 Implementasi Perancangan Sistem

Sub bab implementasi perancangan sistem ini menjelaskan tentang pembangunan sistem aplikasi pendeteksi suara tangis bayi yang menerapkan kombinasi terbaik berdasarkan uji coba pemilihan metode terbaik yang terdiri dari proporsi data latih dan data uji, nilai parameter *wintime* pada ekstraksi fitur MFCC dari package tuneR, metode normalisasi ekstraksi fitur, dan nilai *k* pada klasifikasi *K-nearest Neighbor*. Implementasi perancangan sistem ini dibuat dengan bantuan *package* shiny. Hasil implementasi ini ialah sebuah perangkat lunak dengan GUI yang dapat digunakan oleh pengguna dengan mudah. Pengguna dapat memasukkan *file* audio suara tangis bayi, kemudian sistem akan memberikan keluaran berupa hasil ekstraksi fitur suara tangis bayi dan arti suara tangis bayi dari *file* audio yang dimasukkan oleh pengguna.

4.4.1 Implementasi Pembentukan Data Latih

Pada tahap ini dijelaskan implementasi dari proses pembentukan data latih pada data latih suara tangis bayi dengan fungsi “extractAudio” yang ada pada Kode Sumber 4.8. Dalam proses “extractAudio”, fungsi akan membaca *file* audio data latih berdasarkan *path* yang telah ditentukan. Setelah *file* audio berhasil dibaca, durasi audio akan dicek apakah lebih dari 2 detik atau tidak. Jika durasi audio kurang dari 2 detik, maka variabel pemotong *hoptime* akan bernilai 0,1. Jika durasi audio lebih dari 2 detik, maka variabel pemotong *hoptime* akan bernilai 1. Kemudian, fungsi ekstraksi fitur MFCC bernama “melfcc” dari *package* tuneR dengan nilai parameter *wintime* sebesar 0,08 digunakan untuk mengekstrak data laith rekaman suara tangis bayi menjadi frekuensi Mel. Setelah proses ekstraksi fitur data latih suara tangis bayi selesai, hasil dari

ekstraksi fitur akan dinormalisasi dengan *Standard Deviation Normalization*. Hasil dari pembentukan data latih ini ialah *dataframe* data latih dengan jumlah baris data latih pada masing-masing kelas dan 21 kolom yang terdiri dari 20 kolom hasil ekstraksi fitur MFCC dan 1 kolom yang berisi *flag* kelas tiap data latih yang akan diklasifikasi dengan *K-nearest Neighbor*.

```
extractAudio <- function(x, count, Files)
{
  #parsing direktori file yang akan diekstrak
  tmp_path = x
  # print(tmp_path)
  listFiles = Files
  # print(listFiles)
  readPath <- paste(tmp_path,"\\",listFiles, sep = "")

  # cat("\n","iteration = ", count , readPath, "\n")

  #membaca file audio dari variabel readPath
  sndObj <- readWave(readPath)

  #hitung durasi tangis bayi
  left <- sndObj@left / sndObj@samp.rate
  left_length <- length(left)
  totsec <- left_length / sndObj@samp.rate

  # print(totsec)

  #pengecekan durasi tangis bayi
  if (totsec < 2){
    sec <- 0.1
  } else {
    sec <- 1
  }

  #ekstraksi fitur audio dengan MFCC
  mfcc <- as.vector(melfcc(sndObj, sr=sndObj@samp.rate,
                           wintime = 0.08, nbands = 40,
                           hoptime = totsec-sec,
                           preemph = 0.97, numcep = 10,
                           sumpower = TRUE, lifterexp = 0.6,
                           bwidth = 10, usecmp = TRUE))

  #normalisasi dengan standard deviation normalization
  n <- (mfcc-mean(mfcc))/sd(mfcc)
```

```
  return(n)
}

x1 = "D:\\Tugas Akhir\\Eairh"
x2 = "D:\\Tugas Akhir\\Eh"
x3 = "D:\\Tugas Akhir\\Heh"
x4 = "D:\\Tugas Akhir\\Neh"
x5 = "D:\\Tugas Akhir\\Owh"

listFiles1 = list.files(path = x1, pattern = ".wav")
listFiles2 = list.files(path = x2, pattern = ".wav")
listFiles3 = list.files(path = x3, pattern = ".wav")
listFiles4 = list.files(path = x4, pattern = ".wav")
listFiles5 = list.files(path = x5, pattern = ".wav")

class1 <- matrix(NA, nrow = length(listFiles1), ncol = 20 )
class2 <- matrix(NA, nrow = length(listFiles2), ncol = 20 )
class3 <- matrix(NA, nrow = length(listFiles3), ncol = 20 )
class4 <- matrix(NA, nrow = length(listFiles4), ncol = 20 )
class5 <- matrix(NA, nrow = length(listFiles5), ncol = 20 )

for (j in 1:length(listFiles1))
  class1[j,] <- extractAudio(x1, j, listFiles1[j])
for (j in 1:length(listFiles2))
  class2[j,] <- extractAudio(x2, j, listFiles2[j])

for (j in 1:length(listFiles3))
  class3[j,] <- extractAudio(x3, j, listFiles3[j])

for (j in 1:length(listFiles4))
  class4[j,] <- extractAudio(x4, j, listFiles4[j])

for (j in 1:length(listFiles5))
  class5[j,] <- extractAudio(x5, j, listFiles5[j])

class1 <- as.data.frame(class1)
class2 <- as.data.frame(class2)
class3 <- as.data.frame(class3)
class4 <- as.data.frame(class4)
class5 <- as.data.frame(class5)

class1$c <- 1
class2$c <- 1
class3$c <- 2
class4$c <- 3
class5$c <- 4

class1$file <- listFiles1
```

```
class2$file <- listFiles2
class3$file <- listFiles3
class4$file <- listFiles4
class5$file <- listFiles5

train1 <- class1[1:21,]
train2 <- class2[1:18,]
train3 <- class3[1:20,]
train4 <- class4[1:30,]
train5 <- class5[1:30,]

data_train <- rbind(train1, train2, train3, train4, train5)
```

Kode Sumber 4.11 Implementasi Pembentukan Data Latih pada Pembuatan Aplikasi

4.4.2 Implementasi Menggungah Data Uji

Sistem akan mengecek apakah variabel data untuk menampung data uji yang diunggah masih kosong atau tidak. Jika data sudah terisi, maka proses selanjutnya (ekstraksi fitur data uji) dapat dilakukan. Namun, jika data uji masih kosong, maka sistem akan menunggu sampai data uji ada dan lanjut ke proses berikutnya. Implementasi mengunggah data uji ini ada pada Kode Sumber 4.12.

```
data <- reactive({
    file1 <- input$file1
    if(is.null(file1)) {return()}
    readWave(file1$datapath)
})
```

Kode Sumber 4.12 Implementasi Mengunggah Data Uji pada Aplikasi

4.4.3 Implementasi Ekstraksi Fitur Data Uji

Dalam proses renderText hasil ekstraksi fitur, fungsi membaca ketersediaan *file* audio data uji. Jika *file* audio data uji tersedia, maka proses ekstraksi fitur dapat dilaksanakan. Namun, jika *file* data uji masih belum ada, maka sistem akan menunggu sampai data uji ada dan lanjut ke proses ekstraksi fitur.

Setelah *file* audio berhasil dibaca, durasi audio akan dicek apakah lebih dari 2 detik atau tidak. Jika durasi audio kurang dari 2 detik, maka variabel pemotong *hoptime* akan bernilai 0,1. Jika durasi audio lebih dari 2 detik, maka variabel

pemotong *hoptime* akan bernilai 1. Kemudian, fungsi ekstraksi fitur MFCC bernama "melfcc" dari *package* tuneR dengan nilai parameter *wintime* sebesar 0,08 digunakan untuk mengekstrak data uji rekaman suara tangis bayi menjadi frekuensi Mel. Setelah proses ekstraksi fitur data latih suara tangis bayi selesai, hasil dari ekstraksi fitur akan dinormalisasi dengan *Standard Deviation Normalization*. Hasil dari ekstraksi fitur data uji ini ialah vektor data uji dengan jumlah 1 baris dan 20 kolom berisi hasil ekstraksi fitur MFCC yang akan diklasifikasi dengan *K-nearest Neighbor*. Hasil ekstraksi fitur ini akan ditampilkan ke pengguna dengan format teks. Implementasi ekstraksi fitur data uji ini terdapat pada Kode Sumber 4.13.

```
output$ekstraksi <- renderText({
    if(is.null(data())) {return()}

    sndObj_t <- data()

    #hitung durasi tangis bayi
    left_t <- sndObj_t@left / sndObj_t@samp.rate
    left_length_t <- length(left_t)
    totsec_t <- left_length_t / sndObj_t@samp.rate

    if (totsec_t < 2){
      sec_t <- 0.1
    } else {
      sec_t <- 1
    }

    mfcc_t <- as.vector(melfcc(sndObj_t,
                          sr=sndObj_t@samp.rate,
                           wintime = 0.08, nbands = 40,
                           hoptime = totsec_t-sec_t,
                           preemph = 0.97, numcep = 10,
                           sumpower = TRUE,lifterexp = 0.6,
                           bwidth = 10, usecmp = TRUE))

    # print(mfcc_t)

    data_test <<- (mfcc_t-mean(mfcc_t))/sd(mfcc_t)

    data_test
})
```

Kode Sumber 4.13 Implementasi Ekstraksi Fitur Data Uji

4.4.4 Implementasi Klasifikasi KNN

Dalam proses "renderText" hasil klasifikasi, fungsi membaca ketersediaan *file* audio data uji. Jika *file* audio data uji tersedia, maka proses klasifikasi dapat dilaksanakan. Namun, jika *file* data uji masih belum ada, maka sistem akan menunggu sampai data uji ada dan lanjut ke proses klasifikasi.

Pada tahap ini, data latih dan data uji akan diklasifikasikan dengan *K-nearest Neighbor* yang terdapat pada Kode Sumber 4.14. Nilai *k* yang digunakan ialah 1.

```
output$klasifikasi <- renderText({
    print("klasifikasi")
    print(data_test)
    if(is.null(data())) {return()}
    c <- rep(0, nrow(data_train))
    d <- rep(0, nrow(data_train))
    classified <- "kelas"
    print("jumlah data train")
    print(nrow(data_train))
    for (j in 1:nrow(data_train))
    {
      d[j] <- sqrt(sum((data_test[1:(length(data_test))]-
                 data_train[j,1:(ncol(data_train)-2)])^2))

      c[j] <- data_train$c[j]
    }
    hasil <- data.frame(class = c, distance = d)
    hasil_final <- hasil[order(hasil$distance),]
    print(hasil_final)

    C1 <- 0
    C2 <- 0
    C3 <- 0
    C4 <- 0

    for(a in 1:1)
    {
      ifelse(hasil_final[a,1] == 1,
             C1 <- C1 + 1,
             ifelse(hasil_final[a,1] == 2,
                    C2 <- C2 + 1,
                    ifelse(hasil_final[a,1] == 3,
                           C3 <- C3 + 1, C4 <- C4 + 1)
             )
      )
    }
```

```
    new <- c("C1", "C2", "C3", "C4")[which.max(c(C1, C2, C3,
C4))]

    cry1 <- "Bayi ingin bersendawa / kentut / buang air
besar"
    cry2 <- "Bayi merasa tidak nyaman, waktunya ganti popok"
    cry3 <- "Bayi sedang lapar"
    cry4 <- "Bayi mulai mengantuk"

    classified <- ifelse(new == "C1", classified <- cry1,
                   ifelse(new == "C2", classified <- cry2,
                     ifelse(new == "C3", classified <- cry3,
                       classified <- cry4)
                         )
    )

    classified
  })
```

Kode Sumber 4.14 Implementasi KNN pada Aplikasi

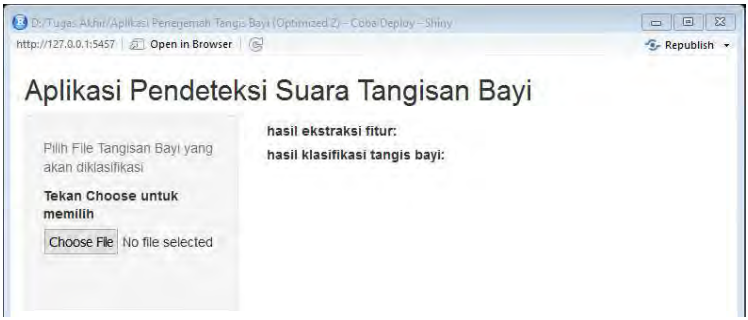
4.4.5 Implementasi GUI

GUI ini diimplementasikan dengan bantuan *package* shiny dalam bahasa pemrograman R. Pada GUI ini, terdapat panel untuk mengunggah *file* audio tangis bayi yang ingin diketahui artinya, label yang bertuliskan "hasil ekstraksi fitur" dan "hasil klasifikasi tangis bayi", serta textOutput yang nantinya menampilkan hasil ekstraksi fitur dan hasil klasifikasi. Hasil implementasi GUI aplikasi pendeteksi tangis bayi dapat dilihat pada Gambar 4.1 dan Gambar 4.2. Implementasi GUI aplikasi pendeteksi tangis bayi ini dapat dilihat pada Kode Sumber 4.15.

```
library(shiny)

shinyUI(fluidPage(
  titlePanel("Aplikasi Penerjemah Tangis Bayi"),

  sidebarLayout(
    sidebarPanel(
      helpText("Pilih File Tangisan Bayi yang akan
diterjemahkan"),

      fileInput("file1","Tekan Choose untuk memilih",
                multiple = FALSE , accept=NULL, width=NULL)
      ),
```

```
    mainPanel(
      h5(strong("hasil ekstraksi fitur:")),
      textOutput("ekstraksi"),
      h5(strong("hasil klasifikasi tangis bayi:")),
      textOutput("klasifikasi")
    )
  )
))
```

Kode Sumber 4.15 Implementasi GUI Aplikasi



Gambar 4.1 GUI Aplikasi Pendeteksi Tangis Bayi

Gambar 4.2 GUI Aplikasi Pendeteksi Tangis Bayi saat Menerima *Input*

[Halaman ini sengaja dikosongkan]

BAB V
UJI COBA DAN EVALUASI

Pada bab kelima ini akan dijelaskan mengenai skenario dan uji coba perangkat lunak yang telah dibangun. Selain itu, hasil uji coba akan dievaluasi kinerjanya sehingga dapat diputuskan apakah perangkat lunak ini mampu menyelesaikan permasalahan yang telah dirumuskan di awal.

5.1 Lingkungan Uji Coba

Sebelumnya, perlu diketahui lingkungan uji coba, baik perangkat keras maupun perangkat lunak yang digunakan pada uji coba Tugas Akhir ini. Lingkungan tersebut ditunjukkan pada Tabel 5.1 berikut ini.

Tabel 5.1 Spesifikasi Lingkungan Uji Coba

Perangkat	Spesifikasi
Perangkat keras	Prosesor: Intel® Core™ i5 M460 CPU @ 2.53 GHz RAM: 8.00 GB
Perangkat lunak	Sistem Operasi: Microsoft Windows Embedded Industry Pro 8.1 64-bit Perangkat Pengembang: Rstudio dengan R versi 3.3.1 *Package* tuneR, caret, e1071, dan neuralnet untuk bahasa pemrograman R Perangkat Pembantu: Microsoft Excel 2013 Adobe Premiere Pro CS6 Adobe Audition CS5.5

5.2 Data Uji Coba

Data uji coba yang digunakan pada Tugas Akhir ini adalah rekaman suara tangis bayi yang didapatkan dari video

tangis bayi versi *Dunstan Baby Language* yang telah dipotong-potong berdasarkan durasi waktu tertentu (maksimal 10 detik) dan dikonversi menjadi *file* audio dengan format ".wav", *channel* audio *mono*, dan *sample rate* 11205 Hz menggunakan *tools* Adobe Premiere Pro CS6. Video tersebut didapatkan dari penelitian [1] yang terdiri atas lima jenis tangis bayi ("Eairh", "Eh", "Heh", "Neh", dan "Owh") yang nantinya akan dikelompokkan menjadi empat kelas yaitu kelas 1 "Eairh/Eh" (bayi ingin bersendawa, kentut, atau buang air besar), kelas 2 "Heh" (bayi merasa tidak nyawan, waktunya ganti popok), kelas 3 "Neh" (bayi merasa lapar), dan kelas 4 "Owh" (bayi mulai mengantuk). Hasil potongan video berupa *file* audio disempurnakan dengan menghilangkan gelombang audio di bagian awal yang bernilai nol menggunakan bantuan *tools* Adobe Audition CS5.5. Detail jumlah data uji coba masing-masing tangis bayi dapat dilihat pada Tabel 3.1.

5.3 Skenario Uji Coba Pemilihan Metode

Sebelum menentukan metode yang akan diterapkan pada aplikasi, perlu ditentukan skenario yang akan digunakan dalam uji coba pemilihan metode. Melalui skenario ini, kombinasi parameter dengan akurasi rata-rata tertinggi akan diterapkan pada perangkat lunak.

Uji coba dilakukan dengan mengklasifikasi hasil ekstraksi fitur yang sudah dinormalisasi pada empat jenis tangis bayi. Hasil dari ekstraksi fitur yang sudah dinormalisasi adalah sebuah *data frame* dataset yang kemudian diklasifikasi dengan *K-nearest Neighbor* untuk menentukan akurasi pada masing-masing kelas. Terdapat 15 macam skenario dari uji coba dengan menggunakan kombinasi beberapa parameter. Berikut ini merupakan penjelasan kombinasi parameter yang digunakan pada 15 skenario uji coba.

- **Parameter *wintime***

  Parameter *wintime* merupakan kombinasi panjang *window* dalam detik pada sebuah *file* audio yang akan diproses melalui ekstraksi fitur MFCC. Panjang *window* menentukan

pola hasil ekstraksi fitur pada masing-masing jenis suara tangis bayi. Kombinasi nilai yang digunakan pada panjang *window* ini adalah 0,02 sampai 0,08 dengan kenaikan 0,02 pada setiap iterasi percobaan.

- **Parameter Proporsi Data Latih dan Data Uji**
  Parameter Proporsi Data Latih dan Data Uji merupakan kombinasi persentase data uji dan data latih pada masing-masing kelas. Kombinasi ini terdiri dari 70% data latih 30% data uji hingga 90% data latih dan 10% data uji dengan kenaikan 5% data latih dan penurunan 5% data uji pada setiap iterasi percobaan. Detail jumlah proporsi data latih dan data uji dapat dilihat pada Tabel 4.1 dan Tabel 4.2.
- **Parameter Normalisasi Hasil Ekstraksi Fitur**
  Parameter Normalisasi Hasil Ekstraksi Fitur merupakan kombinasi pemakaian metode normalisasi pada hasil ekstraksi fitur MFCC. Kombinasi metode normalisasi yang digunakan adalah *Decimal Normalization*, *Min-Max Normalization*, dan *Standard Deviation Normalization* pada setiap iterasi percobaan.
- **Parameter Nilai *k* pada *K-nearest Neighbor***
  Parameter Nilai *k* pada *K-nearest Neighbor* merupakan kombinasi penggunaan nilai *k* pada *K-nearest Neighbor*. Nilai yang digunakan ialah 1 sampai 5 dengan kenaikan 1 pada setiap iterasi percobaan.

Uji coba yang dilakukan dengan mengombinasikan empat parameter di atas akan menghasilkan akurasi yang berbeda-beda. Pada setiap skenario uji coba ini, akurasi tiap kelas (5.1) dan akurasi rata-rata tiap hasil uji (5.2). Detail pengujian terdapat pada Skenario 1-15. Penggunaan masing-masing fitur pada pengujian ini terlampir pada lapiran sub bab 7.1-7.5.

$$Akurasi\ tiap\ kelas = \frac{\sum data\ benar\ pada\ tiap\ kelas}{\sum data\ uji\ pada\ tiap\ kelas} \quad \textbf{(5.1)}$$

$$Akurasi\ rata-rata = \frac{\sum akurasi\ tiap\ kelas}{\sum kelas} \quad \textbf{(5.2)}$$

5.3.1 Skenario 1

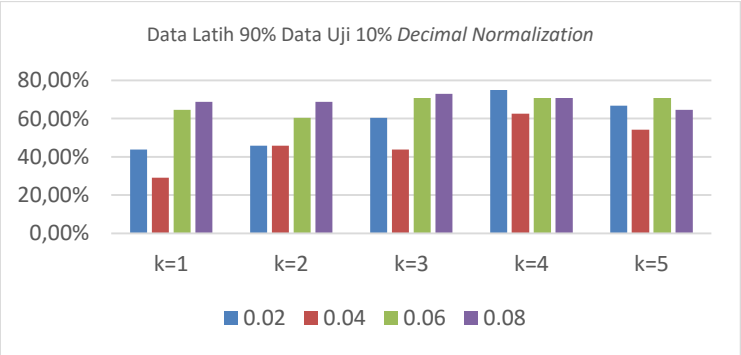
Skenario 1 adalah uji coba yang membandingkan hasil perhitungan performa klasifikasi empat kelas data tangis bayi dengan parameter sebagai berikut:

- *wintime* 0,02 sampai 0,08.
- Proporsi data latih 90% dan data uji 10%.
- Normalisasi hasil ekstraksi fitur dengan *Decimal Normalization*.
- Nilai *k* sebesar 1 sampai 5 pada *K-nearest Neighbor*.

Berdasarkan skenario 1, rata-rata hasil uji terbaik diperoleh sebesar 75% ketika *wintime* 0,02 dan nilai *k*=4 pada *K-nearrest Neighbor*. Hasil uji akurasi rata-rata dari empat percobaan untuk masing-masing fitur kasus empat kelas terdapat pada Tabel 5.2 dan Gambar 5.1.

Tabel 5.2 Rata-rata Hasil Uji Data Latih 90% Data Uji 10% *Decimal Normalization*

wintime	*k*=1	*k*=2	*k*=3	*k*=4	*k*=5
0,02	43,75%	45,83%	60,42%	**75,00%**	66,67%
0,04	29,17%	45,83%	43,75%	62,50%	54,17%
0,06	64,58%	60,42%	70,83%	70,83%	70,83%
0,08	68,75%	68,75%	72,92%	70,83%	64,58%



Gambar 5.1 Grafik Rata-rata Hasil Uji Data Latih 90% Data Uji 10% *Decimal Normalization*

5.3.2 Skenario 2

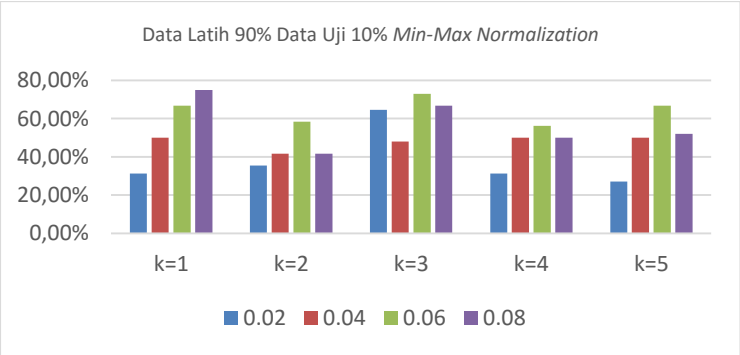
Skenario 2 adalah uji coba yang membandingkan hasil perhitungan performa klasifikasi empat kelas data tangis bayi dengan parameter sebagai berikut:

- *wintime* 0,02 sampai 0,08.
- Proporsi data latih 90% dan data uji 10%.
- Normalisasi hasil ekstraksi fitur dengan *Min-Max Normalization*.
- Nilai *k* sebesar 1 sampai 5 pada *K-nearest Neighbor*.

Berdasarkan skenario 2, rata-rata hasil uji terbaik diperoleh sebesar 75% ketika *wintime* 0,08 dan nilai *k*=1 pada *K-nearrest Neighbor*. Hasil uji akurasi rata-rata dari empat percobaan untuk masing-masing fitur kasus empat kelas terdapat pada Tabel 5.3 dan Gambar 5.2.

Tabel 5.3 Rata-rata Hasil Uji Data Latih 90% Data Uji 10% *Min-Max Normalization*

wintime	*k*=1	*k*=2	*k*=3	*k*=4	*k*=5
0,02	31,25%	35,42%	64,58%	31,25%	27,08%
0,04	50,00%	41,67%	47,92%	50,00%	50,00%
0,06	66,67%	58,33%	72,92%	56,25%	66,67%
0,08	**75,00%**	41,67%	66,67%	50,00%	52,08%



Gambar 5.2 Rata-rata Hasil Uji Data Latih 90% Data Uji 10% *Min-Max Normalization*

5.3.3 Skenario 3

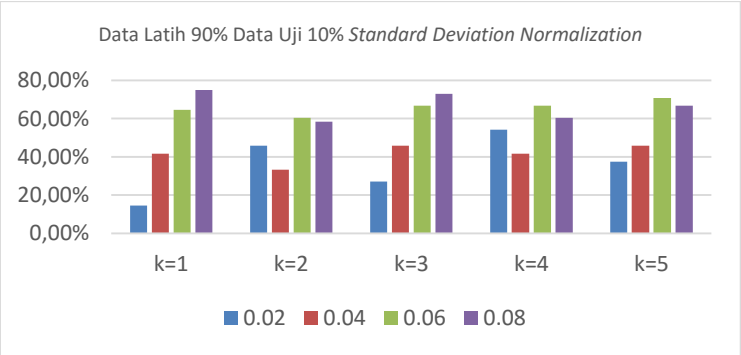
Skenario 3 adalah uji coba yang membandingkan hasil perhitungan performa klasifikasi empat kelas data tangis bayi dengan parameter sebagai berikut:

- *wintime* 0,02 sampai 0,08.
- Proporsi data latih 90% dan data uji 10%.
- Normalisasi hasil ekstraksi fitur dengan *Standard Deviation Normalization*.
- Nilai *k* sebesar 1 sampai 5 pada *K-nearest Neighbor*.

Berdasarkan skenario 3, rata-rata hasil uji terbaik diperoleh sebesar 75% ketika *wintime* 0,08 dan nilai *k*=1 pada *K-nearrest Neighbor*. Hasil uji akurasi rata-rata dari empat percobaan untuk masing-masing fitur kasus empat kelas terdapat pada Tabel 5.4 dan Gambar 5.3.

Tabel 5.4 Rata-rata Hasil Uji Data Latih 90% Data Uji 10% *Standard Deviation Normalization*

wintime	*k*=1	*k*=2	*k*=3	*k*=4	*k*=5
0,02	14,58%	45,83%	27,08%	54,17%	37,50%
0,04	41,67%	33,33%	45,83%	41,67%	45,83%
0,06	64,58%	60,42%	66,67%	66,67%	70,83%
0,08	**75,00%**	58,33%	72,92%	60,42%	66,67%



Gambar 5.3 Grafik Rata-rata Hasil Uji Data Latih 90% Data Uji 10% *Standard Deviation Normalization*

5.3.4 Skenario 4

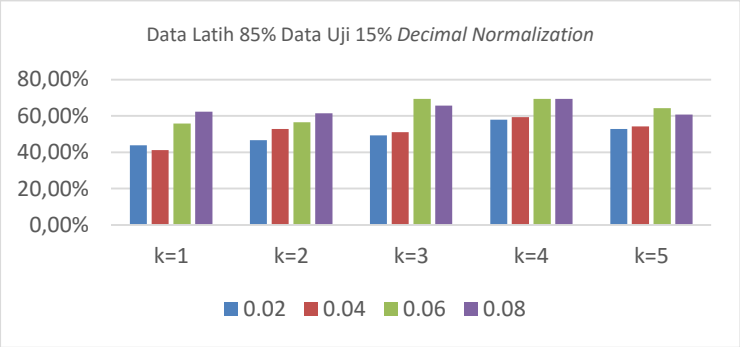
Skenario 4 adalah uji coba yang membandingkan hasil perhitungan performa klasifikasi empat kelas data tangis bayi dengan parameter sebagai berikut:

- *wintime* 0,02 sampai 0,08.
- Proporsi data latih 85% dan data uji 15%.
- Normalisasi hasil ekstraksi fitur dengan *Decimal Normalization*.
- Nilai *k* sebesar 1 sampai 5 pada *K-nearest Neighbor*.

Berdasarkan skenario 4, rata-rata hasil uji terbaik diperoleh sebesar 69,29% ketika *wintime* 0,06 dengan nilai *k*=3 dan *k*=4 dan *wintime* 0,08 dengan nilai *k*=4 pada *K-nearrest Neighbor*. Hasil uji akurasi rata-rata dari empat percobaan untuk masing-masing fitur kasus empat kelas terdapat pada Tabel 5.5 dan Gambar 5.4.

Tabel 5.5 Rata-rata Hasil Uji Data Latih 85% Data Uji 15% *Decimal Normalization*

wintime	*k*=1	*k*=2	*k*=3	*k*=4	*k*=5
0,02	43,81%	46,67%	49,29%	57,86%	52,86%
0,04	41,19%	52,86%	50,95%	59,29%	54,29%
0,06	55,71%	56,43%	69,29%	**69,29%**	64,29%
0,08	62,38%	61,43%	65,71%	**69,29%**	60,71%



Gambar 5.4 Grafik Rata-rata Hasil Uji Data Latih 85% Data Uji 15% *Decimal Normalization*

5.3.5 Skenario 5

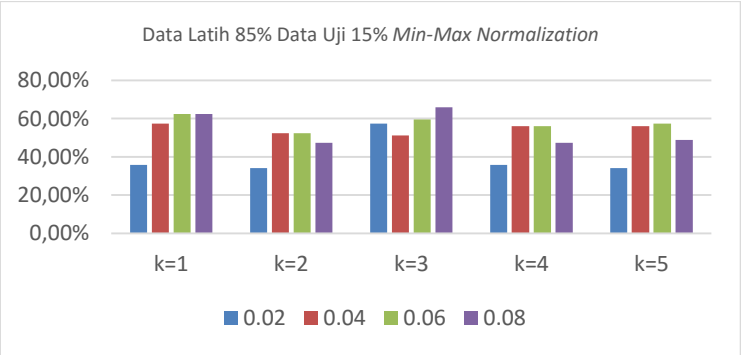
Skenario 5 adalah uji coba yang membandingkan hasil perhitungan performa klasifikasi empat kelas data tangis bayi dengan parameter sebagai berikut:

- *wintime* 0,02 sampai 0,08.
- Proporsi data latih 85% dan data uji 15%.
- Normalisasi hasil ekstraksi fitur dengan *Min-Max Normalization*.
- Nilai *k* sebesar 1 sampai 5 pada *K-nearest Neighbor*.

Berdasarkan skenario 5, rata-rata hasil uji terbaik diperoleh sebesar 65,95% ketika *wintime* 0,08 dengan nilai *k*=3 *K-nearrest Neighbor*. Hasil uji akurasi rata-rata dari empat percobaan untuk masing-masing fitur kasus empat kelas terdapat pada Tabel 5.6 dan Gambar 5.5.

Tabel 5.6 Rata-rata Hasil Uji Data Latih 85% Data Uji 15% *Min-Max Normalization*

wintime	*k*=1	*k*=2	*k*=3	*k*=4	*k*=5
0,02	35,71%	34,05%	57,38%	35,71%	34,05%
0,04	57,38%	52,38%	51,19%	55,95%	55,95%
0,06	62,38%	52,38%	59,52%	55,95%	57,38%
0,08	62,38%	47,38%	**65,95%**	47,38%	48,81%



Gambar 5.5 Grafik Rata-rata Hasil Uji Data Latih 85% Data Uji 15% *Min-Max Normalization.*

5.3.6 Skenario 6

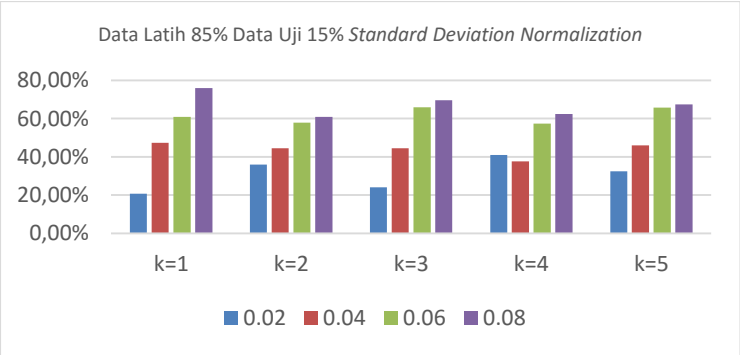
Skenario 6 adalah uji coba yang membandingkan hasil perhitungan performa klasifikasi empat kelas data tangis bayi dengan parameter sebagai berikut:

- *wintime* 0,02 sampai 0,08.
- Proporsi data latih 85% dan data uji 15%.
- Normalisasi hasil ekstraksi fitur dengan *Standard Deviation Normalization*.
- Nilai *k* sebesar 1 sampai 5 pada *K-nearest Neighbor*.

Berdasarkan skenario 6, rata-rata hasil uji terbaik diperoleh sebesar 75,95% ketika *wintime* 0,08 dengan nilai *k*=1 *K-nearrest Neighbor*. Hasil uji akurasi rata-rata dari empat percobaan untuk masing-masing fitur kasus empat kelas terdapat pada Tabel 5.7 dan Gambar 5.6.

Tabel 5.7 Rata-rata Hasil Uji Data Latih 85% Data Uji 15% *Standard Deviation Normalization*

wintime	*k*=1	*k*=2	*k*=3	*k*=4	*k*=5
0,02	20,71%	35,95%	24,05%	40,95%	32,38%
0,04	47,38%	44,52%	44,52%	37,62%	45,95%
0,06	60,95%	57,86%	65,95%	57,38%	65,71%
0,08	**75,95%**	60,95%	69,52%	62,38%	67,38%



Gambar 5.6 Grafik Rata-rata Hasil Uji Data Latih 85% Data Uji 15% *Standard Deviation Normalization*

5.3.7 Skenario 7

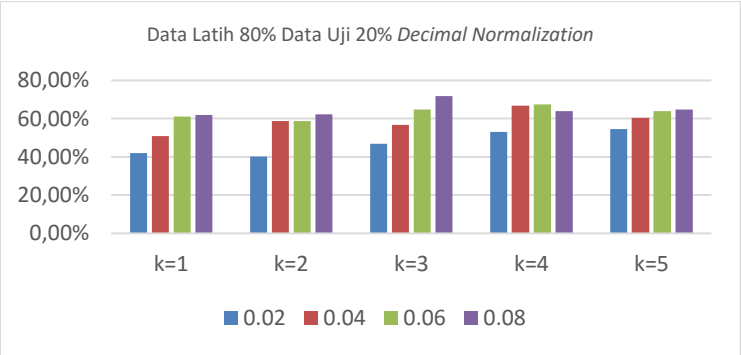
Skenario 7 adalah uji coba yang membandingkan hasil perhitungan performa klasifikasi empat kelas data tangis bayi dengan parameter sebagai berikut:

- *wintime* 0,02 sampai 0,08.
- Proporsi data latih 80% dan data uji 20%.
- Normalisasi hasil ekstraksi fitur dengan *Decimal Normalization*.
- Nilai *k* sebesar 1 sampai 5 pada *K-nearest Neighbor*.

Berdasarkan skenario 7, rata-rata hasil uji terbaik diperoleh sebesar 71,83% ketika *wintime* 0,08 dengan nilai *k*=3 *K-nearrest Neighbor*. Hasil uji akurasi rata-rata dari empat percobaan untuk masing-masing fitur kasus empat kelas terdapat pada Tabel 5.8 dan Gambar 5.7.

Tabel 5.8 Rata-rata Hasil Uji Data Latih 80% Data Uji 20% *Decimal Normalization*

wintime	*k*=1	*k*=2	*k*=3	*k*=4	*k*=5
0,02	41,98%	40,16%	46,75%	53,10%	54,52%
0,04	50,87%	58,73%	56,67%	66,67%	60,32%
0,06	61,11%	58,73%	64,68%	67,46%	63,89%
0,08	61,83%	62,30%	**71,83%**	63,89%	64,68%



Gambar 5.7 Grafik Rata-rata Hasil Uji Data Latih 80% Data Uji 20% *Decimal Normalization*

5.3.8 Skenario 8

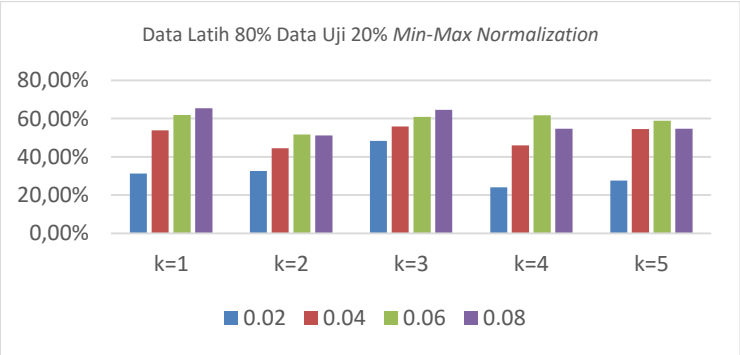
Skenario 8 adalah uji coba yang membandingkan hasil perhitungan performa klasifikasi empat kelas data tangis bayi dengan parameter sebagai berikut:

- *wintime* 0,02 sampai 0,08.
- Proporsi data latih 80% dan data uji 20%.
- Normalisasi hasil ekstraksi fitur dengan *Min-Max Normalization*.
- Nilai *k* sebesar 1 sampai 5 pada *K-nearest Neighbor*.

Berdasarkan skenario 8, rata-rata hasil uji terbaik diperoleh sebesar 65,40% ketika *wintime* 0,08 dengan nilai *k*=1 *K-nearrest Neighbor*. Hasil uji akurasi rata-rata dari empat percobaan untuk masing-masing fitur kasus empat kelas terdapat pada Tabel 5.9 dan Gambar 5.8.

Tabel 5.9 Rata-rata Hasil Uji Data Latih 80% Data Uji 20% *Min-Max Normalization*

wintime	*k*=1	*k*=2	*k*=3	*k*=4	*k*=5
0,02	31,19%	32,62%	48,33%	24,05%	27,62%
0,04	53,89%	44,52%	55,87%	45,95%	54,52%
0,06	61,83%	51,75%	60,87%	61,67%	58,89%
0,08	**65,40%**	51,11%	64,60%	54,68%	54,68%



Gambar 5.8 Grafik Rata-rata Hasil Uji Data Latih 80% Data Uji 20% *Min-Max Normalization*

5.3.9 Skenario 9

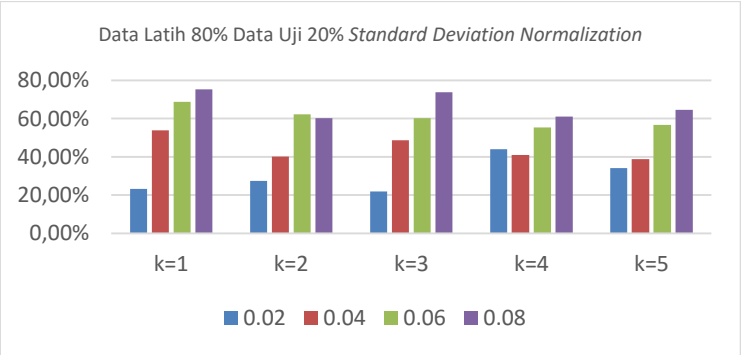
Skenario 9 adalah uji coba yang membandingkan hasil perhitungan performa klasifikasi empat kelas data tangis bayi dengan parameter sebagai berikut:

- *wintime* 0,02 sampai 0,08.
- Proporsi data latih 80% dan data uji 20%.
- Normalisasi hasil ekstraksi fitur dengan *Standard Deviation Normalization*.
- Nilai *k* sebesar 1 sampai 5 pada *K-nearest Neighbor*.

Berdasarkan skenario 9, rata-rata hasil uji terbaik diperoleh sebesar 75,32% ketika *wintime* 0,08 dengan nilai *k*=1 *K-nearrest Neighbor*. Hasil uji akurasi rata-rata dari empat percobaan untuk masing-masing fitur kasus empat kelas terdapat pada Tabel 5.10 dan Gambar 5.9.

Tabel 5.10 Rata-rata Hasil Uji Data Latih 80% Data Uji 20% *Standard Deviation Normalization*

wintime	*k*=1	*k*=2	*k*=3	*k*=4	*k*=5
0,02	23,25%	27,46%	21,90%	43,97%	34,05%
0,04	53,89%	40,16%	48,73%	40,95%	38,81%
0,06	68,81%	62,30%	60,24%	55,32%	56,75%
0,08	**75,32%**	60,24%	73,73%	61,03%	64,60%



Gambar 5.9 Rata-rata Grafik Hasil Uji Data Latih 80% Data Uji 20% *Standard Deviation Normalization*

5.3.10 Skenario 10

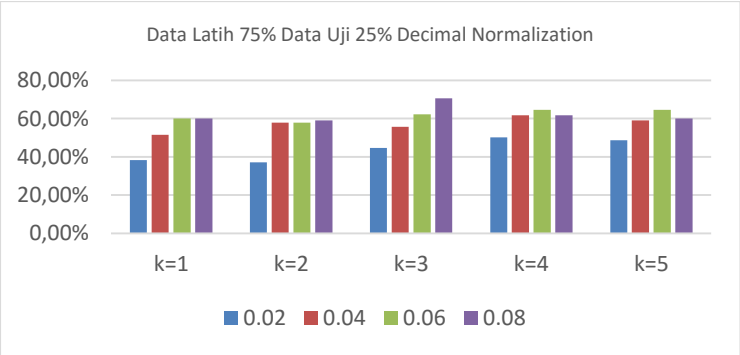
Skenario 10 adalah uji coba yang membandingkan hasil perhitungan performa klasifikasi empat kelas data tangis bayi dengan parameter sebagai berikut:

- *wintime* 0,02 sampai 0,08.
- Proporsi data latih 75% dan data uji 25%.
- Normalisasi hasil ekstraksi fitur dengan *Decimal Normalization*.
- Nilai *k* sebesar 1 sampai 5 pada *K-nearest Neighbor*.

Berdasarkan skenario 10, rata-rata hasil uji terbaik diperoleh sebesar 70,58% ketika *wintime* 0,08 dengan nilai *k*=3 *K-nearrest Neighbor*. Hasil uji akurasi rata-rata dari empat percobaan untuk masing-masing fitur kasus empat kelas terdapat pada Tabel 5.11 dan Gambar 5.10.

Tabel 5.11 Rata-rata Hasil Uji Data Latih 75% Data Uji 25% *Decimal Normalization*

wintime	*k*=1	*k*=2	*k*=3	*k*=4	*k*=5
0,02	38,26%	37,12%	44,57%	50,13%	48,74%
0,04	51,52%	57,95%	55,68%	61,74%	58,96%
0,06	59,97%	57,95%	62,25%	64,52%	64,52%
0,08	59,97%	58,96%	**70,58%**	61,74%	59,97%



Gambar 5.10 Grafik Hasil Uji Data Latih 75% Data Uji 25% *Decimal Normalization*

5.3.11 Skenario 11

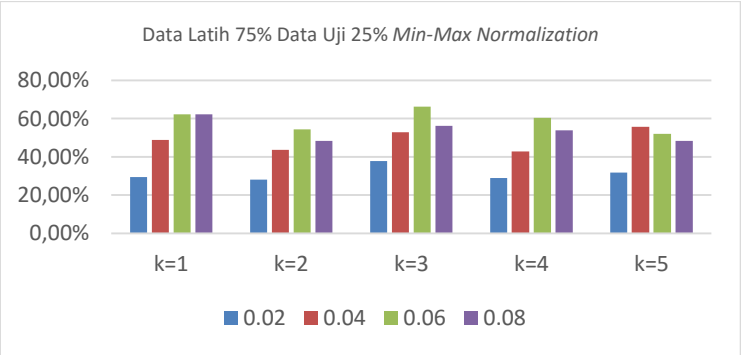
Skenario 11 adalah uji coba yang membandingkan hasil perhitungan performa klasifikasi empat kelas data tangis bayi dengan parameter sebagai berikut:

- *wintime* 0,02 sampai 0,08.
- Proporsi data latih 75% dan data uji 25%.
- Normalisasi hasil ekstraksi fitur dengan *Min-Max Normalization*.
- Nilai *k* sebesar 1 sampai 5 pada *K-nearest Neighbor*.

Berdasarkan skenario 11, rata-rata hasil uji terbaik diperoleh sebesar 66,29% ketika *wintime* 0,06 dengan nilai *k*=3 *K-nearrest Neighbor*. Hasil uji akurasi rata-rata dari empat percobaan untuk masing-masing fitur kasus empat kelas terdapat pada Tabel 5.12 dan Gambar 5.11.

Tabel 5.12 Rata-rata Hasil Uji Data Latih 75% Data Uji 25% *Min-Max Normalization*

wintime	*k*=1	*k*=2	*k*=3	*k*=4	*k*=5
0,02	29,42%	28,03%	37,75%	28,91%	31,69%
0,04	48,86%	43,69%	52,90%	42,80%	55,68%
0,06	62,25%	54,29%	**66,29%**	60,35%	52,02%
0,08	62,25%	48,36%	56,19%	53,91%	48,36%



Gambar 5.11 Grafik Rata-rata Hasil Uji Data Latih 75% Data Uji 25% *Min-Max Normalization*

5.3.12 Skenario 12

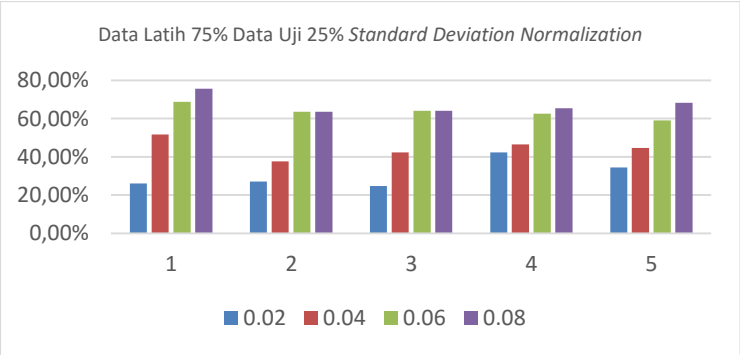
Skenario 12 adalah uji coba yang membandingkan hasil perhitungan performa klasifikasi empat kelas data tangis bayi dengan parameter sebagai berikut:

- *wintime* 0,02 sampai 0,08.
- Proporsi data latih 75% dan data uji 25%.
- Normalisasi hasil ekstraksi fitur dengan *Standard Deviation Normalization*.
- Nilai *k* sebesar 1 sampai 5 pada *K-nearest Neighbor*.

Berdasarkan skenario 12, rata-rata hasil uji terbaik diperoleh sebesar 75,63% ketika *wintime* 0,08 dengan nilai *k*=1 *K-nearrest Neighbor*. Hasil uji akurasi rata-rata dari empat percobaan untuk masing-masing fitur kasus empat kelas terdapat pada Tabel 5.13 dan Gambar 5.12.

Tabel 5.13 Rata-rata Hasil Uji Data Latih 75% Data Uji 25% *Standard Deviation Normalization*

wintime	*k*=1	*k*=2	*k*=3	*k*=4	*k*=5
0,02	26,14%	27,02%	24,75%	42,30%	34,47%
0,04	51,64%	37,63%	42,30%	46,46%	44,57%
0,06	68,69%	63,51%	64,02%	62,63%	58,96%
0,08	**75,63%**	63,51%	64,02%	65,40%	68,18%



Gambar 5.12 Grafik Rata-rata Hasil Uji Data Latih 75% Data Uji 25% *Standard Deviation Normalization*

5.3.13 Skenario 13

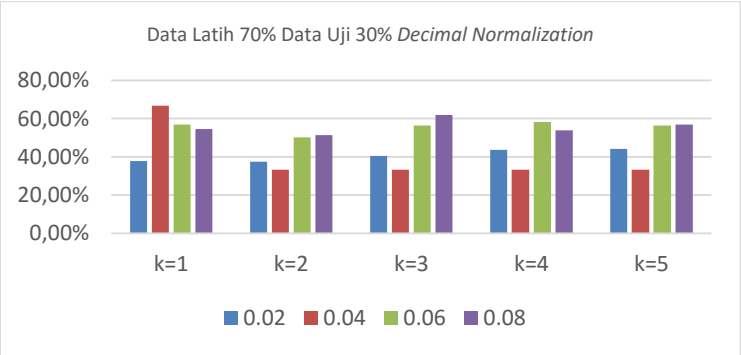
Skenario 13 adalah uji coba yang membandingkan hasil perhitungan performa klasifikasi empat kelas data tangis bayi dengan parameter sebagai berikut:

- *wintime* 0,02 sampai 0,08.
- Proporsi data latih 70% dan data uji 30%.
- Normalisasi hasil ekstraksi fitur dengan *Decimal Normalization*.
- Nilai *k* sebesar 1 sampai 5 pada *K-nearest Neighbor*.

Berdasarkan skenario 13, rata-rata hasil uji terbaik diperoleh sebesar 66,67% ketika *wintime* 0,04 dengan nilai *k*=1 *K-nearrest Neighbor*. Hasil uji akurasi rata-rata dari empat percobaan untuk masing-masing fitur kasus empat kelas terdapat pada Tabel 5.14 dan Gambar 5.13Gambar 2.1.

Tabel 5.14 Rata-rata Hasil Uji Data Latih 70% Data Uji 30% *Decimal Normalization*

wintime	*k*=1	*k*=2	*k*=3	*k*=4	*k*=5
0,02	37,83%	37,45%	40,52%	43,60%	44,09%
0,04	**66,67%**	33,33%	33,33%	33,33%	33,33%
0,06	56,90%	50,16%	56,32%	58,24%	56,32%
0,08	54,48%	51,32%	61,90%	53,82%	56,90%



Gambar 5.13 Grafik Rata-rata Hasil Uji Data Latih 70% Data Uji 30% *Decimal Normalization*

5.3.14 Skenario 14

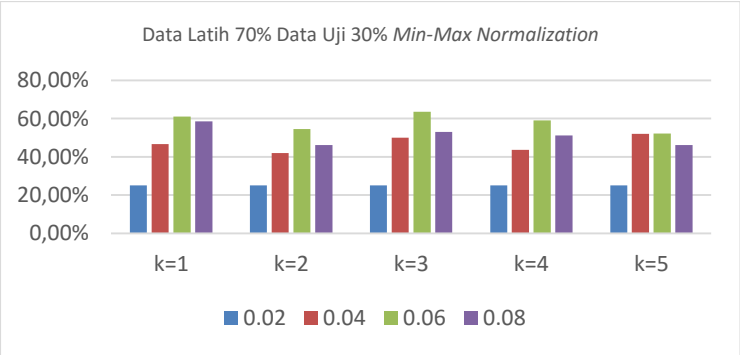
Skenario 14 adalah uji coba yang membandingkan hasil perhitungan performa klasifikasi empat kelas data tangis bayi dengan parameter sebagai berikut:

- *wintime* 0,02 sampai 0,08.
- Proporsi data latih 70% dan data uji 30%.
- Normalisasi hasil ekstraksi fitur dengan *Min-Max Normalization*.
- Nilai *k* sebesar 1 sampai 5 pada *K-nearest Neighbor*.

Berdasarkan skenario 14, rata-rata hasil uji terbaik diperoleh sebesar 63,52% ketika *wintime* 0,06 dengan nilai *k*=3 *K-nearrest Neighbor*. Hasil uji akurasi rata-rata dari empat percobaan untuk masing-masing fitur kasus empat kelas terdapat pada Tabel 5.15 dan Gambar 5.14.

Tabel 5.15 Rata-rata Hasil Uji Data Latih 70% Data Uji 30% *Min-Max Normalization*

wintime	*k*=1	*k*=2	*k*=3	*k*=4	*k*=5
0,02	25,00%	25,00%	25,00%	25,00%	25,00%
0,04	46,68%	41,95%	49,95%	43,60%	52,09%
0,06	61,10%	54,59%	**63,52%**	59,09%	52,17%
0,08	58,60%	46,10%	53,02%	51,10%	46,10%



Gambar 5.14 Grafik Rata-rata Hasil Uji Data Latih 70% Data Uji 30% *Min-Max Normalization*

5.3.15 Skenario 15

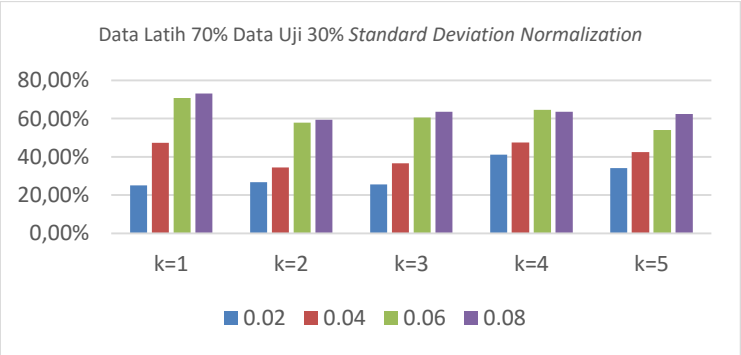
Skenario 15 adalah uji coba yang membandingkan hasil perhitungan performa klasifikasi empat kelas data tangis bayi dengan parameter sebagai berikut:

- *wintime* 0,02 sampai 0,08.
- Proporsi data latih 70% dan data uji 30%.
- Normalisasi hasil ekstraksi fitur dengan *Standard Deviation Normalization*.
- Nilai *k* sebesar 1 sampai 5 pada *K-nearest Neighbor*.

Berdasarkan skenario 15, rata-rata hasil uji terbaik diperoleh sebesar 73,02% ketika *wintime* 0,08 dengan nilai *k*=1 *K-nearrest Neighbor*. Hasil uji akurasi rata-rata dari empat percobaan untuk masing-masing fitur kasus empat kelas terdapat pada Tabel 5.16 dan Gambar 5.15.

Tabel 5.16 Rata-rata Hasil Uji Data Latih 70% Data Uji 30% *Standard Deviation Normalization*

wintime	*k*=1	*k*=2	*k*=3	*k*=4	*k*=5
0,02	25,11%	26,68%	25,60%	41,10%	34,18%
0,04	47,25%	34,45%	36,68%	47,53%	42,45%
0,06	70,74%	57,94%	60,52%	64,59%	54,09%
0,08	**73,02%**	59,37%	63,52%	63,52%	62,45%



Gambar 5.15 Grafik Rata-rata Hasil Uji Data Latih 70% Data Uji 30% *Standard Deviation Normalization*

5.4 Skenario Uji Coba dengan Data Latih Seimbang pada Tiap Kelas

Setelah mendapatkan kombinasi parameter terbaik, pengujian dilakukan dengan menggunakan parameter *wintime* sebesar 0,08 detik, *K-nearest Neighbor* dengan *k*=1 sampai *k*=5, normalisasi hasil ekstraksi fitur dengan *Standard Deviation Normalization*, dan data latih seimbang pada tiap kelas dilakukan untuk menganalisis kecenderungan salah deteksi dari jenis tangis satu ke jenis tangis lainnya. Data latih pada masing-masing kelas jenis tangis bayi akan diatur sebanyak 19, kemudian sisa data yang lain akan digunakan sebagai data uji. Proporsi data latih dan data uji yang digunakan terdapat pada Tabel 5.17. Akurasi hasil uji coba dengan data latih seimbang dapat dilihat pada Tabel 5.18. Rekap hasil uji dan *confusion matrix* hasil pengujian *K-nearest Neighbor k*=1 sampai *k*=5 dengan data latih seimbang dari tiap kelas dapat dilihat pada lampiran Tabel 7.62 sampai Tabel 7.67.

Tabel 5.17 Tabel Pembagian Data Latih Seimbang pada Tiap Kelas

Jenis Tangis	Jumlah Data Keseluruhan	Data Latih	Data Uji
Eairh/Eh	46	19	27
Heh	23	19	4
Neh	35	19	16
Owh	35	19	16

Tabel 5.18 Perbedaan Hasil Klasifikasi KNN dengan Variasi Nilai *k* pada Data Latih Seimbang

Jenis Tangis	Klasifikasi KNN				
	k=1	k=2	k=3	k=4	k=5
Eairh/Eh	51,85%	70,37%	55,56%	51,85%	55,56%
Heh	75,00%	75,00%	100,00%	100,00%	75,00%
Neh	75,00%	50,00%	56,25%	62,50%	56,25%
Owh	56,25%	31,25%	43,75%	37,50%	37,50%
Akurasi rata-rata	**64,53%**	56,66%	63,89%	62,96%	56,08%

5.5 Skenario Uji Coba dengan Metode Klasifikasi Lain

Setelah mendapatkan kombinasi parameter terbaik, uji coba dilakukan dengan menggunakan metode klasifikasi selain *K-nearest Neighbor* dengan *k*=1 untuk mengetahui perbandingan akurasi dari metode lain. Metode klasifikasi yang akan dibandingkan ialah *Naive Bayes*, *Support Vector Machine,* dan *Neural Network*. Pada klasifikasi *Support Vector Machine*, dengan metode *One-againts-one*, kernel yang digunakan ialah *polynomial* dan tidak menggunakan *cross validation*. Pada metode klasifikasi *Neural Network*, jumlah *hidden layer* yang digunakan sebanyak dua dengan 7 dan 5 node pada masing-masing *layer*, fungsi aktivasi yang digunakan ialah fungsi logistik, algoritma *training* yang digunakan ialah *resilient backpropagation* dengan *weight backtracking*, dan koefisien *learning rate* yang digunakan sebesar 0,02. Proporsi data latih dan data uji yang digunakan pada tiga metode pembanding ialah 85% data latih dan 15% data uji dari masing-masing kelas. Nilai *wintime* yang digunakan pada ekstraksi fitur MFCC dari *package* tuneR sebesar 0,08 detik. Normalisasi hasil ekstraksi fitur yang digunakan ialah *Standard Deviation Normalization*. Hasil uji perbandingan akurasi per kelas dan akurasi rata-rata pada tiga metode klasifikasi selain *K-nearest Neighbor* dengan *k*=1 dapat dilihat pada Tabel 5.19. Rekap hasil uji perbandingan tiga metode klasifikasi ini dapat dilihat pada lampiran Tabel 7.68 dan *confusion matrix* Tabel 7.69 sampai Tabel 7.71.

Tabel 5.19 Perbandaingan Akurasi Metode Klasifikasi

Jenis Tangis	Metode Klasifikasi			
	KNN	Naive Bayes	Neural Network	SVM
Eairh/Eh	57,14%	28,57%	42,86%	42,86%
Heh	66,67%	100,00%	100,00%	66,67%
Neh	80,00%	20,00%	40,00%	60,00%
Owh	100,00%	80,00%	80,00%	100,00%
Rata-Rata Akurasi	**75,95%**	57,14%	65,71%	67,38%

5.6 Skenario Uji Coba Aplikasi Pendeteksi Suara Tangis Bayi

Setelah mendapatkan metode terbaik dari berbagai uji coba pemilihan metode yang akan diterapkan pada aplikasi dan aplikasi telah dibuat, uji coba aplikasi dilakukan untuk menguji apakah aplikasi mampu berjalan dengan baik dalam melakukan klasifikasi suara tangis bayi. Uji coba aplikasi dilakukan dengan menerapkan parameter berikut ini:

- *wintime* = 0,08.
- data latih 85% dari setiap kelas jenis tangis bayi.
- nilai *k*=1 pada *K-nearest Neighbor*.
- normalisasi hasil ekstraksi fitur dengan *Standard Deviation Normalization*.

Sebanyak 139 audio tangis bayi diunggah ke aplikasi penerjemah tangis bayi secara satu per satu. Rekap hasil pengujian aplikasi dapat dilihat pada lampiran Tabel 7.72. Hasil pengujian aplikasi dan akurasi tiap kelas dapat dilihat pada Tabel 5.20 dan Tabel 5.21. Apliksi dapat dikunjungi pada *link* https://liemwellys.shinyapps.io/Aplikasi_Penerjemah_Tangis_Bayi/.

Tabel 5.20 *Confusion Matrix* Uji Coba Aplikasi Pendeteksi Tangis Bayi

Hasil Prediksi	**Eairh/Eh**	**Heh**	**Neh**	**Owh**
Eairh/Eh	**43**	0	1	0
Heh	0	**22**	0	0
Neh	1	1	**34**	0
Owh	2	0	0	**35**

Tabel 5.21 Akurasi Tiap Kelas Aplikasi Pendeteksi Tangis Bayi

Eairh/Eh	**Heh**	**Neh**	**Owh**	**Akurasi Rata-Rata**
93,48%	95,65%	97,14%	100,00%	96,57%

5.7 Analisis Hasil Uji Coba Pemilihan Metode

Berdasarkan hasil uji coba pemilihan metode dari 15 skenario yang telah dilakukan, rata-rata akurasi terbaik sebesar 75,95% didapat ketika menggunakan *wintime* 0,08 dengan proporsi data latih 85% data uji 15%, nilai *k*=1 pada *K-nearest Neighbor*, dan normalisasi hasil ekstraksi fitur dengan *Standard Deviation Normalization*. Hasil uji coba pemilihan metode terbaik dapat dilihat pada *confusion matrix* Tabel 5.22 dan lampiran Tabel 7.61.

Tabel 5.22 *Confusion Matrix* Uji Coba Pemilihan Metode Terbaik dengan Parameter *wintime*=0,08 detik, Data Latih 85% Data Uji 15%, Nilai *k*=1 pada *K-nearest Neighbor*, dan Normalisasi Ekstraksi Fitur dengan *Standard Deviation Normalization*

Hasil Prediksi	Eairh/Eh	Heh	Neh	Owh
Eairh/Eh	**4**	0	1	0
Heh	0	**2**	0	0
Neh	1	1	**4**	0
Owh	2	0	0	**5**

Dari 15 skenario tersebut, parameter *wintime* merupakan parameter yang paling berpengaruh terhadap akurasi klasifikasi. Perbandingan rata-rata akurasi klasifikasi dengan kombinasi *wintime* pada 15 skenario uji coba terdapat pada Tabel 5.23. Berdasarkan tabel tersebut, dapat disimpulkan bahwa penentuan durasi *wintime* pada ekstraksi fitur MFCC dari *package* tuneR sangat berperan penting pada akurasi klasifikasi jenis suara tangis bayi. Semakin lama durasi dari audio yang akan diproses dengan ekstraksi fitur MFCC, maka penentuan *wintime* harus semakin besar agar fitur yang didapat semakin baik. Pada penelitian ini, durasi rekaman tangis bayi pada setiap kelas berkisar antara 1-10 detik, sehingga untuk mencapai akurasi klasifikasi terbaik nilai *wintime* diatur sebesar 0,08 detik agar pemotongan *window* pada *frame* audio lebih panjang. Dengan durasi pemotongan *window* yang lebih panjang, maka pola suara tangis bayi dapat dikenali lebih baik jika

dibandingkan dengan pemotongan *window* dengan durasi yang pendek.

Tabel 5.23 Akurasi Rata-rata Penggunaan Kombinasi *wintime* pada 15 skenario percobaan

Proporsi Data Uji Coba	Durasi *wintime* (dalam detik)			
	0,02	0,04	0,06	0,08
Data Latih 90% Data Uji 10%	44,03%	45,56%	65,83%	64,31%
Data Latih 85% Data Uji 15%	40,10%	50,10%	60,70%	61,84%
Data Latih 80% Data Uji 20%	36,73%	51,37%	60,95%	63,33%
Data Latih 75% Data Uji 25%	35,29%	50,16%	61,48%	61,14%
Data Latih 70% Data Uji 30%	32,08%	42,84%	58,42%	57,01%

Pada hasil uji coba pemilihan metode terbaik yang menggunakan parameter *wintime* sebesar 0,08 dengan proporsi data latih 85% data uji 15%, nilai k=1 pada *K-nearest Neighbor*, dan normalisasi hasil ekstraksi fitur dengan *Standard Deviation Normalization*, akurasi hasil klasifikasi suara "Owh" merupakan yang tertinggi, sedangkan akurasi hasil klasifikasi suara "Eairh/Eh" merupakan yang terendah. Hal ini terjadi karena jumlah data yang dipakai untuk data latih "Eairh/Eh" masih sedikit (data jenis tangis "Eairh" sebanyak 25, data jenis tangis "Eh" sebanyak 21) jika dibandingkan dengan data tangis "Owh" yang berjumlah 35. Kesalahan deteksi pada masing-masing kelas terjadi karena hasil perhitungan fungsi jarak *euclidian* dari data uji lebih dekat ke hasil perhitungan fungsi jarak *euclidian* pada kelas lain sehingga data uji tidak terklasifikasi pada kelas yang tepat. Kesalahan deteksi ini dapat

diperbaiki dengan menerapkan nilai dinamis pada parameter *wintime* saat ekstraksi fitur MFCC dari *package* tuneR agar hasil ekstraksi fitur pada data latih maupun data uji lebih beragam.

Faktor lain yang menentukan akurasi klasifikasi ialah variasi usia bayi yang suaranya dijadikan sampel. Semakin tinggi rentang usia antar bayi yang suara tangisnya dijadikan sampel pada suatu jenis tangis bayi, maka fitur yang dihasilkan dalam satu jenis tangis bayi lebih beragam, sehingga saat dilakukan klasifikasi aplikasi bisa saja salah mendeteksi tangis tersebut sebagai jenis tangis yang lain. Hal ini terjadi karena selama masa perkembangan bayi pada usia 0-3 bulan, bayi belajar mendengarkan dan berusaha untuk menirukan suara-suara yang ada di sekitarnya.

5.8 Analisis Hasil Uji Coba dengan Data Latih Seimbang pada Tiap Kelas

Berdasarkan skenario uji coba dengan data latih seimbang pada tiap kelas, akurasi rata-rata terbaik yang didapat untuk mengklasifikasikan empat jenis tangis bayi sebesar 64,53% ketika nilai k=1 pada *K-nearest Neighbor*. Akurasi tertinggi terdapat pada jenis tangis “Heh” dan “Neh”, disusul oleh “Owh” dan “Eairh/Eh”. Akurasi masing-masing kelas dapat dilihat pada Tabel 5.18.

Dari hasil uji dengan *K-nearest Neighbor* k=1, masing-masing suara tangis bayi ada yang dominan salah dideteksi sebagai jenis tangis lain. Kesalahan deteksi suara tangis “Eairh/Eh”dominan pada tangis “Owh”, suara tangis “Heh” ada satu *file* yang salah dideteksi sebagai tangis “Eairh/Eh”, suara tangis “Neh” dideteksi sebagai tangis “Eairh/Eh” dan “Heh”, dan suara tangis “Owh” dominan salah dideteksi sebagai tangis “Neh”. Kesalahan deteksi masing-masing suara tangis bayi dapat dilihat pada *confusion matrix* lampiran Tabel 7.63.

Secara garis besar, jika dibandingkan dengan pembagian proporsi data latih berdasarkan persentase data dari tiap jenis tangis bayi, akurasi yang dihasilkan jauh lebih baik karena data

latih dari masing-masing jenis tangis bayi yang digunakan jauh lebih banyak.

5.9 Analisis Hasil Uji Coba dengan Metode Klasifikasi Lain

Berdasarkan skenario uji coba perbandaingan tiga metode klasifikasi selain *K-nearest Neighbor* dengan *k*=1 pada tiap jenis tangis bayi, rata-rata akurasi klasifikasi terbaik sebesar 75,95% didapat ketika menggunakan metode klasifikasi *K-nearest Neighbor* dengan *k*=1. Rata-rata hasil klasifikasi terendah sebesar 57,14% terjadi ketika menggunakan metode klasifikasi *Naive Bayes*. Metode klasifikasi *Naive Bayes* mendapatkan akurasi terendah karena hanya dapat mengklasifikasikan sebagian data uji pada jenis tangis "Eairh/Eh" dan "Neh".

Dari uji coba tersebut, ada jenis tangis tertentu yang seluruh data ujinya dapat dideteksi benar 100%. Hal tersebut terjadi pada jenis tangis "Heh" ketika menggunakan metode klasifikasi *Naive Bayes* dan *Neural Network* serta jenis tangis Owh ketika menggunakan metode klasifikasi *Support Vector Machine* dan *K-nearest Neighbor* dengan *k*=1. Perbedaan akurasi hasil klasifikasi ini wajar karena setiap metode memiliki kelebihan maupun kekurangan masing-masing dalam menangani data yang akan diklasifikasi.

5.10 Analisis Hasil Uji Coba Aplikasi

Berdasarkan skenario uji coba aplikasi pendeteksi suara tangis bayi yang dilakukan, akurasi tiap kelas jenis tangis bayi dan akurasi rata-rata aplikasi sangat baik (dapat dilihat pada Tabel 5.21). Hasil akurasi yang tinggi pada aplikasi tersebut dapat dicapai karena 85% data latih pada tiap kelas jenis tangis bayi yang digunakan pada aplikasi pendeteksi suara tangis bayi juga digunakan sebagai data uji. Jika data uji yang digunakan adalah 15% data dari masing-masing kelas yang belum menjadi data latih, maka akurasi rata-rata yang dihasilkan sebesar 75,95% (sama dengan akurasi rata-rata pada hasil uji pemilihan

metode terbaik yang diterapkan pada aplikasi) karena metode pada aplikasi diambil dari hasil terbaik berdasarkan kombinasi parameter *wintime*, nilai *k* pada *K-nearest Neighbor*, pembagian proporsi data latih dan data uji, serta metode normalisasi ekstraksi fitur.

LAMPIRAN

7.1 Hasil Uji Klasifikasi *K-nearest Neighbor* Data Latih 90% Data Uji 10% dengan berbagai skenario

Tabel 7.1 Hasil Uji Data Latih 90% Data Uji 10% *wintime* 0,02 *Decimal Normalization*

Jenis Tangis	*k*=1	*k*=2	*k*=3	*k*=4	*k*=5
Eairh/Eh	25,00%	100,00%	75,00%	100,00%	100,00%
Heh	50,00%	50,00%	100,00%	100,00%	100,00%
Neh	66,67%	0,00%	33,33%	33,33%	33,33%
Owh	33,33%	33,33%	33,33%	66,67%	33,33%
Rata-Rata	43,75%	45,83%	60,42%	75,00%	66,67%

Tabel 7.2 Hasil Uji Data Latih 90% Data Uji 10% *wintime* 0,02 *Min-Max Normalization*

Jenis Tangis	*k*=1	*k*=2	*k*=3	*k*=4	*k*=5
Eairh/Eh	25,00%	25,00%	25,00%	25,00%	25,00%
Heh	0,00%	50,00%	100,00%	0,00%	50,00%
Neh	66,67%	33,33%	66,67%	33,33%	0,00%
Owh	33,33%	33,33%	66,67%	66,67%	33,33%
Rata-Rata	31,25%	35,42%	64,58%	31,25%	27,08%

Tabel 7.3 Hasil Uji Data Latih 90% Data Uji 10% *wintime* 0,02 *Standard Deviation Normalization*

Jenis Tangis	*k*=1	*k*=2	*k*=3	*k*=4	*k*=5
Eairh/Eh	25,00%	50,00%	25,00%	50,00%	50,00%
Heh	0,00%	100,00%	50,00%	100,00%	100,00%
Neh	33,33%	33,33%	33,33%	33,33%	0,00%
Owh	0,00%	0,00%	0,00%	33,33%	0,00%
Rata-Rata	14,58%	45,83%	27,08%	54,17%	37,50%

Tabel 7.4 Hasil Uji Data Latih 90% Data Uji 10% *wintime* 0,04 *Decimal Normaliation*

Jenis Tangis	*k*=1	*k*=2	*k*=3	*k*=4	*k*=5
Eairh/Eh	50,00%	50,00%	25,00%	50,00%	50,00%
Heh	0,00%	100,00%	50,00%	100,00%	100,00%
Neh	66,67%	33,33%	66,67%	66,67%	66,67%
Owh	0,00%	0,00%	33,33%	33,33%	0,00%
Rata-Rata	29,17%	45,83%	43,75%	62,50%	54,17%

Tabel 7.5 Hasil Uji Data Latih 90% Data Uji 10% *wintime* 0,04 *Min-Max Normalization*

Jenis Tangis	*k*=1	*k*=2	*k*=3	*k*=4	*k*=5
Eairh/Eh	50,00%	50,00%	75,00%	50,00%	50,00%
Heh	50,00%	50,00%	50,00%	50,00%	50,00%
Neh	66,67%	33,33%	33,33%	33,33%	33,33%
Owh	33,33%	33,33%	33,33%	66,67%	66,67%
Rata-Rata	50,00%	41,67%	47,92%	50,00%	50,00%

Tabel 7.6 Hasil Uji Data Latih 90% Data Uji 10% *wintime* 0,04 *Standard Deviation Normalization*

Jenis Tangis	*k*=1	*k*=2	*k*=3	*k*=4	*k*=5
Eairh/Eh	50,00%	50,00%	50,00%	50,00%	50,00%
Heh	50,00%	50,00%	100,00%	50,00%	100,00%
Neh	66,67%	33,33%	33,33%	33,33%	33,33%
Owh	0,00%	0,00%	0,00%	33,33%	0,00%
Rata-Rata	41,67%	33,33%	45,83%	41,67%	45,83%

Tabel 7.7 Hasil Uji Data Latih 90% Data Uji 10% *wintime* 0,06 *Decimal Normalization*

Jenis Tangis	*k*=1	*k*=2	*k*=3	*k*=4	*k*=5
Eairh/Eh	25,00%	75,00%	50,00%	50,00%	50,00%
Heh	100,00%	100,00%	100,00%	100,00%	100,00%
Neh	100,00%	33,33%	100,00%	100,00%	100,00%
Owh	33,33%	33,33%	33,33%	33,33%	33,33%
Rata-Rata	64,58%	60,42%	70,83%	70,83%	70,83%

Tabel 7.8 Hasil Uji Data Latih 90% Data Uji 10% *wintime* 0,06 *Min-Max Normalization*

Jenis Tangis	*k*=1	*k*=2	*k*=3	*k*=4	*k*=5
Eairh/Eh	50,00%	50,00%	75,00%	75,00%	50,00%
Heh	50,00%	50,00%	50,00%	50,00%	50,00%
Neh	100,00%	66,67%	100,00%	66,67%	100,00%
Owh	66,67%	66,67%	66,67%	33,33%	66,67%
Rata-Rata	66,67%	58,33%	72,92%	56,25%	66,67%

Tabel 7.9 Hasil Uji Data Latih 90% Data Uji 10% *wintime* 0,06 *Standard Deviation Normalization*

Jenis Tangis	*k*=1	*k*=2	*k*=3	*k*=4	*k*=5
Eairh/Eh	75,00%	75,00%	50,00%	50,00%	50,00%
Heh	50,00%	100,00%	50,00%	50,00%	100,00%
Neh	100,00%	66,67%	66,67%	66,67%	66,67%
Owh	33,33%	0,00%	100,00%	100,00%	66,67%
Rata-Rata	64,58%	60,42%	66,67%	66,67%	70,83%

Tabel 7.10 Hasil Uji Data Latih 90% Data Uji 10% *wintime* 0,08 *Decimal Normalization*

Jenis Tangis	*k*=1	*k*=2	*k*=3	*k*=4	*k*=5
Eairh/Eh	25,00%	75,00%	25,00%	50,00%	25,00%
Heh	50,00%	100,00%	100,00%	100,00%	100,00%
Neh	100,00%	66,67%	66,67%	100,00%	100,00%
Owh	100,00%	33,33%	100,00%	33,33%	33,33%
Rata-Rata	68,75%	68,75%	72,92%	70,83%	64,58%

Tabel 7.11 Hasil Uji Data Latih 90% Data Uji 10% *wintime* 0,08 *Min-Max Normalization*

Jenis Tangis	*k*=1	*k*=2	*k*=3	*k*=4	*k*=5
Eairh/Eh	50,00%	50,00%	50,00%	50,00%	25,00%
Heh	50,00%	50,00%	50,00%	50,00%	50,00%
Neh	100,00%	33,33%	100,00%	66,67%	66,67%
Owh	100,00%	33,33%	66,67%	33,33%	66,67%
Rata-Rata	75,00%	41,67%	66,67%	50,00%	52,08%

Tabel 7.12 Hasil Uji Data Latih 90% Data Uji 10% *wintime* 0,08 *Standard Deviation Normalization*

Jenis Tangis	*k*=1	*k*=2	*k*=3	*k*=4	*k*=5
Eairh/Eh	50,00%	50,00%	75,00%	25,00%	50,00%
Heh	50,00%	50,00%	50,00%	50,00%	50,00%
Neh	100,00%	66,67%	100,00%	100,00%	66,67%
Owh	100,00%	66,67%	66,67%	66,67%	100,00%
Rata-Rata	75,00%	58,33%	72,92%	60,42%	66,67%

7.2 Hasil Uji Klasifikasi *K-nearest Neighbor* Data Latih 85% Data Uji 15% dengan berbagai skenario

Tabel 7.13 Hasil Uji Data Latih 85% Data Uji 15% *wintime* 0,02 *Decimal Normalization*

Jenis Tangis	*k*=1	*k*=2	*k*=3	*k*=4	*k*=5
Eairh/Eh	28,57%	100,00%	57,14%	71,43%	71,43%
Heh	66,67%	66,67%	100,00%	100,00%	100,00%
Neh	40,00%	0,00%	20,00%	20,00%	20,00%
Owh	40,00%	20,00%	20,00%	40,00%	20,00%
Rata-Rata	43,81%	46,67%	49,29%	57,86%	52,86%

Tabel 7.14 Hasil Uji Data Latih 85% Data Uji 15% *wintime* 0,02 *Min-Max Normalization*

Jenis Tangis	*k*=1	*k*=2	*k*=3	*k*=4	*k*=5
Eairh/Eh	42,86%	42,86%	42,86%	42,86%	42,86%
Heh	0,00%	33,33%	66,67%	0,00%	33,33%
Neh	60,00%	40,00%	60,00%	40,00%	40,00%
Owh	40,00%	20,00%	60,00%	60,00%	20,00%
Rata-Rata	35,71%	34,05%	57,38%	35,71%	34,05%

Tabel 7.15 Hasil Uji Data Latih 85% Data Uji 15% *wintime* 0,02 *Standard Deviation Normalization*

Jenis Tangis	*k*=1	*k*=2	*k*=3	*k*=4	*k*=5
Eairh/Eh	42,86%	57,14%	42,86%	57,14%	42,86%
Heh	0,00%	66,67%	33,33%	66,67%	66,67%
Neh	40,00%	20,00%	20,00%	40,00%	20,00%
Owh	0,00%	0,00%	0,00%	0,00%	0,00%
Rata-Rata	20,71%	35,95%	24,05%	40,95%	32,38%

Tabel 7.16 Hasil Uji Data Latih 85% Data Uji 15% *wintime* 0,04 *Decimal Normalization*

Jenis Tangis	*k*=1	*k*=2	*k*=3	*k*=4	*k*=5
Eairh/Eh	71,43%	71,43%	57,14%	57,14%	57,14%
Heh	33,33%	100,00%	66,67%	100,00%	100,00%
Neh	60,00%	40,00%	60,00%	60,00%	60,00%
Owh	0,00%	0,00%	20,00%	20,00%	0,00%
Rata-Rata	41,19%	52,86%	50,95%	59,29%	54,29%

Tabel 7.17 Hasil Uji Data Latih 85% Data Uji 15% *wintime* 0,04 *Min-Max Normalization*

Jenis Tangis	*k*=1	*k*=2	*k*=3	*k*=4	*k*=5
Eairh/Eh	42,86%	42,86%	71,43%	57,14%	57,14%
Heh	66,67%	66,67%	33,33%	66,67%	66,67%
Neh	60,00%	40,00%	40,00%	40,00%	40,00%
Owh	60,00%	60,00%	60,00%	60,00%	60,00%
Rata-Rata	57,38%	52,38%	51,19%	55,95%	55,95%

Tabel 7.18 Hasil Uji Data Latih 85% Data Uji 15% *wintime* 0,04 *Standard Deviation Normalization*

Jenis Tangis	*k*=1	*k*=2	*k*=3	*k*=4	*k*=5
Eairh/Eh	42,86%	71,43%	71,43%	57,14%	57,14%
Heh	66,67%	66,67%	66,67%	33,33%	66,67%
Neh	60,00%	40,00%	40,00%	40,00%	40,00%
Owh	20,00%	0,00%	0,00%	20,00%	20,00%
Rata-Rata	47,38%	44,52%	44,52%	37,62%	45,95%

Tabel 7.19 Hasil Uji Data Latih 85% Data Uji 15% *wintime* 0,06 *Decimal Normalization*

Jenis Tangis	*k*=1	*k*=2	*k*=3	*k*=4	*k*=5
Eairh/Eh	42,86%	85,71%	57,14%	57,14%	57,14%
Heh	100,00%	100,00%	100,00%	100,00%	100,00%
Neh	40,00%	20,00%	80,00%	80,00%	80,00%
Owh	40,00%	20,00%	40,00%	40,00%	20,00%
Rata-Rata	55,71%	56,43%	69,29%	69,29%	64,29%

Tabel 7.20 Hasil Uji Data Latih 85% Data Uji 15% *wintime* 0,06 *Min-Max Normalization*

Jenis Tangis	*k*=1	*k*=2	*k*=3	*k*=4	*k*=5
Eairh/Eh	42,86%	42,86%	71,43%	57,14%	42,86%
Heh	66,67%	66,67%	66,67%	66,67%	66,67%
Neh	60,00%	60,00%	60,00%	60,00%	80,00%
Owh	80,00%	40,00%	40,00%	40,00%	40,00%
Rata-Rata	62,38%	52,38%	59,52%	55,95%	57,38%

Tabel 7.21 Hasil Uji Data Latih 85% Data Uji 15% *wintime* 0,06 *Standard Deviation Normalization*

Jenis Tangis	*k*=1	*k*=2	*k*=3	*k*=4	*k*=5
Eairh/Eh	57,14%	71,43%	57,14%	42,86%	42,86%
Heh	66,67%	100,00%	66,67%	66,67%	100,00%
Neh	80,00%	40,00%	60,00%	40,00%	60,00%
Owh	40,00%	20,00%	80,00%	80,00%	60,00%
Rata-Rata	60,95%	57,86%	65,95%	57,38%	65,71%

Tabel 7.22 Hasil Uji Data Latih 85% Data Uji 15% *wintime* 0,08 *Decimal Normalization*

Jenis Tangis	*k*=1	*k*=2	*k*=3	*k*=4	*k*=5
Eairh/Eh	42,86%	85,71%	42,86%	57,14%	42,86%
Heh	66,67%	100,00%	100,00%	100,00%	100,00%
Neh	60,00%	40,00%	60,00%	80,00%	80,00%
Owh	80,00%	20,00%	60,00%	40,00%	20,00%
Rata-Rata	62,38%	61,43%	65,71%	69,29%	60,71%

Tabel 7.23 Hasil Uji Data Latih 85% Data Uji 15% *wintime* 0,08 *Min-Max Normalization*

Jenis Tangis	*k*=1	*k*=2	*k*=3	*k*=4	*k*=5
Eairh/Eh	42,86%	42,86%	57,14%	42,86%	28,57%
Heh	66,67%	66,67%	66,67%	66,67%	66,67%
Neh	60,00%	40,00%	80,00%	60,00%	60,00%
Owh	80,00%	40,00%	60,00%	20,00%	40,00%
Rata-Rata	62,38%	47,38%	65,95%	47,38%	48,81%

Tabel 7.24 Hasil Uji Data Latih 85% Data Uji 15% *wintime* 0,08 *Standard Deviation Normalization*

Jenis Tangis	*k*=1	*k*=2	*k*=3	*k*=4	*k*=5
Eairh/Eh	57,14%	57,14%	71,43%	42,86%	42,86%
Heh	66,67%	66,67%	66,67%	66,67%	66,67%
Neh	80,00%	60,00%	80,00%	80,00%	60,00%
Owh	100,00%	60,00%	60,00%	60,00%	100,00%
Rata-Rata	75,95%	60,95%	69,52%	62,38%	67,38%

7.3 Hasil Uji Klasifikasi *K-nearest Neighbor* Data Latih 80% Data Uji 20% dengan berbagai skenario

Tabel 7.25 Hasil Uji Data Latih 80% Data Uji 20% *wintime* 0,02 *Decimal Normalization*

Jenis Tangis	*k*=1	*k*=2	*k*=3	*k*=4	*k*=5
Eairh/Eh	22,22%	77,78%	55,56%	66,67%	66,67%
Heh	60,00%	40,00%	60,00%	60,00%	80,00%
Neh	42,86%	14,29%	42,86%	42,86%	57,14%
Owh	42,86%	28,57%	28,57%	42,86%	14,29%
Rata-Rata	41,98%	40,16%	46,75%	53,10%	54,52%

Tabel 7.26 Hasil Uji Data Latih 80% Data Uji 20% *wintime* 0,02 *Min-Max Normalization*

Jenis Tangis	*k*=1	*k*=2	*k*=3	*k*=4	*k*=5
Eairh/Eh	33,33%	33,33%	33,33%	33,33%	33,33%
Heh	20,00%	40,00%	60,00%	20,00%	20,00%
Neh	42,86%	28,57%	42,86%	28,57%	28,57%
Owh	28,57%	28,57%	57,14%	14,29%	28,57%
Rata-Rata	31,19%	32,62%	48,33%	24,05%	27,62%

Tabel 7.27 Hasil Uji Data Latih 80% Data Uji 20% *wintime* 0,02 *Standard Deviation Normalization*

Jenis Tangis	*k*=1	*k*=2	*k*=3	*k*=4	*k*=5
Eairh/Eh	44,44%	55,56%	33,33%	44,44%	33,33%
Heh	20,00%	40,00%	40,00%	60,00%	60,00%
Neh	28,57%	14,29%	14,29%	42,86%	28,57%
Owh	0,00%	0,00%	0,00%	28,57%	14,29%
Rata-Rata	23,25%	27,46%	21,90%	43,97%	34,05%

Tabel 7.28 Hasil Uji Data Latih 80% Data Uji 20% *wintime* 0,04 *Decimal Normalization*

Jenis Tangis	*k*=1	*k*=2	*k*=3	*k*=4	*k*=5
Eairh/Eh	77,78%	77,78%	66,67%	66,67%	55,56%
Heh	40,00%	100,00%	60,00%	100,00%	100,00%
Neh	71,43%	42,86%	71,43%	71,43%	71,43%
Owh	14,29%	14,29%	28,57%	28,57%	14,29%
Rata-Rata	50,87%	58,73%	56,67%	66,67%	60,32%

Tabel 7.29 Hasil Uji Data Latih 80% Data Uji 20% *wintime* 0,04 *Min-Max Normalization*

Jenis Tangis	*k*=1	*k*=2	*k*=3	*k*=4	*k*=5
Eairh/Eh	55,56%	66,67%	77,78%	66,67%	66,67%
Heh	60,00%	40,00%	60,00%	60,00%	80,00%
Neh	57,14%	28,57%	42,86%	14,29%	28,57%
Owh	42,86%	42,86%	42,86%	42,86%	42,86%
Rata-Rata	53,89%	44,52%	55,87%	45,95%	54,52%

Tabel 7.30 Hasil Uji Data Latih 80% Data Uji 20% *wintime* 0,04 *Standard Deviation Normalization*

Jenis Tangis	*k*=1	*k*=2	*k*=3	*k*=4	*k*=5
Eairh/Eh	55,56%	77,78%	77,78%	66,67%	66,67%
Heh	60,00%	40,00%	60,00%	40,00%	60,00%
Neh	71,43%	28,57%	42,86%	14,29%	28,57%
Owh	28,57%	14,29%	14,29%	42,86%	0,00%
Rata-Rata	53,89%	40,16%	48,73%	40,95%	38,81%

Tabel 7.31 Hasil Uji Data Latih 80% Data Uji 20% *wintime* 0,06 *Decimal Normalization*

Jenis Tangis	*k*=1	*k*=2	*k*=3	*k*=4	*k*=5
Eairh/Eh	44,44%	77,78%	44,44%	55,56%	55,56%
Heh	100,00%	100,00%	100,00%	100,00%	100,00%
Neh	57,14%	28,57%	71,43%	71,43%	71,43%
Owh	42,86%	28,57%	42,86%	42,86%	28,57%
Rata-Rata	61,11%	58,73%	64,68%	67,46%	63,89%

Tabel 7.32 Hasil Uji Data Latih 80% Data Uji 20% *wintime* 0,06 *Min-Max Normalization*

Jenis Tangis	*k*=1	*k*=2	*k*=3	*k*=4	*k*=5
Eairh/Eh	44,44%	55,56%	77,78%	66,67%	55,56%
Heh	60,00%	80,00%	80,00%	80,00%	80,00%
Neh	57,14%	42,86%	57,14%	57,14%	57,14%
Owh	85,71%	28,57%	28,57%	42,86%	42,86%
Rata-Rata	61,83%	51,75%	60,87%	61,67%	58,89%

Tabel 7.33 Hasil Uji Data Latih 80% Data Uji 20% *wintime* 0,06 *Standard Deviation Normalization*

Jenis Tangis	*k*=1	*k*=2	*k*=3	*k*=4	*k*=5
Eairh/Eh	66,67%	77,78%	66,67%	55,56%	55,56%
Heh	80,00%	100,00%	60,00%	80,00%	100,00%
Neh	71,43%	28,57%	42,86%	28,57%	28,57%
Owh	57,14%	42,86%	71,43%	57,14%	42,86%
Rata-Rata	68,81%	62,30%	60,24%	55,32%	56,75%

7.4 Hasil Uji Klasifikasi *K-nearest Neighbor* Data Latih 75% Data Uji 25% dengan berbagai skenario

Tabel 7.34 Hasil Uji Data Latih 80% Data Uji 20% *wintime* 0,08 *Decimal Normalization*

Jenis Tangis	*k*=1	*k*=2	*k*=3	*k*=4	*k*=5
Eairh/Eh	44,44%	77,78%	44,44%	55,56%	44,44%
Heh	60,00%	100,00%	100,00%	100,00%	100,00%
Neh	71,43%	42,86%	71,43%	71,43%	85,71%
Owh	71,43%	28,57%	71,43%	28,57%	28,57%
Rata-Rata	61,83%	62,30%	71,83%	63,89%	64,68%

Tabel 7.35 Hasil Uji Data Latih 80% Data Uji 20% *wintime* 0,08 *Min-Max Normalization*

Jenis Tangis	*k*=1	*k*=2	*k*=3	*k*=4	*k*=5
Eairh/Eh	44,44%	44,44%	55,56%	44,44%	44,44%
Heh	60,00%	60,00%	60,00%	60,00%	60,00%
Neh	71,43%	57,14%	85,71%	85,71%	71,43%
Owh	85,71%	42,86%	57,14%	28,57%	42,86%
Rata-Rata	65,40%	51,11%	64,60%	54,68%	54,68%

Tabel 7.36 Hasil Uji Data Latih 80% Data Uji 20% *wintime* 0,08 *Standard Deviation Normalization*

Jenis Tangis	*k*=1	*k*=2	*k*=3	*k*=4	*k*=5
Eairh/Eh	55,56%	66,67%	77,78%	55,56%	55,56%
Heh	60,00%	60,00%	60,00%	60,00%	60,00%
Neh	85,71%	57,14%	85,71%	71,43%	57,14%
Owh	100,00%	57,14%	71,43%	57,14%	85,71%
Rata-Rata	75,32%	60,24%	73,73%	61,03%	64,60%

Tabel 7.37 Hasil Uji Data Latih 75% Data Uji 25% *wintime* 0,02 *Decimal Normalization*

Jenis Tangis	*k*=1	*k*=2	*k*=3	*k*=4	*k*=5
Eairh/Eh	36,36%	81,82%	72,73%	72,73%	72,73%
Heh	50,00%	33,33%	50,00%	50,00%	66,67%
Neh	33,33%	11,11%	33,33%	33,33%	44,44%
Owh	33,33%	22,22%	22,22%	44,44%	11,11%
Rata-Rata	38,26%	37,12%	44,57%	50,13%	48,74%

Tabel 7.38 Hasil Uji Data Latih 75% Data Uji 25% *wintime* 0,02 *Min-Max Normalization*

Jenis Tangis	*k*=1	*k*=2	*k*=3	*k*=4	*k*=5
Eairh/Eh	45,45%	45,45%	45,45%	54,55%	54,55%
Heh	16,67%	33,33%	50,00%	16,67%	16,67%
Neh	33,33%	22,22%	33,33%	33,33%	33,33%
Owh	22,22%	11,11%	22,22%	11,11%	22,22%
Rata-Rata	29,42%	28,03%	37,75%	28,91%	31,69%

Tabel 7.39 Hasil Uji Data Latih 75% Data Uji 15% *wintime* 0,02 *Standard Deviation Normalization*

Jenis Tangis	*k*=1	*k*=2	*k*=3	*k*=4	*k*=5
Eairh/Eh	54,55%	63,64%	54,55%	63,64%	54,55%
Heh	16,67%	33,33%	33,33%	50,00%	50,00%
Neh	22,22%	11,11%	11,11%	33,33%	22,22%
Owh	11,11%	0,00%	0,00%	22,22%	11,11%
Rata-Rata	26,14%	27,02%	24,75%	42,30%	34,47%

Tabel 7.40 Hasil Uji Data Latih 75% Data Uji 25% *wintime* 0,04 *Decimal Normalization*

Jenis Tangis	*k*=1	*k*=2	*k*=3	*k*=4	*k*=5
Eairh/Eh	72,73%	81,82%	72,73%	63,64%	63,64%
Heh	33,33%	83,33%	50,00%	83,33%	83,33%
Neh	77,78%	55,56%	66,67%	66,67%	66,67%
Owh	22,22%	11,11%	33,33%	33,33%	22,22%
Rata-Rata	51,52%	57,95%	55,68%	61,74%	58,96%

Tabel 7.41 Hasil Uji Data Latih 75% Data Uji 15% *wintime* 0,04 *Min-Max Normalization*

Jenis Tangis	*k*=1	*k*=2	*k*=3	*k*=4	*k*=5
Eairh/Eh	45,45%	63,64%	72,73%	54,55%	72,73%
Heh	50,00%	33,33%	50,00%	50,00%	83,33%
Neh	55,56%	33,33%	44,44%	22,22%	33,33%
Owh	44,44%	44,44%	44,44%	44,44%	33,33%
Rata-Rata	48,86%	43,69%	52,90%	42,80%	55,68%

Tabel 7.42 Hasil Uji Data Latih 75% Data Uji 15% *wintime* 0,04 *Standard Deviation Normalization*

Jenis Tangis	*k*=1	*k*=2	*k*=3	*k*=4	*k*=5
Eairh/Eh	45,45%	72,73%	63,64%	63,64%	72,73%
Heh	50,00%	33,33%	50,00%	33,33%	50,00%
Neh	66,67%	33,33%	44,44%	33,33%	44,44%
Owh	44,44%	11,11%	11,11%	55,56%	11,11%
Rata-Rata	51,64%	37,63%	42,30%	46,46%	44,57%

Tabel 7.43 Hasil Uji Data Latih 75% Data Uji 25% *wintime* 0,06 *Decimal Normalization*

Jenis Tangis	*k*=1	*k*=2	*k*=3	*k*=4	*k*=5
Eairh/Eh	45,45%	81,82%	54,55%	63,64%	63,64%
Heh	83,33%	83,33%	83,33%	83,33%	83,33%
Neh	66,67%	44,44%	66,67%	77,78%	77,78%
Owh	44,44%	22,22%	44,44%	33,33%	33,33%
Rata-Rata	59,97%	57,95%	62,25%	64,52%	64,52%

Tabel 7.44 Hasil Uji Data Latih 75% Data Uji 25% *wintime* 0,06 *Min-Max Normalization*

Jenis Tangis	*k*=1	*k*=2	*k*=3	*k*=4	*k*=5
Eairh/Eh	54,55%	72,73%	81,82%	63,64%	63,64%
Heh	50,00%	66,67%	83,33%	66,67%	66,67%
Neh	55,56%	33,33%	55,56%	55,56%	33,33%
Owh	88,89%	44,44%	44,44%	55,56%	44,44%
Rata-Rata	62,25%	54,29%	66,29%	60,35%	52,02%

Tabel 7.45 Hasil Uji Data Latih 75% Data Uji 25% *wintime* 0,06 *Standard Deviation Normalization*

Jenis Tangis	*k*=1	*k*=2	*k*=3	*k*=4	*k*=5
Eairh/Eh	63,64%	81,82%	72,73%	72,73%	63,64%
Heh	66,67%	83,33%	50,00%	66,67%	83,33%
Neh	77,78%	44,44%	55,56%	44,44%	44,44%
Owh	66,67%	44,44%	77,78%	66,67%	44,44%
Rata-Rata	68,69%	63,51%	64,02%	62,63%	58,96%

Tabel 7.46 Hasil Uji Data Latih 75% Data Uji 25% *wintime* 0,08 *Decimal Normalization*

Jenis Tangis	*k*=1	*k*=2	*k*=3	*k*=4	*k*=5
Eairh/Eh	45,45%	63,64%	54,55%	63,64%	45,45%
Heh	50,00%	83,33%	83,33%	83,33%	83,33%
Neh	77,78%	55,56%	77,78%	77,78%	88,89%
Owh	66,67%	33,33%	66,67%	22,22%	22,22%
Rata-Rata	59,97%	58,96%	70,58%	61,74%	59,97%

Tabel 7.47 Hasil Uji Data Latih 75% Data Uji 25% *wintime* 0,08 *Min-Max Normalization*

Jenis Tangis	*k*=1	*k*=2	*k*=3	*k*=4	*k*=5
Eairh/Eh	54,55%	54,55%	63,64%	54,55%	54,55%
Heh	50,00%	50,00%	50,00%	50,00%	50,00%
Neh	55,56%	44,44%	55,56%	77,78%	44,44%
Owh	88,89%	44,44%	55,56%	33,33%	44,44%
Rata-Rata	62,25%	48,36%	56,19%	53,91%	48,36%

Tabel 7.48 Hasil Uji Data Latih 75% Data Uji 25% *wintime* 0,08 *Standard Deviation Normalization*

Jenis Tangis	*k*=1	*k*=2	*k*=3	*k*=4	*k*=5
Eairh/Eh	63,64%	81,82%	72,73%	72,73%	72,73%
Heh	50,00%	50,00%	50,00%	66,67%	66,67%
Neh	88,89%	66,67%	66,67%	66,67%	55,56%
Owh	100,00%	55,56%	66,67%	55,56%	77,78%
Rata-Rata	75,63%	63,51%	64,02%	65,40%	68,18%

7.5 Hasil Uji Klasifikasi *K-nearest Neighbor* Data Latih 70% Data Uji 30% dengan berbagai skenario

Tabel 7.49 Hasil Uji Data Latih 70% Data Uji 30% *wintime* 0,02 *Decimal Normalization*

Jenis Tangis	*k*=1	*k*=2	*k*=3	*k*=4	*k*=5
Eairh/Eh	38,46%	76,92%	69,23%	61,54%	69,23%
Heh	42,86%	42,86%	42,86%	42,86%	57,14%
Neh	40,00%	10,00%	30,00%	30,00%	40,00%
Owh	30,00%	20,00%	20,00%	40,00%	10,00%
Rata-Rata	37,83%	37,45%	40,52%	43,60%	44,09%

Tabel 7.50 Hasil Uji Data Latih 70% Data Uji 30% *wintime* 0,02 *Min-Max Normalization*

Jenis Tangis	*k*=1	*k*=2	*k*=3	*k*=4	*k*=5
Eairh/Eh	38,46%	46,15%	46,15%	53,85%	53,85%
Heh	14,29%	28,57%	42,86%	14,29%	14,29%
Neh	30,00%	20,00%	30,00%	30,00%	30,00%
Owh	20,00%	10,00%	20,00%	20,00%	30,00%
Rata-Rata	25,69%	26,18%	34,75%	29,53%	32,03%

Tabel 7.51 Hasil Uji Data Latih 70% Data Uji 30% *wintime* 0,02 *Standard Deviation Normalization*

Jenis Tangis	*k*=1	*k*=2	*k*=3	*k*=4	*k*=5
Eairh/Eh	46,15%	53,85%	53,85%	61,54%	53,85%
Heh	14,29%	42,86%	28,57%	42,86%	42,86%
Neh	30,00%	10,00%	20,00%	40,00%	30,00%
Owh	10,00%	0,00%	0,00%	20,00%	10,00%
Rata-Rata	25,11%	26,68%	25,60%	41,10%	34,18%

Tabel 7.52 Hasil Uji Data Latih 70% Data Uji 30% *wintime* 0,04 *Decimal Normalization*

Jenis Tangis	*k*=1	*k*=2	*k*=3	*k*=4	*k*=5
Eairh/Eh	69,23%	84,62%	61,54%	61,54%	61,54%
Heh	28,57%	71,43%	42,86%	71,43%	71,43%
Neh	70,00%	50,00%	60,00%	60,00%	60,00%
Owh	20,00%	10,00%	30,00%	30,00%	20,00%
Rata-Rata	46,95%	54,01%	48,60%	55,74%	53,24%

Tabel 7.53 Hasil Uji Data Latih 70% Data Uji 30% *wintime* 0,04 *Min-Max Normalization*

Jenis Tangis	*k*=1	*k*=2	*k*=3	*k*=4	*k*=5
Eairh/Eh	53,85%	69,23%	76,92%	61,54%	76,92%
Heh	42,86%	28,57%	42,86%	42,86%	71,43%
Neh	50,00%	30,00%	40,00%	30,00%	30,00%
Owh	40,00%	40,00%	40,00%	40,00%	30,00%
Rata-Rata	46,68%	41,95%	49,95%	43,60%	52,09%

Tabel 7.54 Hasil Uji Data Latih 70% Data Uji 30% *wintime* 0,04 *Standard Deviation Normalization*

Jenis Tangis	*k*=1	*k*=2	*k*=3	*k*=4	*k*=5
Eairh/Eh	46,15%	69,23%	53,85%	61,54%	76,92%
Heh	42,86%	28,57%	42,86%	28,57%	42,86%
Neh	60,00%	30,00%	40,00%	50,00%	40,00%
Owh	40,00%	10,00%	10,00%	50,00%	10,00%
Rata-Rata	47,25%	34,45%	36,68%	47,53%	42,45%

Tabel 7.55 Hasil Uji Data Latih 70% Data Uji 30% *wintime* 0,06 *Decimal Normalization*

Jenis Tangis	*k*=1	*k*=2	*k*=3	*k*=4	*k*=5
Eairh/Eh	46,15%	69,23%	53,85%	61,54%	53,85%
Heh	71,43%	71,43%	71,43%	71,43%	71,43%
Neh	60,00%	40,00%	60,00%	70,00%	70,00%
Owh	50,00%	20,00%	40,00%	30,00%	30,00%
Rata-Rata	56,90%	50,16%	56,32%	58,24%	56,32%

Tabel 7.56 Hasil Uji Data Latih 70% Data Uji 30% *wintime* 0,06 *Min-Max Normalization*

Jenis Tangis	*k*=1	*k*=2	*k*=3	*k*=4	*k*=5
Eairh/Eh	61,54%	76,92%	76,92%	69,23%	61,54%
Heh	42,86%	71,43%	57,14%	57,14%	57,14%
Neh	50,00%	30,00%	50,00%	50,00%	30,00%
Owh	90,00%	40,00%	70,00%	60,00%	60,00%
Rata-Rata	61,10%	54,59%	63,52%	59,09%	52,17%

Tabel 7.57 Hasil Uji Data Latih 70% Data Uji 30% *wintime* 0,06 *Standard Deviation Normalization*

Jenis Tangis	*k*=1	*k*=2	*k*=3	*k*=4	*k*=5
Eairh/Eh	61,54%	84,62%	69,23%	76,92%	69,23%
Heh	71,43%	57,14%	42,86%	71,43%	57,14%
Neh	70,00%	40,00%	50,00%	40,00%	40,00%
Owh	80,00%	50,00%	80,00%	70,00%	50,00%
Rata-Rata	70,74%	57,94%	60,52%	64,59%	54,09%

Tabel 7.58 Hasil Uji Data Latih 70% Data Uji 30% *wintime* 0,08 *Decimal Normalization*

Jenis Tangis	*k*=1	*k*=2	*k*=3	*k*=4	*k*=5
Eairh/Eh	30,77%	53,85%	46,15%	53,85%	46,15%
Heh	57,14%	71,43%	71,43%	71,43%	71,43%
Neh	70,00%	50,00%	70,00%	70,00%	90,00%
Owh	60,00%	30,00%	60,00%	20,00%	20,00%
Rata-Rata	54,48%	51,32%	61,90%	53,82%	56,90%

Tabel 7.59 Hasil Uji Data Latih 70% Data Uji 30% *wintime* 0,08 *Min-Max Normalization*

Jenis Tangis	*k*=1	*k*=2	*k*=3	*k*=4	*k*=5
Eairh/Eh	61,54%	61,54%	69,23%	61,54%	61,54%
Heh	42,86%	42,86%	42,86%	42,86%	42,86%
Neh	50,00%	40,00%	50,00%	70,00%	40,00%
Owh	80,00%	40,00%	50,00%	30,00%	40,00%
Rata-Rata	58,60%	46,10%	53,02%	51,10%	46,10%

Tabel 7.60 Hasil Uji Data Latih 70% Data Uji 30% *wintime* 0,08 *Standard Deviation Normalization*

Jenis Tangis	*k*=1	*k*=2	*k*=3	*k*=4	*k*=5
Eairh/Eh	69,23%	84,62%	76,92%	76,92%	76,92%
Heh	42,86%	42,86%	57,14%	57,14%	42,86%
Neh	80,00%	60,00%	60,00%	60,00%	50,00%
Owh	100,00%	50,00%	60,00%	60,00%	80,00%
Rata-Rata	73,02%	59,37%	63,52%	63,52%	62,45%

7.6 Hasil Uji Coba Pemilihan Metode Terbaik

Tabel 7.61 Hasil Uji Coba Pemilihan Metode Terbaik

Nama *file*	**Kelas Asal**	**Kelas Prediksi**
Eairh_8.wav	1	1

Eairh_9_1.wav	1	4
Eairh_9_2.wav	1	3
Eairh_9_3.wav	1	4
Eh_7.wav	1	1
Eh_8.wav	1	1
Eh_9.wav	1	1
Heh_7.wav	2	2
Heh_8.wav	2	3
Heh_9.wav	2	2
Neh_5.wav	3	3
Neh_6.wav	3	1
Neh_7.wav	3	3
Neh_8.wav	3	3
Neh_9.wav	3	3
Owh_5.wav	4	4
Owh_6.wav	4	4
Owh_7.wav	4	4
Owh_8.wav	4	4
Owh_9.wav	4	4

7.7 Hasil Uji Coba dengan Data Latih Seimbang pada Tiap Kelas

Tabel 7.62 Hasil Uji Coba KNN dengan Data Latih Seimbang

Nama File	**Kelas Asal**	**Klasifikasi KNN**				
		k=1	**k=2**	**k=3**	**k=4**	**k=5**
Eairh_18.wav	1	1	1	1	1	3
Eairh_19.wav	1	1	1	3	3	3
Eairh_2.wav	1	4	1	1	1	1
Eairh_20.wav	1	1	1	1	1	1

Nama File	Kelas Asal	Klasifikasi KNN				
		k=1	k=2	k=3	k=4	k=5
Eairh_3.wav	1	4	4	4	4	4
Eairh_4.wav	1	1	1	1	1	1
Eairh_5_1.wav	1	4	4	4	4	4
Eairh_5_2.wav	1	3	1	1	1	1
Eairh_6_1.wav	1	1	1	1	1	1
Eairh_6_2.wav	1	1	1	1	1	1
Eairh_7.wav	1	1	1	1	1	1
Eairh_8.wav	1	4	2	2	3	2
Eairh_9_1.wav	1	4	1	4	1	1
Eairh_9_2.wav	1	3	3	3	3	1
Eairh_9_3.wav	1	4	4	4	4	1
Eh_16.wav	1	2	2	2	2	2
Eh_17.wav	1	2	1	1	1	3
Eh_18.wav	1	3	3	2	3	3
Eh_19.wav	1	1	1	1	1	1
Eh_2.wav	1	1	1	1	3	3
Eh_3.wav	1	2	2	2	3	3
Eh_4.wav	1	1	1	1	1	1
Eh_5.wav	1	1	1	4	4	4
Eh_6.wav	1	3	1	3	3	3
Eh_7.wav	1	1	1	1	1	1
Eh_8.wav	1	1	1	1	1	1
Eh_9.wav	1	1	1	1	2	1
Heh_6.wav	2	2	2	2	2	2
Heh_7.wav	2	2	2	2	2	2
Heh_8.wav	2	1	1	2	2	1
Heh_9.wav	2	2	2	2	2	2

Nama File	Kelas Asal	Klasifikasi KNN				
		k=1	k=2	k=3	k=4	k=5
Neh_25.wav	3	2	2	3	3	3
Neh_26.wav	3	2	2	1	3	3
Neh_27.wav	3	3	3	3	2	2
Neh_28.wav	3	3	3	3	3	3
Neh_29.wav	3	3	1	3	3	3
Neh_3.wav	3	3	3	3	3	3
Neh_30.wav	3	1	1	1	1	1
Neh_31.wav	3	3	3	4	3	3
Neh_32_1.wav	3	3	3	3	3	3
Neh_32_2.wav	3	3	2	3	2	3
Neh_4.wav	3	3	3	3	3	3
Neh_5.wav	3	3	3	2	3	2
Neh_6.wav	3	1	1	1	1	1
Neh_7.wav	3	3	2	2	2	2
Neh_8.wav	3	3	3	3	3	2
Neh_9.wav	3	3	2	2	2	2
Owh_24.wav	4	2	1	1	3	3
Owh_25.wav	4	1	1	1	1	3
Owh_26.wav	4	4	4	4	4	1
Owh_27.wav	4	3	3	4	4	4
Owh_28.wav	4	2	2	3	3	3
Owh_29.wav	4	1	1	1	1	1
Owh_3.wav	4	4	3	2	2	2
Owh_30.wav	4	4	4	4	3	4
Owh_31_1.wav	4	4	1	1	1	1
Owh_31_2.wav	4	4	3	1	1	1
Owh_4.wav	4	3	1	3	3	2

Nama File	Kelas Asal	Klasifikasi KNN				
		k=1	k=2	k=3	k=4	k=5
Owh_5.wav	4	4	4	4	4	4
Owh_6.wav	4	4	2	2	4	2
Owh_7.wav	4	4	4	4	1	4
Owh_8.wav	4	4	4	4	4	4
Owh_9.wav	4	3	3	4	4	4

Tabel 7.63 *Confusion Matrix* Hasil Uji Data Latih Seimbang dengan KNN *k*=1

Hasil Prediksi	Eairh/Eh	Heh	Neh	Owh
Eairh/Eh	**14**	1	2	2
Heh	3	**3**	2	2
Neh	4	0	**12**	3
Owh	6	0	0	**9**

Tabel 7.64 *Confusion Matrix* Hasil Uji Data Latih Seimbang dengan KNN *k*=2

Hasil Prediksi	Eairh/Eh	Heh	Neh	Owh
Eairh/Eh	**19**	1	3	5
Heh	3	**3**	5	2
Neh	2	0	**8**	4
Owh	3	0	0	**5**

Tabel 7.65 *Confusion Matrix* Hasil Uji Data Latih Seimbang dengan KNN *k*=3

Hasil Prediksi	Eairh/Eh	Heh	Neh	Owh
Eairh/Eh	**15**	0	3	5
Heh	4	**4**	3	2
Neh	3	0	**9**	2
Owh	5	0	1	**7**

Tabel 7.66 *Confusion Matrix* Hasil Uji Data Latih Seimbang dengan KNN k=4

Hasil Prediksi	Eairh/Eh	Heh	Neh	Owh
Eairh/Eh	**14**	0	2	5
Heh	2	**4**	4	1
Neh	7	0	**10**	4
Owh	4	0	0	**6**

Tabel 7.67 Confusion Matrix Hasil Uji Data Latih Seimbang dengan KNN k=5

Hasil Prediksi	Eairh/Eh	Heh	Neh	Owh
Eairh/Eh	**15**	1	2	4
Heh	2	**3**	5	3
Neh	7	0	**9**	3
Owh	3	0	0	**6**

7.8 Hasil Uji Coba dengan Metode Klasifikasi Lain

Tabel 7.68 Hasil Klasifikasi dengan metode *Naive Bayes*, *Neural Network*, dan *Support Vector Machine*

Nama *file*	Kelas Asal	Kelas Prediksi Naive Bayes	Kelas Prediksi Neural Network	Kelas Prediksi SVM
Eairh_8.wav	1	3	4	3
Eairh_9_1.wav	1	4	1	4
Eairh_9_2.wav	1	3	4	3
Eairh_9_3.wav	1	4	4	3
Eh_7.wav	1	1	1	1
Eh_8.wav	1	3	1	1
Eh_9.wav	1	1	1	1
Heh_7.wav	2	2	3	2
Heh_8.wav	2	2	3	3

Nama *file*	Kelas Asal	Kelas Prediksi Naive Bayes	Kelas Prediksi Neural Network	Kelas Prediksi SVM
Heh_9.wav	2	2	3	2
Neh_5.wav	3	1	4	1
Neh_6.wav	3	1	1	1
Neh_7.wav	3	3	3	3
Neh_8.wav	3	1	1	3
Neh_9.wav	3	2	1	3
Owh_5.wav	4	4	4	4
Owh_6.wav	4	4	4	4
Owh_7.wav	4	1	1	4
Owh_8.wav	4	4	4	4
Owh_9.wav	4	4	4	4

Tabel 7.69 *Confusion Matrix* Hasil Uji Klasifikasi *Naive Bayes*

Hasil Prediksi	Eairh/Eh	Heh	Neh	Owh
Eairh/Eh	**2**	0	3	1
Heh	0	**3**	1	0
Neh	3	0	**1**	0
Owh	2	0	0	**4**

Tabel 7.70 *Confusion Matrix* Hasil Uji Neural Network

Hasil Prediksi	Eairh/Eh	Heh	Neh	Owh
Eairh/Eh	**4**	0	3	1
Heh	0	**0**	0	0
Neh	0	3	**1**	0
Owh	3	0	1	**4**

Tabel 7.71 *Confusion Matrix* Hasil Uji *Support Vector Machine*

Hasil Prediksi	Eairh/Eh	Heh	Neh	Owh
Eairh/Eh	**3**	0	2	0
Heh	0	**2**	0	0
Neh	3	1	**3**	0
Owh	1	0	0	**5**

7.9 Hasil Uji Coba Aplikasi Pendeteksi Suara Tangis Bayi

Tabel 7.72 Hasil Klasifikasi Aplikasi Pendeteksi Suara Tangis Bayi

Nama *file*	Kelas Asal	Hasil Prediksi
Eairh_1.wav	1	1
Eairh_11_1.wav	1	1
Eairh_11_2.wav	1	1
Eairh_12_1.wav	1	1
Eairh_12_2.wav	1	1
Eairh_13.wav	1	1
Eairh_14.wav	1	1
Eairh_15.wav	1	1
Eairh_16.wav	1	1
Eairh_17.wav	1	1
Eairh_18.wav	1	1
Eairh_19.wav	1	1
Eairh_2.wav	1	1
Eairh_20.wav	1	1
Eairh_3.wav	1	1
Eairh_4.wav	1	1
Eairh_5_1.wav	1	1
Eairh_5_2.wav	1	1

Nama *file*	**Kelas Asal**	**Hasil Prediksi**
Eairh_6_1.wav	1	1
Eairh_6_2.wav	1	1
Eairh_7.wav	1	1
Eairh_8.wav	1	1
Eairh_9_1.wav	1	4
Eairh_9_2.wav	1	3
Eairh_9_3.wav	1	4
Eh.wav	1	1
Eh_1_1.wav	1	1
Eh_1_2.wav	1	1
Eh_10.wav	1	1
Eh_11.wav	1	1
Eh_12.wav	1	1
Eh_13.wav	1	1
Eh_14.wav	1	1
Eh_15.wav	1	1
Eh_16.wav	1	1
Eh_17.wav	1	1
Eh_18.wav	1	1
Eh_19.wav	1	1
Eh_2.wav	1	1
Eh_3.wav	1	1
Eh_4.wav	1	1
Eh_5.wav	1	1
Eh_6.wav	1	1
Eh_7.wav	1	1
Eh_8.wav	1	1
Eh_9.wav	1	1

Nama *file*	**Kelas Asal**	**Hasil Prediksi**
Heh.wav	2	2
Heh_1.wav	2	2
Heh_10.wav	2	2
Heh_11.wav	2	2
Heh_12.wav	2	2
Heh_13.wav	2	2
Heh_14.wav	2	2
Heh_15.wav	2	2
Heh_16.wav	2	2
Heh_17.wav	2	2
Heh_18.wav	2	2
Heh_19.wav	2	2
Heh_2.wav	2	2
Heh_20.wav	2	2
Heh_20_1.wav	2	2
Heh_20_2.wav	2	2
Heh_3.wav	2	2
Heh_4.wav	2	2
Heh_5.wav	2	2
Heh_6.wav	2	2
Heh_7.wav	2	2
Heh_8.wav	2	3
Heh_9.wav	2	2
Neh.wav	3	3
Neh_1.wav	3	3
Neh_10.wav	3	3
Neh_11.wav	3	3
Neh_12.wav	3	3

Nama *file*	**Kelas Asal**	**Hasil Prediksi**
Neh_13.wav	3	3
Neh_14.wav	3	3
Neh_15.wav	3	3
Neh_16.wav	3	3
Neh_17.wav	3	3
Neh_18.wav	3	3
Neh_19_1.wav	3	3
Neh_19_2.wav	3	3
Neh_2.wav	3	3
Neh_20.wav	3	3
Neh_21.wav	3	3
Neh_22.wav	3	3
Neh_23.wav	3	3
Neh_24.wav	3	3
Neh_25.wav	3	3
Neh_26.wav	3	3
Neh_27.wav	3	3
Neh_28.wav	3	3
Neh_29.wav	3	3
Neh_3.wav	3	3
Neh_30.wav	3	3
Neh_31.wav	3	3
Neh_32_1.wav	3	3
Neh_32_2.wav	3	3
Neh_4.wav	3	3
Neh_5.wav	3	3
Neh_6.wav	3	1
Neh_7.wav	3	3

Nama *file*	**Kelas Asal**	**Hasil Prediksi**
Neh_8.wav	3	3
Neh_9.wav	3	3
Owh.wav	4	4
Owh_1.wav	4	4
Owh_10.wav	4	4
Owh_11.wav	4	4
Owh_12.wav	4	4
Owh_13.wav	4	4
Owh_14.wav	4	4
Owh_15.wav	4	4
Owh_16.wav	4	4
Owh_17.wav	4	4
Owh_18.wav	4	4
Owh_19.wav	4	4
Owh_2.wav	4	4
Owh_2_1.wav	4	4
Owh_2_2.wav	4	4
Owh_20.wav	4	4
Owh_21.wav	4	4
Owh_22.wav	4	4
Owh_23.wav	4	4
Owh_24.wav	4	4
Owh_25.wav	4	4
Owh_26.wav	4	4
Owh_27.wav	4	4
Owh_28.wav	4	4
Owh_29.wav	4	4
Owh_3.wav	4	4

Nama *file*	**Kelas Asal**	**Hasil Prediksi**
Owh_30.wav	4	4
Owh_31_1.wav	4	4
Owh_31_2.wav	4	4
Owh_4.wav	4	4
Owh_5.wav	4	4
Owh_6.wav	4	4
Owh_7.wav	4	4
Owh_8.wav	4	4
Owh_9.wav	4	4

BAB VI
KESIMPULAN DAN SARAN

Bab VI ini membahas tentang kesimpulan yang didasari oleh hasil uji coba pada bab sebelumnya. Kesimpulan tersebut nantinya menjawab rumusan masalah yang telah ada pada pendahuluan. Selain itu, juga terdapat saran sebagai acuan untuk mengembangkan topik Tugas Akhir ini lebih lanjut di masa depan.

6.1 Kesimpulan Uji Coba Pemilihan Metode

Berdasarkan hasil uji coba pemilihan metode, dapat diambil kesimpulan sebagai berikut:

- Parameter *wintime* pada proses ekstraksi fitur MFCC dari *package* tuneR ditentukan berdasarkan panjangnya durasi audio. Semakin panjang durasi audio yang akan diproses melalui ekstraksi fitur, maka nilai *wintime* harus semakin besar.
- Rata-rata akurasi terbaik sebesar 79,95% pada klasifikasi *K-nearest Neighbor* dengan k=1 didapat ketika proporsi data latih 85% data uji 15% dan proses ekstraksi fitur MFCC dengan durasi *wintime* 0,08 detik yang dinormalisasi dengan *Standard Deviation Normalization*. Hasil ini lebih baik dibandingkan dengan kombinasi parameter lainnya yang terdapat pada skenario pengujian pemilihan metode terbaik.

6.2 Kesimpulan Uji Coba dengan Data Latih Seimbang

Berdasarkan hasil uji coba dengan data latih seimbang dari tiap kelas, dapat diambil kesimpulan bahwa hasil klasifikasi dengan pembagian proporsi data latih berdasarkan persentase data dari masing-masing jenis tangis bayi menghasilkan akurasi yang lebih baik jika dibandingkan dengan klasifikasi yang menggunakan data latih seimbang dari tiap jenis tangis bayi.

6.3 Kesimpulan Uji Coba dengan Metode Klasifikasi Lain

Berdasarkan hasil uji coba dengan metode klasifikasi lain aplikasi, dapat diambil kesimpulan bahwa *K-nearest Neighbor* k=1 merupakan metode klasifikasi terbaik untuk kasus klasifikasi tangis bayi pada penelitian ini.

6.4 Kesimpulan Pembuatan Aplikasi

Berdasarkan hasil uji coba pembuatan aplikasi, dapat diambil kesimpulan bahwa aplikasi dapat mendeteksi masing-masing jenis tangis bayi dengan akurasi rata-rata 96,57%. Hasil ini diperoleh karena 85% dari data uji pada tiap kelas digunakan sebagai data latih dalam aplikasi.

6.5 Saran

Berikut merupakan beberapa saran oleh penulis untuk pengembangan perangkat lunak di masa yang akan datang:

- Memperbanyak jumlah data latih agar aplikasi dapat mempelajari macam-macam jenis tangis bayi lebih baik.
- Menentukan nilai *wintime* secara dinamis sesuai dengan durasi audio yang akan diproses dengan ekstraksi fitur MFCC dari *package* tuneR agar akurasi klasifikasi lebih tinggi.
- Mengembangkan perangkat lunak dalam versi perangkat bergerak agar lebih mudah dipakai sehari-hari.
- Menambahkan informasi usia bayi yang suara tangisnya dijadikan sampel dan menambahkan fitur deteksi tangis bayi berdasarkan rentang usia bayi agar hasil klasifikasi lebih akurat.
- Memperlebar rentang usia bayi yang suara tangisnya dijadikan sampel hingga usia 6 bulan agar perangkat lunak dapat digunakan lebih lama oleh orang tua dari bayi yang bersangkutan.

DAFTAR PUSTAKA

[1] K. Srijiranon dan N. Eiamkanitchat, “Application of Neuro-Fuzy Approaches to Recognition and Classification of Infant Cry,” dalam *TENCON 2014 - 2014 IEEE Region 10 Conference*, Bangkok, 2014.

[2] M. D. Renanti, “Identifikasi Arti Tangis Bayi Versi Dunstan Baby Language Menggunakan Jarak Terpendek dari Jarak Mahalanobis,” *Jurnal Sains Terapan,* vol. 3, no. 5, pp. 39-44, 2013.

[3] M. Hariharan, R. Sindhu dan S. Yaacob, “Normal and Hypoacoustic Infant Cry Signal Classification Using Time-Frequency Analysis and General Regression Neural Network,” *Computer Methods and Programs in Biomedcine,* vol. 108, no. 2, pp. 559-569, November 2012.

[4] P. Dunstan, “Open Up and Discover Your Baby's Language,” 2006. [Online]. Available: http://www.babytaal.nl/media/PDF/ComprehensiveBooklet(2).pdf. [Diakses 2 Juli 2016].

[5] L. Muda, M. Begam dan I. E. , “Voice Recognition Algorithms using Mel Frequency Cepstral Coefficient (MFCC) and Dynamic Time Warping (DTW) Techniques,” *JOURNAL OF COMPUTING,* vol. 2, no. 3, pp. 138-143, 2010.

[6] D. Putra dan A. Resmawan, “Verifikasi Biometrika Suara Menggunakan Metode MFCC dan DTW,” *Lontar Komputer,* vol. 2, pp. 8-21, 2011.

[7] R. Jang, “12-2 MFCC,” [Online]. Available: http://mirlab.org/jang/books/audiosignalprocessing/speechFeatureMfcc.asp?title=12-2%20MFCC. [Diakses 3 Juli 2016].

[8] "Frames Representation of Speech Signal," [Online]. Available: http://basic-programming.blogspot.co.id/2005/11/frames-representation-of-speech-signal.html. [Diakses 21 Juli 2016].

[9] Kalid, "An Interactive Guide To The Fourier Transform," [Online]. Available: http://betterexplained.com/articles/an-interactive-guide-to-the-fourier-transform/. [Diakses 5 Juli 2016].

[10] Z.-H. Tan dan B. Lindberg, "Use of Cepstral Liftering for MFCC Variance Normalization," dalam *Automatic Speech Recognition on Mobile Devices and over Communication Networks*, Springer Science & Business Media, 2008, pp. 148-150.

[11] Y.-p. Mack, K-Nearest neighbor estimation, California: The University of California, 1978.

[12] J. Brownlee, "How To Implement Naive Bayes From Scratch in Python," MachineLearningMastery.com, 8 Desember 2014. [Online]. Available: http://machinelearningmastery.com/naive-bayes-classifier-scratch-python/. [Diakses 11 Juli 2016].

[13] D. Suhartono, S.Kom., M.TI., "Dasar Pemahaman Neural Network," Binus University School of Computer Science, 26 Juli 2012. [Online]. Available: http://socs.binus.ac.id/2012/07/26/konsep-neural-network/. [Diakses 11 Juli 2016].

[14] N. Susanto, "Pengembangan Model Jaringan Syaraf Tiruan Resilient Backpropagation untuk Identifikasi Pembicara dengan Praproses MFCC," Insitut Pertanian Bogor, Bogor, 2007.

[15] K. Sembiring, "Tutorial SVM Bahasa Indonesia Oleh Krisantus," September 2007. [Online]. Available: http://sutikno.blog.undip.ac.id/files/2011/11/tutorial-

svm-bahasa-indonesia-oleh-krisantus.pdf. [Diakses 11 Juli 2016].

[16] U. Ligges, S. Krey, O. Mersmann, S. Schnackenberg, G. Guenard, A. Preusse, A. Thieler dan C. Weihs, "CRAN - Package tuneR," 20 Februari 2015. [Online]. Available: https://cran.r-project.org/web/packages/tuneR/index.html. [Diakses 2 Januari 2016].

[17] M. Kuhn., J. Wing, S. Weston, A. Williams, C. Keefer, A. Engelhardt, T. Cooper, Z. Mayer, B. Kenkel, . M. Benesty, R. Lescarbeau, A. Ziem, L. Scrucca, Y. Tang dan . C. Candan, "CRAN - Package caret," 11 April 2016. [Online]. Available: https://cran.r-project.org/web/packages/caret/index.html. [Diakses 1 Mei 2016].

[18] D. Meyer, E. Dimitriadou, K. Hornik, A. Weingessel, C. Chih-Chung dan L. Chih-Chen , "CRAN - Package e1071," 5 Agustus 2015. [Online]. Available: https://cran.r-project.org/web/packages/e1071/e1071.pdf. [Diakses 3 Juli 2016].

[19] S. Fritsch, F. Guenther dan M. Suling, "CRAN - Package neuralnet," 20 Februari 2015. [Online]. Available: https://cran.r-project.org/web/packages/neuralnet/neuralnet.pdf. [Diakses 3 Juli 2016].

[20] N. Dave, "Feature Extraction Methods LPC, PLP and MFCC In Speech Recognition," *International Journal for Advance Research in Engineering and Technology,* vol. 1, no. 6, pp. 1-5, 2013.

[21] P. P. Singh dan P. Rani, "An Approach to Extract Feature using MFCC," *IOSR Journal of Engineering (IOSRJEN),* vol. 4, no. 8, pp. 21-25, 2014.

[22] “Pengertian, kekurangan dan kelebihan K-NN,” 2012. [Online]. Available: http://cgeduntuksemua.blogspot.com/2012/03/pengertian-kelebihan-dan-kekurangan-k.html. [Diakses 15 Januari 2016].

[23] A. “Neural Network - Asro Pun' Blog,” 1 Juni 2010. [Online]. Available: https://asro.wordpress.com/2010/06/01/neural-network/. [Diakses 11 Juli 2016].

BIODATA PENULIS

Welly Setiawan Limantoro, lahir di Surabaya pada tanggal 24 Maret 1994. Penulis telah menempuh pendidikan di TKK Taman Rini Surabaya (1997-2000), SDK St. Angela Surabaya (2000-2006), SMPK Angelus Custos I Surabaya (2006-2009), SMAK Frateran Surabaya (2009-2012), dan sekarang sedang menjalani pendidikan S1 Teknik Informatika di ITS. Penulis aktif dalam organisasi Himpunan Mahasiswa Teknik Computer (HMTC) sebagai staff Kesejahteraan Mahasiswa periode 2013-2014 dan organisasi kerohanian Buddha TPKB ITS sebagai staff Pengembangan Sumber Daya Mahasiswa pada periode 2013-2014 dan periode 2014-2015. Penulis juga aktif dalam kegiatan kepanitiaan Schematics, diantaranya penulis pernah menjadi staff publikasi, dekorasi, dan dokumentasi pada Schematics 2013 dan Schematics 2014. Penulis juga merupakan salah satu admin di Mobile Innovation Studio. Penulis mengambil bidang minat Komputasi Cerdas dan Visi (KCV). Komunikasi dengan penulis dapat melalui email: **liem.wellysetiawan@gmail.com**.