TUGAS AKHIR – KI141502

DETEKSI MULTILEVEL DIABETES SECARA NON-INVASIVE DENGAN ANALISIS NAPAS MANUSIA MENGGUNAKAN BREATHALYZER

HARIYANTO
NRP 5113100061

Dosen Pembimbing I
Prof. Drs. Ec. Ir. Riyanarto Sarno, M.Sc., Ph.D.

Dosen Pembimbing II
Dedy Rahman Wijaya, S.T., M.T.

DEPARTEMEN TEKNIK INFORMATIKA
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya 2017

TUGAS AKHIR – KI141502

DETEKSI MULTILEVEL DIABETES SECARA NON-INVASIVE DENGAN ANALISIS NAPAS MANUSIA MENGGUNAKAN BREATHALYZER

HARIYANTO
NRP 5113100061

Dosen Pembimbing I
Prof. Drs. Ec. Ir. Riyanarto Sarno, M.Sc., Ph.D.

Dosen Pembimbing II
Dedy Rahman Wijaya, S.T., M.T.

DEPARTEMEN TEKNIK INFORMATIKA
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya 2017

[Halaman ini sengaja dikosongkan]

FINAL PROJECT– KI141502

NON-INVASIVE DETECTION OF MULTILEVEL DIABETES FROM ANALYSIS OF HUMAN BREATH USING BREATHALYZER

HARIYANTO
NRP 5113100061

Supervisor I
Prof. Drs. Ec. Ir. Riyanarto Sarno, M.Sc., Ph.D.

Supervisor II
Dedy Rahman Wijaya, S.T., M.T.

Department of Informatics
Faculty of Information and Technology
Institut Teknologi Sepuluh Nopember
Surabaya 2017

[Halaman ini sengaja dikosongkan]

LEMBAR PENGESAHAN

DETEKSI MULTILEVEL DIABETES SECARA NON-INVASIVE DENGAN ANALISIS NAPAS MANUSIA MENGGUNAKAN BREATHALYZER

TUGAS AKHIR

Diajukan Guna Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Bidang Studi Manajemen Informasi
Program Studi S-1 Departemen Teknik Informatika
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember

Oleh :
HARIYANTO
NRP : 5113 100 061

Disetujui oleh Dosen Pembimbing Tugas Akhir :

Prof. Drs. Ec. Ir. Riyanarto Sarno, M.Sc.,Ph.D. ..... (pembimbing 1)
NIP: 19590803 198601 1001

Dedy Rahman Wijaya, S.T., M.T. .....
NIP: 07840011 (pembimbing 2)

SURABAYA
JULI 2017

[Halaman ini sengaja dikosongkan]

DETEKSI MULTILEVEL DIABETES SECARA NON-INVASIVE DENGAN ANALISIS NAPAS MANUSIA MENGGUNAKAN BREATHALYZER

Nama Mahasiswa	**: Hariyanto**
NRP	**: 5113 100 061**
Jurusan	**: Teknik Informatika FTIf-ITS**
Dosen Pembimbing 1	**: Prof. Drs. Ec. Ir. Riyanarto Sarno, M.Sc., Ph.D.**
Dosen Pembimbing 2	**: Dedy Rahman Wijaya, S.T., M.T.**

ABSTRAK

Diabetes merupakan penyakit metabolic yang banyak diderita. Namu sayangnya, hanya sedikit orang yang mengetahui penyakit metabolic ini, terutama di Indonesia dimana orang hanya melakukan pemeriksaan kesehatan saat mereka merasa sakit. Oleh karena itu, dalam penelitian ini kami mengusulkan sistem non-invasive yang mudah digunakan dan berbiaya rendah yang dapat membedakan orang sehat dan orang diabetes sehingga dapat dilakukan pencegahan dini.

Ada tujuh tahap utama untuk membangun sistem ini, pembuatan perangkat keras e-Nose menggunakan sensor mikrokontroler dan gas, akuisisi data ground-truth untuk set pelatihan, pemrosesan sinyal menggunakan Discrete Wavelet Transform (DWT) dan normalisasi Z-score, fitur statistik Ekstraksi, pemilihan fitur untuk optimasi, klasifikasi, dan evaluasi kinerja e-Nose.

Hasil percobaan menunjukkan bahwa sistem ini dapat membedakan pasien sehat dan diabetes dengan kinerja yang menjanjikan (95,0% akurasi, ketepatan 91,30% diabetes, ketepatan 94,12% sehat dan 0,898 kappa statistik) dengan menggunakan classifier k-NN.

Kata kunci: classification, diabetes, e-Nose, k-NN, microcontroller, signal processing

[Halaman ini sengaja dikosongkan]

NON-INVASIVE DETECTION OF MULTICLASS DIABETES FROM ANALYSIS OF HUMAN BREATH USING BREATHALYZER

Student Name : Hariyanto
Student ID : 5113 100 061
Major : Informatics Department FTIf-ITS
Supervisor 1 : Prof. Drs. Ec. Ir. Riyanarto Sarno, M.Sc., Ph.D.
Supervisor 2 : Dedy Rahman Wijaya, S.T., M.T.

.

ABSTRACT

Diabetes is a disease that many people suffer. However, unfortunately only a few people that aware of this metabolic disease especially in Indonesia where people only do health check when they are feeling sick. Therefore, in this research we propose non-invasive, easy to use, and low-cost system that can distinguish healthy or diabetes people so they can have early preventive action.

There are seven main stages to build this system, the making of e-Nose hardware using microcontroller and gas sensors, ground-truth data acquisitions for the training set, signal processing using Discrete Wavelet Transform (DWT) and Z-score normalization, statistical features extraction, feature selection for optimization, classification, and e-Nose performance evaluation.

The experiment results show that this system can distinguish healthy and diabetes patients with promising performance (95.0% of accuracy, 91.30% precision of diabetes, 94.12% precision of healthy and 0.898 kappa statistic's value) using k-NN classifier.

Keyword: classification, diabetes, e-Nose, k-NN, microcontroller, signal processing

[Halaman ini sengaja dikosongkan]

KATA PENGANTAR

"*Give thanks to the LORD, for HE is good; His love endures forever." – Psalm 118:1*

Puji syukur kepada Tuhan Yesus Kristus atas segala berkat, bantuan karunia dan rahmat-Nya penulis dapat menyelesaikan Tugas Akhir yang berjudul:

Deteksi Multilevel Diabetes secara Non-Invasive dengan Analisis Napas Manusia Menggunakan Breathalyzer

Tugas Akhir ini boleh selesai karena tidak terlepas dari bantuan dan dukungan banyak pihak. Oleh karena itu, melalui lembar ini, penulis ingin menyampaikan ucapan terimakasih dan penghormatan yang sebesar-besarnya kepada:

1. Ayah dan Ibu Penulis, Johannes dan Mulyawati yang tiada hentinya memberikan dukungan doa, moral, dan material kepada penulis sehingga penulis dapat menyelesaikan Tugas Akhir ini.
2. Bapak Prof. Drs. Ec. Ir. Riyanarto Sarno, M.Sc., Ph.D. dan Bapak Dedy Rahman Wijaya, M.T. selaku dosen pembimbing I dan dosen pembimbing II yang telah membagikan ilmunya serta membimbing dan memberikan nasihat dan motivasi dalam menyelesaikan Tugas Akhir ini.
3. dr. RR. Endang Dwihastutiningsih dan dr. Pipiet Paramita sebagai pembimbing untuk mendapatkan data *ground truth* di Puskesmas Kedungdoro
4. Grup "theBurgator" dengan anggota Devira Wiena Pramintya, Ayu Kardina Sukmawati, Budi Pangestu Tanujaya, Gian Sebastian Anjasmara, Achmad Rizky Haqiqi, dan Franky Setiawan yang telah menjadi keluarga penulis sejak masa awal perkuliahan dan menghibur disaat susah.

5. Astidhita Nuraini Latifa, Nindyasari Dewi Utari, Kinasih Nur Azizah, Rifqi Nur Fadhillah, Luwandino Wismar, dan Nanang Taufan yang telah menjadi sahabat yang selalu menemani dan berjuang bersama
6. Aldhiaz Fathra Daiva yang menjadi teman seperjuangan penulis dalam mengerjakan tugas akhir
7. Teman-teman lain yang tidak dapat penulis sebutkan satu-persatu yang telah menghibur dan membantu penulis selama perkuliahan

Penulis menyadari masih terdapat banyak kekurangan baik dalam pelaksanaan tugas akhir maupun penyusunan buku laporan ini, namun penulis berharap buku tugas akhir ini dapat menambah wawasan pembaca dan dapat menjadi sumber referensi. Penulis mengharapkan kritik dan saran yang membangun untuk kesempurnaan penulisan buku tugas akhir ini

Surabaya, Juli 2017

Hariyanto.

DAFTAR ISI

DAFTAR GAMBAR

[Halaman ini sengaja dikosongkan]

DAFTAR TABEL

DAFTAR KODE SUMBER

BAB I
PENDAHULUAN

Pada bab ini dibahas hal-hal yang mendasari tugas akhir. Bahasan meliputi latar belakang, tujuan, rumusan dan batasan masalah, manfaat, metodologi, dan sistematika laporan tugas akhir.

1.1. Latar Belakang Masalah

Diabetes merupakan salah satu penyakit metabolik yang dapat mempengaruhi hampir semua sistem organ tubuh yang berjangka panjang dan ditandai dengan kadar gula darah (glukosa) yang jauh di atas normal. Indonesia termasuk dalam 10 negara terbesar penderita diabetes, dengan perkiraan jumlah 8,5 juta orang dalam rentang usia 20-79 tahun (Federasi Diabetes Internasional, 2013), tetapi, hanya 50% dari jumlah tersebut yang menyadari teridap penyakit diabetes.

Pada umumnya, orang melakukan cek diabetes dengan melakukan pemeriksaan darah di laboratorium. Pengecekan diabetes dengan menggunakan sampel darah di laboratorium seringkali memberikan stigma yang negatif, yaitu mahal, sakit, dan lama. Sakit, karena pengujian kadar gula dalam darah saat ini masih menggunakan teknik *invasive* dengan mengambil darah pasien menggunakan jarum suntik. Hal ini merupakan salah satu penyebab beberapa pasien enggan melakukan pemeriksaan darah. Selain itu, diperlukan waktu $\pm$ 2 jam untuk melakukan pengujian diabetes pada darah.

Diabetes dapat juga dilihat dari kandungan keton dalam darah, yaitu sebuah senyawa asam hasil residu ketika tubuh mulai membentuk energi dengan memaksimalkan pembakaran lemak dibanding karbohidrat. Seringkali, orang yang mengidap diabetes berbau napas aseton, karena adanya badan keton di dalam darah yang salah satu senyawanya adalah aseton. Maka dari itu, orang yang mengidap diabetes dapat ditandai juga dengan kandungan

gas dalam napas yang berbeda dengan orang yang tidak mengidap diabetes. Melihat salah satu ciri yang khas dari napas penderita diabetes, maka dapat di analisa data hubungan kadar kandungan gas dalam napas manusia dan penyakit diabetes.

Penggunaan *breathalyzer* untuk mendeteksi seseorang mengidap penyakit diabetes tentunya akan memberikan nilai kepraktisan yang lebih dan memudahkan seorang pasien untuk mengecek resiko diabetes secara *non-invasive*. Sistem yang dibuat akan melakukan pembelajaran dari kumpulan data orang yang mengidap penyakit diabetes dengan data pasien baru yang menggunakan *breathalyzer* untuk klasifikasi diabetes pasien.

1.2. Rumusan Permasalahan

Rumusan masalah yang terdapat dalam Tugas Akhir ini antara lain:

1. Bagaimana mendapatkan data kandungan gas dalam napas orang yang terkena diabetes?
2. Bagaimana cara merakit e-Nose untuk deteksi kandungan gas dalam napas orang yang terkena diabetes?
3. Bagimana cara pembelajaran data untuk melakukan klasifikasi seseorang terkena penyakit diabetes berdasar data yang direkam oleh e-Nose?
4. Bagaimana cara penyajian hasil klasifikasi dari analisa data pasien baru yang direkam oleh e-Nose?

1.3. Batasan Permasalahan

Permasalahan yang dibahas dalam Tugas Akhir ini memiliki beberapa batasan, antara lain:

1. Data kandungan gas dalam napas pasien dikumpulkan dari pasien diabetes di Puskesmas Kedungdoro, Kota Surabaya.
2. Alat yang dipakai untuk mengumpulkan data kandungan gas dalam napas adalah e-Nose yang dirakit menggunakan

microcontroller (Arduino) yang tersambung dengan 6 *electrochemical gas sensor*.

3. *Ground-truth* yang dipakai adalah data kandungan gas dalam napas pasien terdiagnosa diabetes yang direkam oleh e-Nose.
4. Kotak yang digunakan untuk menampung *sample* napas pasien ± 250 mL.

1.4. Tujuan

Tujuan dari pembuatan Tugas Akhir ini adalah:

1. Membangun sebuah e-Nose untuk mendapatkan data kandungan karbon monoksida, karbon dioksida, keton, temperatur, kelembapan serta *volatile organic compound* (VOC) dalam napas pasien.
2. Melakukan deteksi bahwa seorang terindikasi mengidap penyakit diabetes berdasar data yang direkam oleh e-Nose

1.5. Manfaat

Memudahkan individu untuk mendeteksi penyakit diabetes secara *non-invasive*, praktis, dan cepat sehingga dapat melakukan tindakan deteksi dini.

1.6. Metodologi

Langkah-langkah yang ditempuh dalam pengerjaan Tugas Akhir ini yaitu:

a. Penyusunan proposal tugas akhir

Proposal tugas akhir ini berisi tentang penduluan dan gambaran umum dari tugas akhir yang akan dibuat. Pendahuluan ini terdiri dari beberapa bab berupa latar belakang, rumusan masalah yang diangkat, batasan masalah untuk tugas akhir, tujuan dari pembuatan tugas akhir, dan manfaat dari hasil pembuatan tugas akhir. Selain itu dijelaskan pula mengenai tinjuan pustaka

sebagai referensi pendukung dalam pembuatan tugas akhir. Sub-bab metodologi menjelaskan mengenai tahapan penyusunan tugas akhir mulai dari pembuatan proposal hingga penyusuan buku tugas akhir. Dilampirkan pula sub-bab jadwal kegiatan yang menjadwalkan pembuatan tugas akhir.

b. Studi literatur

Pada studi literatur akan dipelajari sejumlah referensi yang diperlukan dalam pembuatan tugas akhir yaitu mengenai diabetes, badan keton, mikrokontroler dan Arduino, Bahasa pemrograman Arduino, sensor elektrokimia dan sensor lingkungan serta cara perhitungan data, perhitungan estimasi konsentrasi gas, analisis sinyal dengan wavelet, seleksi fitur, algoritma klasifikasi yaitu Jaringan Syaraf Tiruan serta *Support Vector Machine* serta metode evaluasi klasifikasi.

c. Analisis dan desain perangkat lunak

Pada tahap ini akan dilakukan analisis dan desain perangkat lunak untuk sistem diagnose diabetes. Analisis yang dilakukan adalah perbandingan algoritma klasifikasi (*neural network* dan *support vector machine)* dan tahapan persiapan data yang cocok untuk data yang dihimpun dari pasien di salah satu puskesmas di Surabaya.

Adapun desain perangkat lunak pada Tugas Akhir ini meliputi langkah-langkah berikut:

1. Pembuatan *breathalyzer* dengan sensor temperatur, tekanan udara, kelembapan, dan sensor elektrokimia
2. Pengumpulan data pasien terdiagnosa diabetes menggunakan breathalyzer yang dibuat
3. Pengujian metode *preprocessing* dan algoritma klasifikasi yang dapat memberikan performa terbaik untuk diagnosis pasien diabetes
4. Tahap pembuatan GUI (*Graphical User Interface*) berbasis *web* untuk menampilkan hasil diagnosa oleh sistem dari data sensor

d. Implementasi perangkat lunak

Sistem ini akan menggunakan *Integrated Development Environment*:

- Arduino Software: untuk memrogram *board Arduino* dan sensor.
- PyCharm 2016.2.3: untuk memrogram *classifier*.
- Bracket: untuk memrogram tampilan *web* sebagai GUI pengguna.
- Rapidminer Studio: untuk melakukan uji coba *classifier* dari dataset.

dan Bahasa pemrograman:

- Bahasa Pemrograman Arduino: untuk *board Arduino*.
- Python 2.7: untuk membuat *classifier*.
- PHP, HTML, JavaScript, dan CSS: untuk membuat GUI berbasis *web*.

e. Pengujian dan evaluasi

Pengujian dari sistem ini akan dilakukan dalam beberapa cara yaitu:

1. Pengujian hasil pembacaan sensor
   Pengujian ini dilakukan untuk mencoba apakah sensor yang ada pada Arduino berhasil mendapatkan data dari inputan pengguna. Serta pada pengujian ini akan dilakukan apakah komunikasi antara papan Arduino dengan komputer *desktop* dapat berjalan lancer
2. Pengujian Usabilitas
   Pengujian ini akan dilakukan dengan cara survei langsung ke calon pasien dengan menggunakan alat yang dibuat. Survei dilakukan untuk mengukur tingkat kegunaan alat dan sistem yang dibuat dalam membantu pengguna.
3. Pengujian akurasi, presisi, dan *recall*
   Pengujian dilakukan menggunakan *dataset* yang dihimpun dari puskesmas dengan menggunakan metode evaluasi *cross-validation* dan perhitungan *accuracy*, *precission*,

recall dan *kappa statistic* untuk mengukur performa sistem dalam melakukan diagnosis pasien diabetes

f. 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. Sistematika penulisan buku tugas akhir secara garis besar antara lain:

1. Pendahuluan
    a. Latar Belakang Masalah
    b. Rumusan Permasalahan
    c. Batasan Permasalahan
    d. Tujuan
    e. Metodologi
    f. Sistematika Penulisan
2. Dasar Teori
3. Desain dan Implementasi
4. Pengujian dan Evaluasi
5. Kesimpulan dan Saran
6. Daftar Pustaka

1.7. Sistematika Penulisan

Buku Tugas Akhir ini bertujuan untuk mendapatkan gambaran dari pengerjaan Tugas Akhir ini. Selain itu, diharapkan dapat berguna untuk pembaca yang tertarik untuk melakukan pengembangan lebih lanjut. Secara garis besar, buku Tugas Akhir terdiri atas beberapa bagian seperti berikut ini.

Bab I Pendahuluan

Bab ini berisi latar belakang masalah, tujuan dan manfaat pembuatan Tugas Akhir, permasalahan, batasan masalah, metodologi yang digunakan, dan sistematika penyusunan Tugas Akhir.

Bab II Dasar Teori

Bab ini membahas beberapa teori penunjang yang berhubungan dengan pokok pembahasan dan mendasari pembuatan Tugas Akhir ini.

Bab III Metodologi

Bab ini membahasa mengenai metodologi pengambilan data *ground-truth*, metodologi pembuatan e-nose dan uji performa e-nose

Bab IV Analisis dan Perancangan Sistem

Bab ini membahas mengenai perancangan perangkat lunak. Perancangan perangkat lunak meliputi perancangan data, arsitektur, proses dan perancangan antarmuka aplikasi.

Bab V Implementasi

Bab ini berisi implementasi dari perancangan dan implementasi fitur-fitur penunjang aplikasi.

Bab VI Pengujian dan Evaluasi

Membahas tentang lingkungan pengujian, skenario pengujian, dan evaluasi pengujian setelah aplikasi selesai dikembangkan.

Bab VII Kesimpulan dan Saran

Bab ini berisi kesimpulan dari hasil pengujian yang dilakukan. Bab ini membahas saran-saran untuk pengembangan sistem lebih lanjut.

Daftar Pustaka

Merupakan daftar referensi yang digunakan untuk mengembangkan Tugas Akhir.

Lampiran

Merupakan bab tambahan yang berisi daftar istilah yang penting pada aplikasi ini.

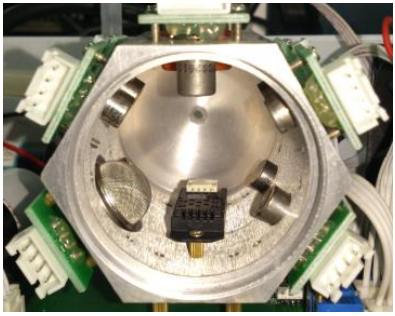
[Halaman ini sengaja dikosongkan]

BAB II
DASAR TEORI

Pada bab ini akan dibahas mengenai teori-teori yang menjadi dasar dari pembuatan Tugas Akhir.

2.1. Penelitian Terkait

Pada penelitian sebelumnya yang terakit dengan tugas akhir ini, yaitu penggunaan beberapa sensor elektrokimia untuk deteksi biomarker dalam napas manusia untuk prediksi kadar gula dalam darah [1]. Penelitian tersebut menggunakan 11 sensor yang terdiri atas enam sensor *metal oxide semiconductor* (MOS), tiga sensor MOS termodulasi suhu, satu sensor karbon dioksida, dan satu sensor temperatur-kelembapan. Sensor-sensor yang digunakan pada penelitian tersebut di dirancang untuk mengukur kandungan *volatile organic compound* (VOC), karbon dioksida, kelembapan dan temperatur dalam sampel napas. Untuk mendapatkan sampel napas, subjek diminta untuk menarik napas kemudian menghembuskan napas ke dalam 600 mL kantong gas Tedlar melalui corong sekali pakai [1]. Rancangan alat pada penelitian tersebut dapat dilihat pada **Error! Reference source not found.** berupa sensor gas yang menempel pada dinding ruang gas.



Gambar 2.1 Ruang Gas pada Penelitian Sebelumnya

Setelah didapatkan data dari sensor gas, kemudian diterapkan metode pra-proses data dan ekstraksi fitur untuk mendapatkan informasi identifikasi subjek. Setelah itu, untuk melakukan diagnosis diabetes terhadap subjek dilakukan dengan melakukan pembelajaran data menggunakan *Support Vector Machine* terhadap data sampel dalam basis data. Performa sensitivitas dan spesifisitas yang didapat oleh *classifier* untuk membedakan pasien diabetes dan pasien sehat dalam penelitian tersebut masing-masing sebesar 91.5% dan 90.77%.

2.2. Glukosa dan Diabetes

Glukosa merupakan zat yang ada di dalam darah yang bersal dari karbohidrat dati makanan dan minuman yang dikonsumsi setiap hari oleh manusia. Diabetes merupakan salah satu penyakit metabolik yang dapat mempengaruhi hampir semua sistem organ tubuh yang berjangka panjang dan ditandai dengan kadar gula darah (glukosa) yang jauh di atas normal serta adanya gula dalam urin.

Dalam 100 ml darah manusia normal akan mengandung konsentrasi gula darah yang tetap, yaitu 70-100 mg. Pada orang yang menderita diabetes melitus, jumlah konsentrasi gula darah adalah 130 mg/100 ml darah. Beberapa macam hormon terlibat dalam pengaturan darah ini, salah satunya insulin.Diabetes terjadi ketika tubuh tidak menghasilkan insulin yang cukup untuk mempertahankan kadar gula darah yang normal.

Ada 2 macam diabetes, yaitu diabetes tipe 1 dan diabetes tipe 2. Diabetes tipe 1 adalah diabetes yang bergantung pada insulin, dimana tubuh kekurangan hormon insulin. Sedangkan diabetes tipe 2 terjadi jika hormon insulin di dalam tubuh tidak dapat berfungsi dengan seharusnya. Kedua jenis diabetes ini mengakibatkan banyaknya kandungan glukosa yang terdapat

dalam tubuh. Dapat dilihat kandungan gula dalam darah dan interpretasi penyakit diabetes pada **Tabel 2.1** [2].

Tabel 2.1 Tabel Kandungan Gula Darah dan Interpretasi Diabetes

mmol/L	mg/dL	Interpretasi
2,0	35	Sangat rendah
3,0	55	Rendah
4,0	75	Agak rendah
4,4	80	Normal
5,5	100	Normal
5-6	90-100	Normal sebelum makan untuk non-diabetic
8,0	150	Normal setelah makan untuk non-diabetic
10,0	180	Maksimal setelah untuk non-diabetic
15,0	270	Sedikit tinggi ke agak tinggi tergantung pada pasien
20,0	360	Sangat tinggi
22,0	400	Maksimal (Maksimal untuk beberapa test meter)
33,0	600	Bahaya tinggi

Selain kadar gula darah yang dapat menunjukkan interpretasi diabetes, kandungan gas yang ada dalam sampel napas pasien diabetes juga dapat dijadikan patokan dalam membedakan pasien sehat dengan pasien diabetes. Sebagai contoh, pasien yang mengidap penyakit diabetes memiliki kandungan keton dalam napas dan keringat, sehingga menyebabkan napas dan keringat yang mengeluarkan aroma aseton [3]. Hal itu disebabkan oleh hasil metabolisme lemak. Keton merupakan produk sampingan dari metabolisme lemak. Ketika sel tubuh tidak mendapatkan asupan glukosa yang cukup, maka hati akan mengubah lemak menjadi asam keton yang akan digunakan sebagai bahan bakar oleh otot. Pada pasien yang mengidap diabetes, insulin tidak diproduksi sesuai dengan kadar yang seharusnya, sehingga glukosa tidak dapat dihantarkan dengan optimal menuju sel-sel tubuh. Untuk memenuhi kebutuhan energi, kemudian sel mulai membakar lemak dan memaksimalkan produksi energi dan pembakaran lemak. Selain kandungan keton yang menjadi tolak ukur pasien diabetes, senyawa lain seperi ethanol [4], karbon monoksida [5], alkane [6], dan methyl nitrat [7] dalam napas juga telah dibuktikan memiliki

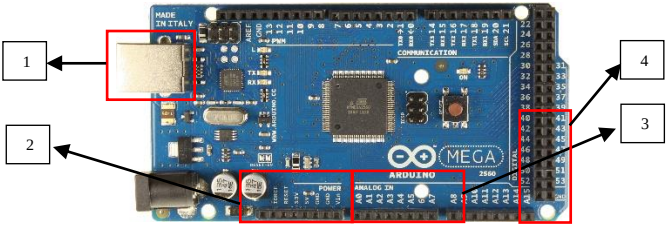
konsentrasi yang berbeda dengan pasien sehat dan memiliki korelasi terhadap prediksi gula darah. Pada penelitian ini, hanya dilakukan pengukuran terhadap empat kandungan gas dalam napas pasien yaitu: karbon monoksida (CO), karbon dioksida (CO_2), ketone, dan *volatile organic compound* (VOC) serta temperatur dan kelembapan napas pasien untuk membedakan pasien sehat dan pasien diabetes.

2.3. Mikrokontroler dan Arduino

Mikrokontroler merupakan sebuah komputer mikro dalam satu *chip* tunggal yang merupakan alat elektronika digital yang mempunyai masukan dan keluaran serta kendali dengan program yang bisa ditulis dan dihapus dengan cara khusus [1]. Arduino merupakan papan rangkaian elektronik *open source* yang di dalamnya terdapat komponen utama berupa *chip* mikrokontroler dengan jenis AVR yang dirancang untuk memudahkan penggunaan elektronik dalam berbagai bidang [2]. Kelebihan dari Arduino, antara lain:

a. Tidak perlu perangkat *chip programmer*, karena sudah ada *bootloader* yang akan menangani *upload* program dari komputer.
b. Sudah ada sarana komunikasi USB
c. Memiliki modul siap pakai yang bisa ditancapkan pada *board* Arduino. Contoh: *shiled* GPS, Ethernet, sensor gas, dll.

Pada penelitian ini, papan Arduino digunakan untuk membuat *electronic nose* (e-Nose) yang berguna untuk mengukur kandungan gas karbon monoksida (CO), karbon dioksida (CO_2), keton, temperature, kelembapan, dan *volatile organic compound* (VOC) dalam sampel napas pasien dengan menggunakan sensor elektrokimia. Papan Arduino yang digunakan merupakan papan ArduinoBoard MEGA 2560.



Gambar 2.2 Komponen pada Papan Arduino Mega 2560

Terdapat 3 bagian penting pada papan Arduino yang digunakan untuk mengambil data kandungan gas dalam napas pasien. Ketiga nama komponen beserta fungsinya dapat dilihat pada **Tabel 2.2**

Tabel 2.2 Deskripsi Komponen yang Digunakan pada Papan Arduino MEGA 2560

No.	Nama	Fungsi
1.	*USB Jacks*	Menghubungkan papan Arduino dengan komputer untuk transfer data
2.	*Power Pins*	Terdiri dari 6 pin, tetapi pada penelitian ini yang hanya 3 pin yang dipakai: • 5V: menyediakan keluaran 5 *volt* • GND (*Ground*): untuk mengubungkan dengan sirkuit (gas sensor) • Vin: digunakan untuk memberi tenaga pada gas sensor yang terhubung pada *bread board*
3.	*Analog Input Pins*	Pin ini dapat membaca sinyal yang berasal dari sensor analog dan mengkonversi data sensor analog

		menjadi nilai digital yang dapat dibaca oleh mikroprosesor
4.	*Digital Input Pins*	Pin ini membaca sinyal yang berasal dari sensor digital (DHT-22)

2.4. Sensor Elektrokimia dan Sensor Kelembapan

Sensor elektrokimia merupakan peralatan pendeteksi polutan gas yang bekerja berdasarkan reaksi antara komponen sensor dengan analit yang dapat berupa gas atau ion kemudian menghasilkan sinyal elektrik yang setara dengan konsentrasi analit [8]. Sensor elektrokimia (MOS Sensor) inilah yang akan dihubungkan dengan papan Arduino untuk mendeteksi adanya kandungan gas dalam napas pasien. Pada penelitian ini, digunakan 4 buah sensor elektrokimia untuk mendeteksi kandungan gas karbon monoksida, karbon dioksida, keton, dan *volatile organic compound* (VOC) serta 1 buah sensor temperatur-kelembapan dalam napas pasien. Sensor yang digunakan merupakan sensor analog yang mengeluarkan *output* beruapa *analog to digital conversion* (ADC). MOS Sensor mempunyai empat pin, yaitu VCC, GND, AO, dan DO. Pin VCC dihubungkan pada pin tegangan di mikrokontroler untuk memberikan daya pada sensor. Pin GND dihubungkan pada pin GND pada mikrokontroler. Pin AO (*analog output*) dihubungkan pada salah pin pada bagian ANALOG IN pada *board* mikrokontroler. Pada penelitian ini, ketiga pin itu yang akan digunakan.

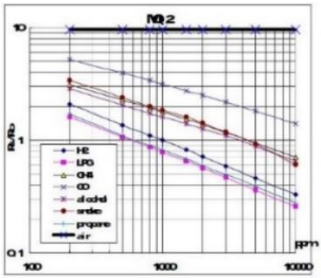
Tabel 2.3 Daftar Sensor Gas yang Digunakan dalam Pembuatan e-Nose

Sensor	Function
MQ-7	Carbon Monoxide (CO)
MQ-135	Carbon Dioxide (CO2)
MQ-138	Acetone

MiCS-5524	Volatile Organic Compound (VOC)
DHT-22	Temperature-Humidity

2.4.1. MQ-2

MQ-2 termasuk dalam keluarga sensor analog, yang banyak digunakan untuk mendeteksi kebocoran gas di perumahan maupun industri. MQ-2 dapat mendeteki berbagai macam gas: LPG, i-butane, Propana, Metana, Alkohol, Hidrogen, dan karbon monoksida. Sensor ini memiliki sensitivitas tinggi dan waktu respon yang cepat. Berikut dapat dilihat pada **Gambar 2.3** karakteristik MQ-2 dari respon sensor terhadap konsentrasi berbagai gas [9]. Pada penelitian ini, MQ-2 digunakan untuk mendeteki karbon monoksida (CO).

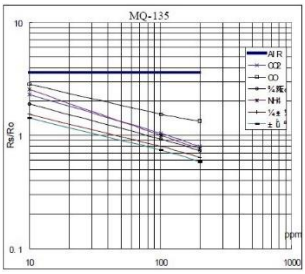


Gambar 2.3 Karakteristik Sensitivitas Sensor MQ-2 terhadap Berbagai Gas

2.4.2. MQ-135

Gas sensor tipe MQ-135 merupakan sensor yang mempunyai 2 keluaran yaitu digital dan *analog to digital conversion* (ADC). Sensor tipe MQ-135 disebut juga sebagai *air quality control* sensor, karena dapat mendeteksi keluaran gas yang mencemari lingkungan atau membahayakan kesehatan seperti karbon dioksida (CO_2), sulfur, ammonia, dan NH_3 . Pada penelitian ini, MQ-135 digunakan untuk mendeteksi kandungan

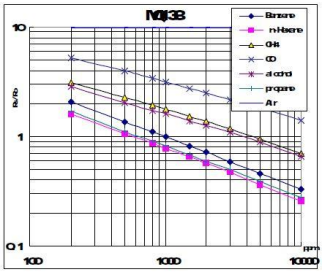
CO_2 dalam napas pasien. Berikut dapat dilihat pada **Gambar 2.4** sensitivitas respon sensor MQ-135 dalam mendeteksi konsentrasi beberapa gas [10].



Gambar 2.4 Karakteristik Sensitivitas Sensor MQ-135 terhadap Berbagai Gas

2.4.3. MQ-138

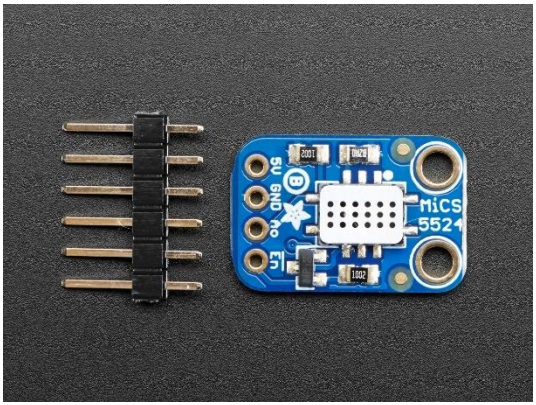
Sensor MQ-138 merupakan sensor semikonduktor yang bekerja Bersama dengan penghirup udara dan mikrokontroler untuk mengambil data dari napas pasien. Sensor MQ-138 dilapisi oleh bahan semikonduktor yang mempunyai nilai konduktivitas rendah pada udara normal. Apabila udara sekitar sensor terdapat aroma lain, maka konduktivitas sensor meningkat tergantung dari jenis aroma dan konsentrasi yang terkandung. Pada penelitian ini, MQ-138 digunakan untuk mengukur konsentrasi gas aseton dalam napas pasien. Berikut dapat dilihat pada **Gambar 2.5** sensitivitas respon sensor MQ-138 dalam mendeteksi konsentrasi beberapa gas [11].



Gambar 2.5 Karakteristik Sensitivitas Sensor MQ-138 terhadap Berbagai Gas

2.4.4. MiCS-5524

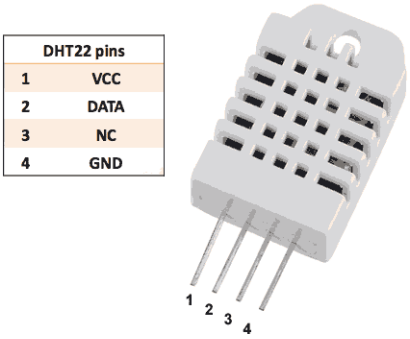
MiCS-5524 merupakan sebuah sensor *microelectromechanical systems* (MEMS) yang dapat mendeteksi karbon monoksida dan *volatile organic compound* (VOC). Sensor ini cocok digunakan untuk pemantauan kualitas udara dalam ruangan, pemeriksaan nafas, dan deteksi dini kebakaran. Pada penelitian ini, sensor MiCS-5524 digunakan untuk mengetahui kandungan VOC dalam napas pasien. Berikut dapat dilihat pada **Gambar 2.6** bentuk dari sensor MiCS-5524.



Gambar 2.6 Bentuk Sensor MiCS-5524 untuk Deteksi Kandungan VOC

2.4.5. DHT-22

DHT-22 yang juga dikenal sebagai AM2302 merupakan sensor yang dapat mengukur suhu dan kelembapan udara di sekitar sensor. Keluaran dari sensor sudah berupa sinyal digital dengan konversi dan perhitungan dilakukan oleh MCU 8-bit terpadu. Keluaran sensor ada pada pin DO (*digital* output) tidak perlu lagi di konversikan karena sudah berupa keluaran angka suhu dan kelembapan dalam celcius dan *relative humidity* (RH). Berikut dapat dilihat pada **Gambar 2.7** bentuk sensor DHT-22.



Gambar 2.7 Bentuk Sensor DHT-22 untuk Mengukur Suhu dan Kelembapan Napas

2.5. Perhitungan Nilai Respon dan Konsentrasi Gas dalam Sensor

Sensor elektrokimia (MOS Sensor) yang dipakai merupakan sensor analog sehingga nilai respon dari sensor merupakan hasil dari *analog to digital conversion* (ADC). Untuk menghitung resistansi sensor (R_s) dapat dicari dengan nilai ADC menggunakan rumus (2.1) dan (2.2) [12].

$$Rs = \frac{Vc - VRL}{VRL} \; x \; RL \qquad (0.1)$$

Dimana,

$$VRL = \frac{ADC \; x \; Vc}{1023} \qquad (0.2)$$

Vc adalah tegangan pada mikrokontroler, VRL adalah tegangan sensor di ruang sampel, RL adalah beban sensor resistensi (diukur menggunakan Ω meter, dan ADC adalah analog ke konversi nilai digital.

Untuk mencari konsentrasi gas (C), dalam napas pasien maka diturunkan rumus dari *datasheet* masing-masing sensor dengan parameter nilai resistansi sensor sebenarnya (R_s) dan nilai resistansi sensor pada udara bersih (R_O). Ada dua rumus yang dapat digunakan untuk mengukur kosentrasi gas berdasar resistansi sensor, yaitu rumus (2.3) [12] dan (2.4).

$$C = \gamma \left[\frac{R_s}{R_o}\right]^{\tau} \quad , \quad \gamma, \tau \epsilon R^{+} \qquad (0.3)$$

$$C = 10^{\frac{\log(\frac{Rs}{Ro} - \gamma)}{\beta}} \text{, } \alpha,\beta,\gamma \qquad (0.4)$$

Parameter nilai Vc dan RL masing-masing sensor yang digunakan pada penelitian ini dapat dilihat pada **Tabel 2.4.**

Tabel 2.4 Nilai Parameter Vc, RL, dan RO untuk Sensor Gas

Parameter	Nilai
V_c	5
R_L MQ-7	10
R_L MQ-135	9.94
R_L MQ-138	1.03
R_O MQ-7	28
R_O MQ-135	3.75
R_O MQ-138	1

Sedangkan untuk nilai parameter yang digunakan pada rumus (2.3) dan (2.4) dapat dilihat pada **Tabel 2.5.**

Tabel 2.5 Nilai Koefisien yang Digunakan untuk (2.3) dan (2.4)

Sensor	Gas	Persamaan (2.3)	Persamaan (2.4)

		γ	τ	α	β	γ
MQ-7	CO	34.29	-0.76	47.6	-0.85	0.07
MQ-135	CO_2	5.42	-0.35	5.89	-0.17	-1.55
MQ-138	Keton e	6.206	-0.75	-0.00026	1.40	0.38

2.6. *Ground Truth Data*

Pada penelitian ini, tidak terdapat dataset pasien sehat dan pasien diabetes berdasar respon sensor e-Nose yang tersedia di internet dikarenakan bedanya sensor yang dipakai. Maka dari itu, data *ground-truth* yang dipakai adalah data yang diambil sendiri menggunakan e-Nose yang dirangkai dengan bekerjasama dengan dokter dan institusi kesehatan untuk menentukan seseorang masuk ke kelas pasien sehat atau pasien diabetes berdasar kadar gula darah. Penentuan kelas data *ground truth* berdasar kadar gula darah dapat dilihat pada **Tabel 2.6**. Pasien yang dicatat adalah pasien gula darah acak.

Tabel 2.6 Penentuan Data *Ground Truth* berdasar *Blood Glucose Level (BGL)*

Kadar Gula Pasien	Kelas
<120	Healthy
>150	Chronic

2.7. Praproses Data

Praproses data merupakan tahapan yang penting dalam pengolahan data untuk membantu meningkatkan kualitas data yang berdampak pada hasil akhir klasifikasi, karena sangat berpengaruh terhadap proses yang lain dalam klasifikasi [13].

Ada dua tahapan yang dilakukan dalam praproses data pada penelitian ini, yaitu *signal denoising* dan normalisasi data.

2.7.1. Transformasi Wavelet Diskrit

Denoising sinyal merupakan salah satu tahapan penting dan membantu dalam meningkatkan performa akurasi dan sensitivitas *electronic nose* (e-Nose) [14]. Metode *signal denoising* yang digunakan adalah *discrete wavelet transform* (DWT) yang merupakan salah satu *wavelet transform* yang merepresentasikan sinyal dalam domain waktu dan frekuensi dan berasal dari *Continuous Wavelet Transform* dengan koefisien diskrit. Dekomposisi sinyal merepresentasikan fitur sinyal yang berubah secara perlahan di *band* frekuensi rendah sedangkan fitur yang berubah dengan cepat ada pada *band* dengan frekuensi yang lebih tinggi.

Ada dua nilai yang dihasilkan setelah melakukan DWT pada suatu sinyal yang dalam kasus ini adalah sinyal respon sensor terhadap perubahan konsentrasi gas dalam napas pasien, nilai tersebut merupakan *low-pass* dan *high-pass*. Nilai *low-pass* dianggap sebagai respon asli sensor terhadap perubahan konsentrasi sebuah gas yang diukur oleh sebuah sensor analog elektrokimia, sedangkan nilai *high-pass* merupakan resistansi sensor terhadap gas lain yang tidak responsive terhadap sensor.

Pada penelitian ini, koefisien yang diambil merupakan koefisien aproksimasi yang berasal dari *low-pass filter*. Nilai aproksimasi yang digunakan sangat bergantung pada level dekomposisi dan *mother wavelet* yang digunakan. *Mother wavelet* dan level dekomposisi yang digunakan adalah jenis *daubechies* 6 dan dekomposisi level 1 yang berdasar pada penelitian sebelumnya [15]. Semakin tinggi level dekomposisi maka resiko sinyal rusak juga semakin besar, tetapi semakin rendah level dekomposisi maka sinyal akan rentan terhadap noise.

DWT dan CWT memiliki rumus yang sama. Perbedaannya, untuk DWT, $s = 2^j$, $\tau = k^*2^j$, dimana

$(j,k) \in Z^2$. Sehingga *mother wavelet* dirumuskan pada persamaan (2.5)

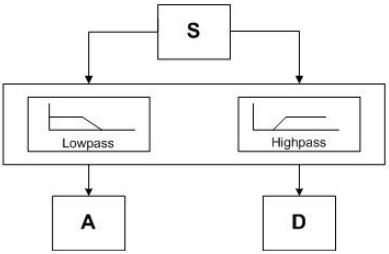
$$\psi_{s,\tau}(t) = \frac{1}{\sqrt{2^j}}\psi\left(\frac{t - k^*2^j}{2^j}\right) \quad \textbf{(0.5)}$$

Time series x(n) dapat di dekomposisi menjadi 2 *band* frekuensi (*low pass* dan *high pass*) menggunakan rumus DWT pada persamaan (2.6) dan (2.7).

$$y_h[m] = \sum_n x[n]h_0[2m - n] \quad \textbf{(0.6)}$$

$$y_g[m] = \sum_n x[n]g_0[2m - n] \quad \textbf{(0.7)}$$

Dimana y_h dan y_g merupakan *output* dari *high-pass* dan *low-pass* filter. Pada **Gambar 2.8** dapat dilihat hasil dekomposisi level 1 setelah dilakukan DWT, A merupakan koefisien aproksimasi dan D merupakan koefisien detail.



Gambar 2.8 Dekomposisi Level 1 menggunakan DWT

2.7.2. Normalisasi Skor Standar

Normalisasi atau *feature scaling* adalah teknik untuk menstandarkan atau menyamakan rentang data sehingga tidak ada satu atribut yang terlalu dominan atas atribut yang lain. Pada penelitian ini, normalisasi data dilakukan satu kali dan menggunakan normalisasi skor standar (*Z-Score Normalization*).

Feature Scaling menggunakan normalisasi skor dilakukan agar data sinyal yang dihasilkan setelah penghilangan *noise* memiliki properti seperti sebuah distribusi normal [16]. Hal ini dikarenakan, *output* setiap sensor yang kadar *parts per million* (ppm) yang di deteksi untuk setiap gas berbeda. Berikut adalah persamaan (2.8) untuk menghitung nilai standar (Z-Score) setiap respon sensor dalam sinyal.

$$z = \frac{x-\mu}{\sigma} \qquad (0.8)$$

z, x, μ, dan σ merupakan nilai standar, data asli (respon sensor per waktu), rata-rata data asli, dan standar deviasi data asli.

2.8. Fitur Statistik

Respon sensor terhadap konsentrasi gas terhadap waktu merupakan sebuah sinyal stasioner. Sinyal stasioner memiliki nilai parameter statistik yang konstan terhadap waktu [17]. Karena perilaku inilah, maka ekstraksi fitur yang dilakukan terhadap sinyal respon sensor adalah ekstraksi fitur statistik dengan memanfaatkan empat parameter statistik, yaitu: nilai rata-rata (*average*), nilai maksimal (*maximum value*), nilai minimum (*minimum value*), dan standar deviasi dari satu sinyal respon sensor. Perhitungan untuk mendapatkan keempat fitur statistik dapat dilihat pada persamaan (2.9), (2.10), (2.11), (2.12).

$$\mu = \frac{1}{n} x \sum_{t=0}^{n} y(t) \qquad (0.9)$$

μ adalah nilai rata-rata sinyal dalam satu periode waktu, n adalah panjang sinyal dan y(t) adalah nilai sinyal (respon sensor dalam ppm) terhadap waktu (t).

$$\sigma = \sqrt{\frac{1}{N} x \sum_{t=0}^{N} (y(t) - \mu)^2} \qquad (0.10)$$

σ adalah nilai standar deviasai sinyal y, N adalah panjang sinyal, y(t) adalah nilai sinyal terhadap waktu (t) dan adalah nilai rata-rata sinyal y.

$$V = max(y(t)) \quad (0.11)$$

$$S = min(y(t)) \quad (0.12)$$

V dan S adalah nilai maksimum dan minimum dari sinyal y.

2.9. Seleksi Fitur

Metode seleksi fitur diaplikasikan pada data untuk menghadapi masalah redundansi fitur yang dikarenakan tingginya korelasi fitur. Tujuan utama teknik seleksi fitur adalah mencari kombinasi optimal dari fitur statistik setiap sensor atau atribut optimal dari fitur statistik yang telah dilakukan perhitungan sebelumnya [18].

Pada penelitian ini, metode seleksi fitur yang digunakan adalah *weighting by Chi Square Statistic* yang digunakan untuk mengukur relevansi atribut terhadap kelas dengan menghitung nilai *chi-squared statistic* dari setiap data *ground-truth*. Untuk menghitung nilai *chi-squared statistic* digunakan persamaan (2.13).

$$X^2 = \sum \frac{(O-E)^2}{E} \quad (0.13)$$

X^2 adalah nilai *chi-squared statistic*, O adalah frekuensi hasil observasi, dan E adalah frekuensi yang diharapkan.

2.10. K-Nearest Neighbour

K-*Nearest Neighbour* merupakan algoritma *supervised learning* dimana hasil dari *instance* yang baru diklasifikasikan berdasarkan mayoritas dari kategori k-tetangga terdekat. Algoritma k-nn menggunakan *neighbourhood classification* yang didasarkan pada *voting* pada *training sample* untuk prediksi kelas *instance* (data) yang baru.

Pada penelitian sebelumnya digunakan *Support Vector Machine* untuk melakukan klasifikasi terhadap empat kelas

diabetes dan didapatkan akurasi 68.66% [19]. Pada penilitian ini digunakan k-nn untuk melakukan prediksi dua kelas data, yaitu: sehat dan diabetes dengan menggunakan *Canberra distance* untuk *similarity measuresement*/*distance method*. Perhitungan jarak Canberra dapat dilihat pada persamaan (2.14).

$$d_{CAD}(x,y) = \sum_{i=1}^{n} \frac{|x_i - y_i|}{|x_i| + |y_i|} \quad (0.14)$$

d_{CAD} adalah hasil perhitungan jarak Canberra vektor x terhadap vektor y, n adalah panjang vektor atribut, x adalah vektor atribut data sedangkan y adalah vector atribut *training set*. Jarak Canberra merupakan *weighted manhattan distance* [20].

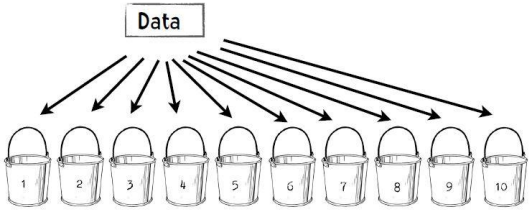
2.11. Metode Evaluasi

Metode evaluasi dalam penelitian ini adalah metode *training-testing* dan pengukuran yang digunakan untuk mengukur kinerja *classifier* yang digunakan.

2.11.1. Metode Validasi Silang

Metode validasi silang atau *cross-validation* merupakan metode yang cukup popular untuk evaluasi yang membagi data menjadi dua bagian, data pelatihan (*training set*) dan data pengujian (*testing set*). Selanjutnya, setelah data uji dilakukan proses silang dimana data pengujian dijadikan data pelatihan ataupun sebaliknya [21].

Pembagian data menjadi *training* dan *testing* set tidak dilakukan seperti pada metode *hold-out* dimana data *training* biasanya lebih banyak daripada data *testing*. Pada metode validasi silang, dataset (*ground-truth*) dibagi menjadi beberapa bagian secara random, biasanya disebut dengan k-*fold cross validation* (k-ember validasi silang). Nilai k yang biasa digunakan adalah 10. Sebagai contoh penerapan pembagian data pada metode 10-*fold cross validation* dapat dilihat pada **Gambar 2.9.**



Gambar 2.9 Pembagian Dataset pada Metode 10-*Fold Cross Validation*

Selain metode 10-*fold cross validation*, metode *leave-one-out* atau n-*fold cross validation*, dimana n adalah jumlah data dalam dataset memberikan pengukuran akurasi yang lebih presisi tetapi memakan waktu perhitungan yang lebih lama. Metode *k-fold cross validation* dianggap kurang presisi, karena metode tersebut membagi data kedalam k ember secara *random* sehingga ketika dijalankan kembali metode tersebut untuk kedua kalinya, mungkin memberikan kinerja akurasi yang berbeda akibat peletakan data yang acak.

2.11.2. Confusion Matrix

Confusion Matrix adalah sebuah statistik klasifikasi yang menyimpan informasi mengenai prediksi kelas dan kelas asli. *Confusion matrix* banyak digunakan untuk menguji performa dari suatu metode klasifikasi [22]. Pada penelitian ini, *confusion matrix* menyimpan informasi klasifikasi pasien diabetes dan pasien sehat oleh k-NN. Struktur *confusion matrix* dapat dilihat pada **Gambar 2.10.**

Dalam *confusion matrix* pada tugas akhir ini terdapat statistik *true positive* (TP) yaitu jumlah pasien yang diprediksi oleh klasifier sebagai pasien sehat adalah pasien yang benar-benar sehat, *true negative* (TN) yaitu jumlah pasien yang diprediksi oleh klasifier sebagai pasien diabetes adalah pasien yang benar-benar berpenyakit diabetes, *false positive* (FP) yaitu jumlah pasien yang diabetes tetapi diprediksi oleh klasifier sebagai pasien sehat dan

false negative (FN) yaitu jumlah pasien yang sehat tetapi diprediksi oleh klasifier sebagai pasien diabetes.

	Predicted: Healthy	**Predicted: Diabetes**
Actual: Healthy	TruePositive(TP)	FalseNegative(FN)
Actual: Diabetes	FalsePoisitve(FP)	TrueNegative(TN)

Gambar 2.10 *Confusion Matrix* untuk Klasifikasi Diabetes

Dari *confusion matrix* bisa didapatkan berbagai informasi mengenai performa *classifier*, di antaranya akurasi, *precision*, *recall*, dan *kappa statistic*.

2.11.3. Akurasi

Akurasi adalah jumlah total data yang prediksi kelas hasil klasifikasinya sesuai dengan data *ground-truth* [22]. Rumus perhitungan akurasi ditunjukkan pada persamaan (2.15).

$$Accuracy = \frac{TP+TN}{TP+FP+FN+TN} \quad (0.15)$$

2.11.4. Precision

Precision adalah proporsi pasien yang diprediksi sebagai pasien diabetes yang benar-benar mengidap penyakit diabetes. Rumus perhitungan *precision* ditunjukkan pada persamaan (2.16)

$$Precision\ of\ Diabetes = \frac{TN}{Predicted\ Diabetes} \quad (0.16)$$

2.11.5. Sensitivity

Sensitivity atau *recall* adalah proporsi pasien yang benar-benar mengidap penyakit diabetes diprediksi oleh *classifier*

sebagai pasien diabetes. Rumus perhitungan *recall* ditunjukkan pada persamaan (2.17).

$$\textbf{\textit{Recall of Diabetes}} = \frac{TN}{Actual\ Diabetes} \qquad (0.17)$$

2.11.6. Kappa Statistic

Kappa statistic sangat cocok digunakan untuk mengukur performa *classifier* untuk kelas biner. *Kappa statistic* membandingkan performa *classifier* (akurasi) dengan *classifier* yang membuat prediksi berdasarkan kebetulan (*solely on chance*) [23]. Rumus perhitungan nilai *kappa statistic* berdasarkan *confusion matrix* ditunjukkan oleh persamaan (2.18).

$$\textbf{\textit{Kappa}} = \frac{P(c)-P(r)}{1-P(r)} \qquad (0.18)$$

kappa adalah nilai koefisien kappa, P(c) adalah performa *classifier* (akurasi), dan P(r) adalah performa *random classifier*. Performa *random classifier* dapat dihitung dengan persamaan (2.19)

$$P(r) = \frac{actualH*\%predictedDiabetes + actualD*\%predictedHealthy}{totalData} \qquad (0.19)$$

actualH adalah jumlah pasien yang benar-benar sehat dan actualD adalah jumlah pasien yang benar-benar mengidap diabetes.
Setelah didapatkan nilai *kappa statistic* maka interpretasi nilai tersebut terhadap performa *classifier* dapat dilihat pada **Tabel 2.7** [24]**.**

Tabel 2.7 Interpretasi Nilai Koefisien Kappa

Kappa coefficient	**Interpretasi**
< 0	*Less than chance performance*
0.01 – 0.20	*Slightly good*

0.21 – 0.40	*Fair performance*
0.41 – 0.60	*Moderate performance*
0.61 – 0.80	*Substantially good performance*
0.81 – 1.00	*Near perfect performance*

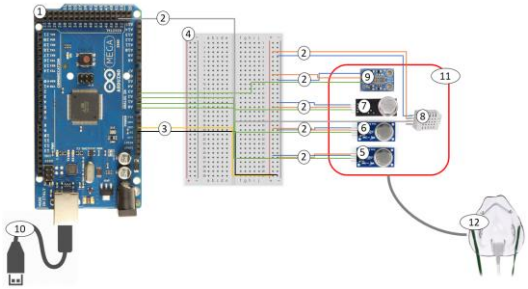
BAB III
METODOLOGI

Bab ini menjelaskan metodologi yang digunakan dalam pengerjaan Tugas Akhir. Untuk mencapai tujuan Tugas Akhir ini ada beberapa tahapan yang harus dilakukan. Pertama, membuat sebuah *Electronic Nose* (*e-Nose*) yang digunakan untuk mengukur kandungan konsentrasi *biomarker* dalam napas pasien. Kedua, pengumpulan data *ground truth* pasien sehat dan pasien diabetes menggunakan *e-Nose*. Ketika pengujian perfroma *e-Nose* dalam mendeteksi *biomarker* dalam napas pasien.

3.1. Perangkaian *Electronic Nose* (e-Nose)

Penciuman elektronik atau *Electronic Nose* (e-Nose) merupakan sebuah instrument yang digunakan untuk mendeteksi bau atau aroma. Dalam tugas akhir ini, *e-Nose* dibuat dengan tujuan untuk mengetahui kandungan beberapa *biomarker,* yaitu karbon monoksida (CO), karbon dioksida (CO_2), ketone, temperatur, kelembapan, dan *volatile organic compound* (VOC) dalam napas pasien untuk membedakan pasien sehat dan pasien diabetes berdasar perbedaan konsentrasi *biomarker* dalam napas.

Desain *e-Nose* yang dibuat dapat dilihat pada Error! Reference source not found.**.** Alat-alat yang dibutuhkan untuk membuat *e-Nose* dapat dilihat pada **Tabel 0.1.**



Gambar 3.1 Desain e-Nose untuk Deteksi Diabetes

Tabel 0.1 Daftar Peralatan untuk Membuat *e-Nose*

No.	Nama Alat	Jumlah	Fungsi
1	ARDUINO MEGA 2560	1x	Mikrokontroler untuk disambungkan ke sensor
2	Kabel Female to Male	8x	menyambungkan sensor ke mikrokontroler melalui *breadboard*
3	Kabel Male to Male	2x	Menyambungkan tegangan dari mikrokontroler ke *breadboard*
4	Breadboard 400 Tie Point Project Board	1x	meletakkan kabel yang menghubungkan sensor dan mikrokontroler
5	Sensor MQ-7	1x	Deteksi kandungan CO
6	Sensor MQ-135	1x	Deteksi kandungan CO_2
7	Sensor MQ-138	1x	Deteksi kandungan keton
8	Sensor DHT-22	1x	Pengukuran suhu dan kelembapan
9	Sensor MiCS-5524	1x	Deteksi kandungan VOC
10	Kabel USB	1x	Menghubungkan mikrokontroler dan komputer
11	Kotak Penampung	1x	Tempat peletakan sensor dan penampung napas pasien sementara
12	Masker Oksigen	20x	Menyalurkan napas pasien kedalam kotak penampung

Untuk membuat koneksi antara mikrokontroler dan sensor-sensor gas serta sensor suhu-kelembapan maka diperlukan kabel yang terhubung ke *pin* pada sensor dan *pin* pada mikrokontroler. *Breadboard* digunakan agar dapat memberikan tegangan ke semua sensor dikarenakan hanya terdapat satu *pin*

tegangan (Vcc) dan satu pin GND pada mikrokontroler sedangkan masing-masing sensor harus melakukan koneksi *pin* Vcc dan GND pada mikrokontroler. Daftar koneksi pin pada *breadboard* ke mikrokontroler dan pin pada sensor ke mikrokontroler melalui *breadboard* dapat dilihat pada **Tabel 0.2** dan **Tabel 0.3.**

Tabel 0.2 Koneksi Pin pada BreadBoard ke Mikrokontroler

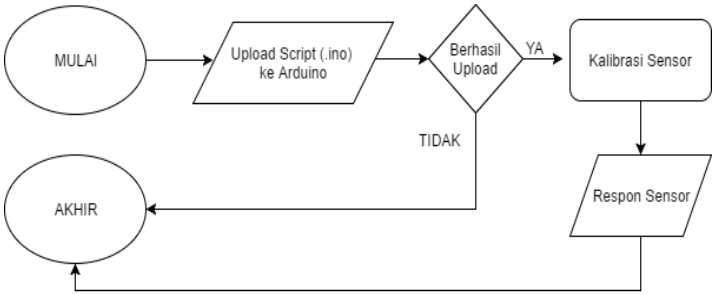
Pin pada *BreadBoard*	**Pin pada Mikrokontroler**
+	5V
-	GND

Tabel 0.3 Koneksi Pin pada Sensor ke Mikrokontroler

Pin pada Sensor *Biomarker*	**Pin pada Mikrokontorler**
VCC	+ (pada *breadboard*)
GND	- (pada *breadboard*)
AO MQ-7	A0
AO MQ-135	A1
AO MQ-138	A2
AO MiCS-5524	A3
Pin pada Sensor Suhu-Kelembapan	**Pin pada Mikrokontroler**
VCC	+ (pada *breadboard*)
GND	- (pada *breadboard*)
DO	D52

3.2. Pengujian Performa *Electronic Nose* (e-Nose)

Setelah perangkaian *e-Nose* telah selesai maka dilakukan uji performa *e-Nose* untuk mengetahui apakah sensor-sensor yang telah dipasang sensitif terhadap perubahan *biomarker* (CO, CO_2, ketone, VOC) dan perubahan suhu-kelembapan dalam ruang sampel napas. Lagkah pengujian performa *e-Nose* dapat dilihat pada *flowchart* di **Gambar 3.2**.



Gambar 3.2 *Flowchart* Uji Performa e-Nose

Script Arduino ditulis menggunakan Bahasa Arduino dengan ekstensi file .ino. *Baud Rate* yang digunakan adalah 9600. *Script* ini berisi perhitungan untuk:

1. Kalibrasi sensor-sensor analog seperti pada **Kode Sumber 0.1.** Nilai kalibrasi masing-masing sensor adalah nilai resistansi sensor pada udara bersih (R_O).
2. Perhitungan konsentrasi gas dari nilai respon sensor menggunakan rumus (2.1), (2.2), (2.4) seperi pada **Kode Sumber 0.2**, **Kode Sumber 0.3**, **Kode Sumber 0.4**
3. Keluaran respon sensor asli (ADC) dan setelah perhitungan konsentrasi (ppm) seperti pada **Gambar 3.3** dan **Gambar 3.4**.

```
1.  float MQCalibration(int mq_pin, float RO_CLEAN_AIR_FACTOR)
2.  {
```

```
3.    int i;
4.    float val=0;
5.    for (i=0;i<CALIBARAION_SAMPLE_TIMES;i++) {
6.      val += MQResistanceCalculation(analogRead(mq_pin));
7.      delay(CALIBRATION_SAMPLE_INTERVAL);
8.    }
9.    val = val/CALIBARAION_SAMPLE_TIMES;
10.   val = val/RO_CLEAN_AIR_FACTOR;
11.   return val; }
```

Kode Sumber 0.1 Kalibrasi Sensor Analog

```
1.  float resistanceOfSensor(float vc_board, float rl,
    int adc)
2.  {
3.    double vrl = getVoltageSensor(adc, vc_board);
4.    double rs  = (vc_board-vrl)/vrl*rl;
5.
6.    return rs;
7.  }
```

Kode Sumber 0.2 Perhitungan Nilai Resistansi Sensor R_s Sebenarnya (2.1)

```
1.  double getVoltageSensor(int adc, int vc_board)
2.  {
3.    double vrl = (float) (adc*vc_board)/1023.0;
4.
5.    return vrl; }
```

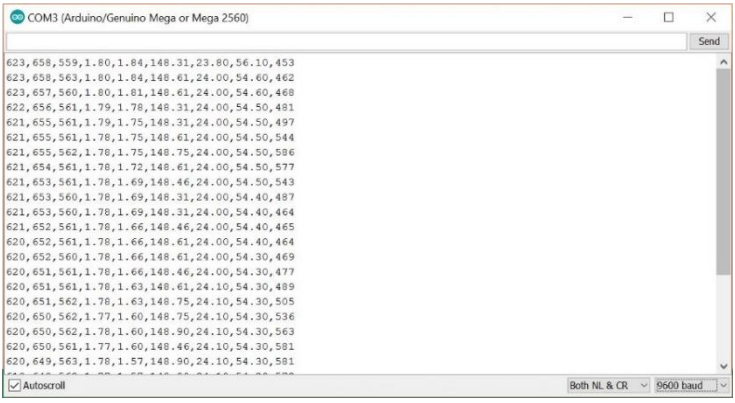
Kode Sumber 0.3 Perhitungan Nilai Tegangan Sensor V_{RL} (2.2)

```
1.  float getGasPercentage(float a, float b, float c, f
    loat Ro, float Rs)
2.  {
3.    float power = (float)log10(((Rs/Ro)-c)/a)/b;
4.    float ppm = (float)pow(10,power);
5.
6.    return ppm;
7.  }
```

Kode Sumber 0.4 Perhitungan Konsentrasi Gas Tiap Sensor *C* (2.4)

Raw_ co	Raw_ co2	Raw_ket one	PPM _co	PPM_ co2	PPM_ket one	Su hu	Kelemba pan	Raw_ voc

Gambar 3.3 Skema Keluaran Hasil dari *Running Script*



Gambar 3.4 Keluaran dari *Running Script* pada Serial Monitor

Keluaran dari setiap sensor memberikan respon dan nilai yang baik, maka dari itu *e-Nose* sudah siap digunakan untuk pengambilan data.

3.3. Pengumpulan Data *Ground Truth*

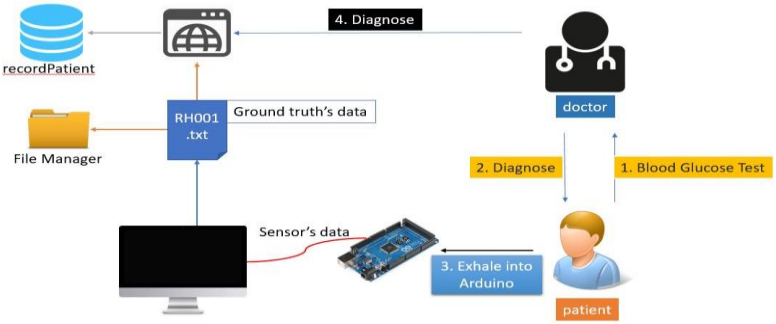
Ground truth data diperlukan untuk mendapatkan data latih (*training set*) yang nantinya juga akan digunakan sebagai data pengujian (*testing set*) menggunakan *cross-validation. Ground truth* data diambil pada,

Rentang Waktu : 8 Mei 2017 – 20 Mei 2017

Tempat : **Puskesmas Kedungdoro Surabaya**
Jalan Kaliasin Pompa No. 79 - 81, Kedungdoro, Tegalsari, Kota SBY, Jawa Timur 60261

Pembimbing : dr. Rr Endang Dwihastutiningsih dan dr. Pipiet Paramita

Prosedur pengambilan data pasien dapat dilihat pada **Gambar 3.5**. Perlu diingat kembali, dalam penelitian ini data *ground truth* diklasifikasikan sebagai sehat atau diabetes dengan asumsi pasien diabetes adalah pasien dengan kadar gula lebih dari 150 mg/dl dan pasien sehat adalah pasien dengan kadar gula dibawah 120 mg/dl. Pasien yang dijadikan *ground truth* adalah pasien dengan hasil cek gula darah acak. Data yang terkumpul sejumlah 40 data *ground truth* dengan rincian 20 orang adalah pasien acak sehat dan 20 orang pasien acak diabetes.



Gambar 3.5 Prosedur Pencatatan Data Pasien sebagai *Ground Truth*

Langkah-langkah yang dilakukan untuk pencatatan data *ground truth* seperti yang digambarkan pada **Gambar 3.5** adalah sebagai berikut:

1. Pasien memberikan hasil cek darah kepada dokter yang salah satu informasinya mengandung kadar gula darah pasien.
2. Dokter memberikan diagnosa (sehat atau diabetes) ke pasien berdasar hasil cek gula darah acak.
3. Pasien diminta untuk meniupkan napasnya ke *e-Nose* selama **150 detik** untuk diambil datanya sebagai *ground truth* menggunakan masker oksigen yang terhubung ke kotak penampung sensor dan napas.
4. Arduino mengirimkan data hasil respon sensor ke komputer menggunakan USB.

5. Data hasil respon sensor dan konsentrasi gas dalam napas pasien dicatat oleh *e-Nose* dalam bentuk *text file* (.txt) dengan format data *comma separated value* (csv) seperti pada **Gambar 3.4.**
6. Data *text file* respon sensor dan konsentrasi gas beserta identitas singkat, foto pasien dan diagnose pasien *ground truth* di unggah ke *website* agar data *ground truth* tercatat ke dalam basis data.

BAB IV
ANALISIS DAN PERANCANGAN SISTEM

Bab ini membahas tahap analisis dan perancangan sistem yang akan dibangun. Analisis membahas semua persiapan yang akan menjadi pokok pikiran pembuatan aplikasi ini. Mulai dari masalah yang melatar belakangi, analisis perbedaan respon sensor dan konsentrasi gas dalam napas pasien sehat dan pasien diabetes hingga gambaran awal sistem yang akan dibuat. Perancangan sistem membahas hal-hal yang berkaitan dengan pondasi atau dasar pembuatan aplikasi, yang meliputi perancangan basis data, tampilan antar muka halaman aplikasi, hingga perancangan alur proses yang akan diimplementasikan di dalam aplikasi.

4.1. Analisis

Tahap analisis meliputi analisis masalah, analisis perbedaan respon sensor dan konsentrasi gas terhadap napas pasien sehat dan pasien diabetes, analisis kebutuhan, deskripsi umum sistem, dan kasus penggunaan sistem yang dibuat.

4.1.1. Analisis Permasalahan

Diabetes adalah penyakit metabolik yang banyak di derita manusia mulai dari rentang umur muda sampai dewasa (30 – 80 tahun). Indonesia sendiri masuk dalam daftar teratas negara yang memiliki jumlah pasien diabetes terbesar. Sayangnya, dari jumlah yang besar tersebut sebagai tidak mengetahui bahwa mereka mengidap penyakit diabetes karena tingkat *awareness* orang Indonesia terhadap penyakit yang tergolong rendah. Apalagi, cara yang umum untuk mengetahui apakah seseorang mengidap penyakit diabetes adalah dengan pengambilan sampel darah di laboratorium untuk mengetahui apakah kadar gula darah pasien masih dalam ambang normal. Hal inilah yang dianggap sebagai salah satu masalah yang mendasari, yaitu pemeriksaan diabetes dirasa masih susah dengan harus datang ke laboratorium dan sakit

karena sampel darah diambil secara *invasive* dan memakan biaya yang tidak murah.

Oleh karena itu, penelitian ini diajukan untuk membuat sebuah sistem yang dapat memberikan diagnosis kepada pasien untuk mengetahui apakah pasien tersebut tergolong ke dalam pasien sehat atau pasien diabetes dengan cara *non-invasive* dan murah yaitu analisa napas manusia sehingga dapat dilakukan tindakan *preventif* dini.

4.1.2. Analisis Perhitungan Konsentrasi Gas terhadap Respon Sensor

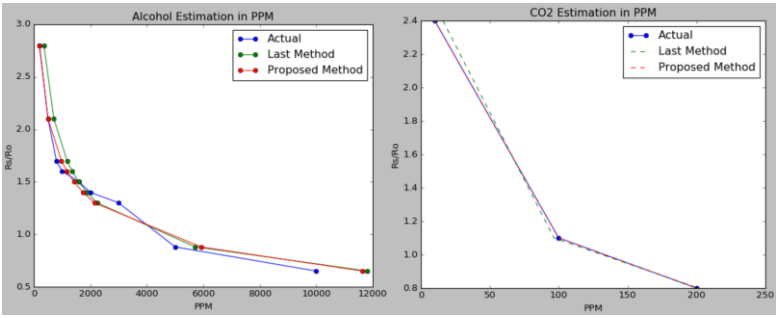
Perhitungan konsentrasi gas terhadap respon sensor didapatkan dari persamaan (2.3) atau (2.4). Pada sub-bab ini akan dilakukan perbandingan estimasi konsentrasi gas yang dari kedua persamaan terhadap data *ground truth* yang ada pada *datasheet* masing-masing sensor. Parameter-parameter yang digunakan pada persamaan (2.3) dan (2.4) dapat dilihat pada **Tabel 2.5**. Hasil perbandingan *error* dari kedua persamaan dapat dilihat pada **Tabel 0.1.**

Tabel 0.1 Perbandingan *Error* Estimasi Konsentrasi Gas terhadap Respon Sensor dari Hasil Perhitungan (2.3) dan (2.4)

Sensor	Gas	Equation (2.3)		Equation (2.4)	
		Standard Error	Adj. R^2	Standard Error	Adj. R^2
MQ-2	Alch.	0.1001	0.9752	0.09884	0.9759
MQ-7	CO	0.0407	0.9967	0.03045	0.9982
MQ-135	CO_2	0.04861	0.9967	0.0212	1
MQ-138	Ketone	0.03376	0.8408	0.01878	0.9507

Maka dari itu, dikarenakan persamaan (2.4) menghasilkan *error* yang lebih kecil dan nilai *adjusted* R^2 yang lebih besar maka pada penelitian ini untuk perhitungan konsentrasi gas dalam napas pasien digunakan persamaan (2.4). Beberapa hasil *plot* estimasi

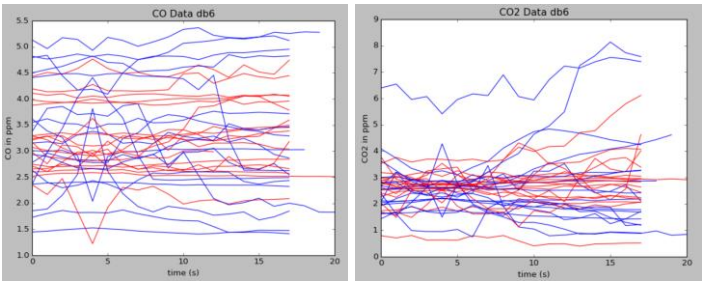
persamaan (2.3) dan (2.4) terhadap respon sensor pada *datasheet* dapat dilihat pada Error! Reference source not found..



Gambar 4.1 Hasil Estimasi (2.3) dan (2.4) terhadap *Datasheet*

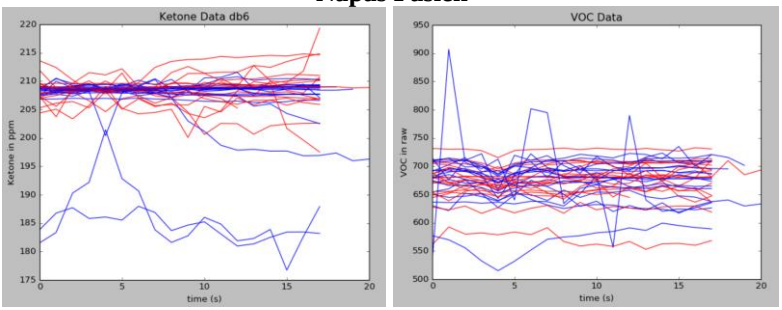
4.1.3. Analisis Perbedaan Respon Sensor dan Konsentrasi Gas terhadap Napas Pasien Sehat dan Pasien Diabetes

Ada empat *biomarker* yang akan di deteksi, yaituL CO, CO_2, ketone, dan VOC dan suhu serta kelembapan. Setiap *biomarker* yang di deteksi akan dicari empat fitur statistiknya, yaitu: nilai rata-rata, nilai maksimum, nilai minimum, dan standar deviasai. Sebelum mencari empat fitur statistic tersebut, dapat dilihat pada Gambar 4.2 respon sensor terhadap napas pasien diabetes (merah) dan pasien sehat (biru).



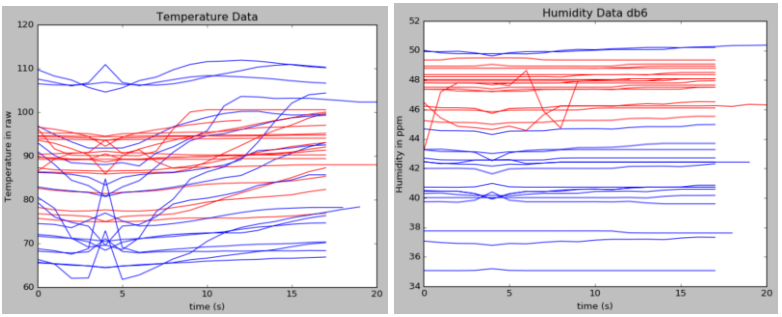
(a) (b)

Gambar 4.2 Respon Sensor terhadap (a) CO dan (b) CO2 dalam Napas Pasien



(a) (b)

Gambar 4.3 Respon Sensor terhadap (a) Ketone dan (b) VOC dalam Napas Pasien



(a) (b)

Gambar 4.4 Respon Sensor terhadap (a) Suhu dan (b) Kelembapan dalam Napas Pasien

Secara visual dapat diberikan hipotesa awal bahwa fitur statistik yang dapat membedakan pasien diabetes dan pasien sehat dari respon sensor terhadap napas adalah nilai rata-rata CO (karena kebanyakan respon sensor pasien diabetes ada di rentang atas dan pasien sehat di rentang bawah), standar deviasi CO_2 (karena sinyal respon sensor pasien sehat rata sedangkan pasien diabetes bergerigi), standar deviasi ketone (karena sinyal respon sensor pasien sehat rata dan pasien diabetes bergerigi), standar deviasi VOC dan nilai rata-rata kelembapan (karena kebanyakan sinyal respon kelembapan pasien diabetes berada di rentang atas dan pasien sehat berada di rentang bawah).

Untuk mengetahui atribut mana yang paling relevan terhadap kelas *ground truth* maka digunakan persamaan (2.13) untuk mengukur nilai *chi-squared statistic* masing-masing atribut. Setelah dijalankan *weighting by chi-squared statistic* maka didapatkan enam atribut terbaik seperi pada **Tabel 0.2.**

Tabel 0.2 Atribut Terpilih dengan Nilai *Chi-Squared Statistic* Terbesar untuk Masing-Masing Gas

Attributes	Chi-Squared Statistic Value
AVG_HUMID	32
AVG_CO	10.7
STD_CO	10.5
STD_CO2	9.5
STD_KETONE	9
STD_VOC	7

Oleh karena itu, sebelumya vektor atribut *ground truth* memiliki panjang 24 atribut dari masing-masing empat fitur statistik gas, tetapi setelah dijalankan proses seleksi fitur maka vector atribut *ground truth* memiliki panjang menjadi enam atribut. Analisis Kebutuhan Sistem

Kebutuhan utama penelitian ini adalah pengumpulan data *ground truth* untuk dijadikan data latih dan diagnosis data napas

pasien baru yang belum terkategori. Secara rinci, daftar kebutuhan fungsional perangkat lunak yang dibangun dapat dilihat pada **Tabel 0.3.**

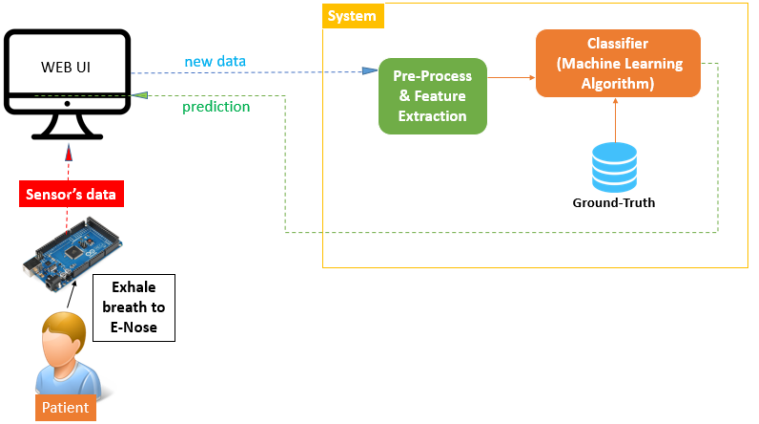
Tabel 0.3 Daftar Kebutuhan Fungsional Perangkat Lunak

Kode Kebutuhan	Kebutuhan Fungsional	Deskripsi
F-01	Sistem memungkinkan administrator mencatat data *ground truth*/pasie baru	Admin dapat memasukkan identitas pasien dan data sensor pasien sebagai *ground truth* maupun sebagai pasien baru
F-02	Sistem memungkinkan administrator mengubah data *ground truth*/pasien baru	Admin dapat mengubah data pasien baru atau *ground truth* yang telah tercatat
F-03	Sistem memungkinkan administrator menghapus data *ground truth*/pasien baru	Admin dapat menghapus data pasien baru atau *ground truth* yang telah tercatat
F-04	Sistem dapat menampilkan seluruh data pasien yang telah tercatat	Admin dapat melihat seluruh data pasien atau *ground truth* yang tercatat

F-05	Sistem dapat menampilkan profil pasien dan melakukan klasifikasi data pasien	Admin dapat mengetahui hasil klasifikasi data *ground truth* melalui *cross-validation* maupun data pasien baru

4.1.4. Deskripsi Umum Sistem

Aplikasi yang akan dibuat pada Tugas Akhir ini adalah program aplikasi berbasis web. Secara garis besar, jalannya alur penggunaan sistem untuk deteksi diabetes secara dini dapat dijabarkan dalam **Gambar 4.5.**



Gambar 4.5 Gambaran Besar Sistem Deteksi Diabetes

Berdasarkan **Gambar 4.5** maka akan dijelaskan dengan rinci mulai dari *input* (sensor data di unggah ke web), *pre-processing*, ekstraksi fitur, dan *output* yang berupa prediksi atau diagnosis. Detail runtutan aplikasi dari *input* hingga *output* akan dijelaskan pada subbab berikut:

4.1.4.1. *Input*

Untuk melakukan diagnosis maka presodur yang dilakukan hampir mirip dengan **Gambar 3.5**, yaitu pencatatan data respon sensor terhadap napas pasien selama 150 detik dengan timer kemudian identitas singkat pasien dan data sensor napas pasien tersebut di unggah melalui *website*. Input sistem merupakan data sensor napas pasien yang diunggah ke *website*.

4.1.4.2. Proses

Proses yang terjadi pada sistem ditangani oleh PHP dan Python. HTML, PHP, dan Javascript memiliki fungsi utama yaitu untuk melakukan koneksi dengan *database*, menerima masukan dan menampilkan data sedangkan Python digunakan untuk melakukan perhitungan dan tahap-tahap pengolahan data. Perhitungan dan pengolahan data yang dimaksud adalah *preprocessing* dan fiktur ekstraksi serta klasifikasi data. Detail dari perancangan ekstraksi fitur, *preprocessing* dan klasifikasi data akan dibahas pada sub-bab berikut.

A. Ekstraksi Fitur

Setelah data respon sensor dan konsentrasi gas dalam napas pasien diunggah ke *website*, maka dilakukan ekstraksi fitur statistik untuk setiap gas. Keempat fitur statistic tersebut adalah nilai rata-rata, nilai maksimal, nilai minimum, dan nilai standar deviasi. Akan tetapi pada sub-bab 4.1.3 setelah proses seleksi fitur maka vektor atribut yang digunakan hanya nilai rata-rata CO, nilai rata-rata kelembapan, nilai standar deviasi CO_2, nilai standar deviasi ketone dan nilai standar deviasi VOC. Vektor atribut data *ground truth* setelah seleksi fitur dapat dilihat pada **Gambar 4.6.**

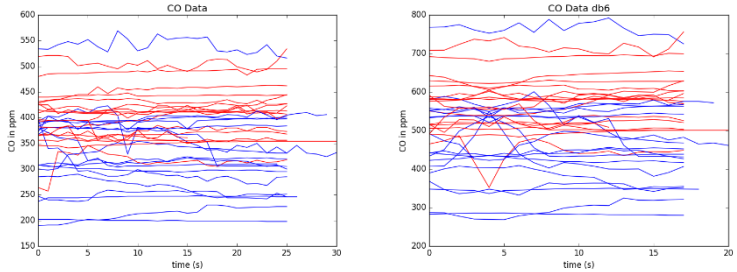
I	avgC	stdC	stdKeto	stdC	avgHum	stdVO	Clas

D	O	O^2	ne	O	id	C	s

Gambar 4.6 Vektor Atribut Data *Ground Truth* setelah Seleksi Fitur

B. *Preprocessing*

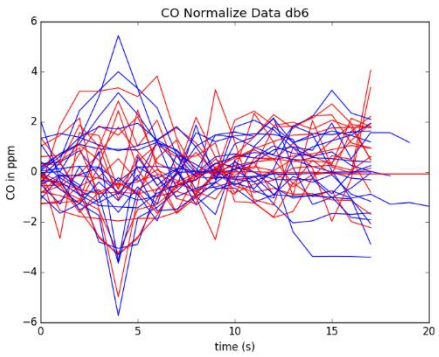
Sebelum dilakukan ekstraksi fitur data sinyal respon dibersihkan dahulu dari *noise* dengan menggunakan *Discrete Wavelet Transform* dan dilakukan *feature scaling* menggunakan Z-*score normalization*. Sinyal yang di *denoising* adalah sinyal mentah (asli) respon sensor terhadap gas yang telah dikonveri dalam bentuk ppm. Hasil yang didapat pada *denoising* adalah koefisin aproksimasi pada level 1 DWT. Setelah dilakukan *denoising* maka dilakukan normalisasi untuk koefisien aproksimasi untuk menstandarkan skala tiap atribut menggunakan Z-Score. Berikut dapat dilihat pada **Gambar 4.7** perbedaan sinyal respon karbon monoksida terhadap napas pasien sebelum dan sesudah *denosing*.



(a) (b)

Gambar 4.7 Perbedaan Sinyal Respon terhadap Kandungan CO dalam Napas Pasien (a) sebelum *denoising* (b) setelah *denoising*

Dapat dilihat bahwa sinyal respon setelah *denoising* lebih *smooth*. Setelah dilakukan *denoising* sinyal selanjutnya adalah *feature scaling* untuk menyamakan skala tiap atribut. Hasil respon sensor terhadap sinyal CO yang telah di *denoising* setelah normalisasi dapat dilihat pada **Gambar 4.8.**



Gambar 4.8 Sinyal Respon CO *denoising* yang Telah di Normalisasi

C. Klasifikasi

Data latih (*training set*) dan data pengecekan (*testing set*) didapatkan dari data *ground truth* yang telah terkumpul dengan menggunakan *cross-validation*. Klasifikasi yang akan diimplementasikan adalah k-NN yang akan mengukur *similiraty* data baru pasien dengan data *ground truth* kemudian melakukan *voting* dan *voting* terbanyak dari suatu kelas akan menentukan kelas atau diagnosis pasien baru. Dalam beberapa kasus jika jumlah *voting* kedua kelas berjumlah sama, maka diambil *voting* yang mempunyai nilai akumulasi jarak paling kecil.

4.1.4.3. *Output*

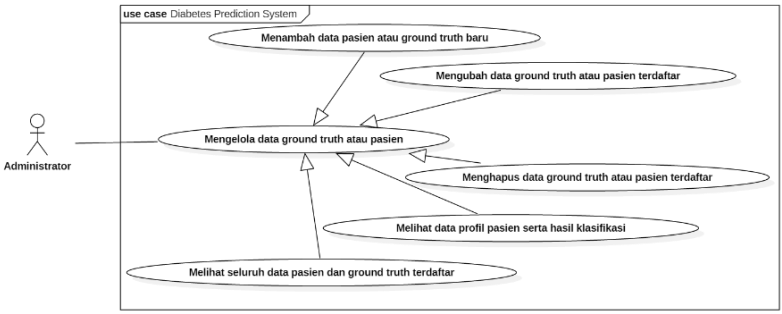
Output atau prediksi dari sistem berupa kelas pasien yaitu *healthy* (sehat) atau *chronic* (diabetes) yang dihasilkan oleh *classifier*.

4.1.5. Kasus Penggunaan

Sesuai pada spesifikasi kebutuhan fungsional yang telah dipaparkan, maka akan disusun tabel kasus penggunaan, yang diharapkan dapat memenuhi kebutuhan fungsional. Kasus penggunaan dijelaskan lebih lanjut pada Tabel 0.4 dan diagram kasus penggunaan ditunjukkan pada Gambar 4.9.

Tabel 0.4 Daftar Kode Kasus Penggunaan

Kode Kasus Penggunaan	Nama	Aktor
UC-1	Menambah data pasien atau *ground truth* baru	Administrator
UC-2	Melihat data profil pasien serta hasil klasifikasi	Administrator
UC-3	Mengubah data *ground truth* atau pasien terdaftar	Administrator
UC-4	Menghapus data *ground truth* ata pasien terdaftar	Administrator
UC-5	Melihat seluruh data pasien dan *ground truth*	Administrator



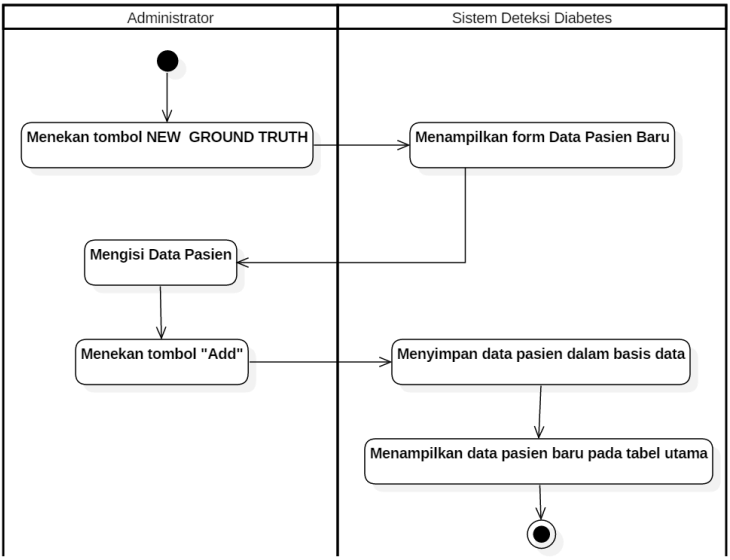
Gambar 4.9 Diagram Kasus Penggunaan

4.1.5.1. Menambah Data Pasien atau *Ground Truth* Baru (UC-1)

Pada kasus penggunaan ini, administrator dapat memasukkan data pasien atau *ground truth* baru melalui sistem. Spesifikasi kasus penggunaan dapat dilihat pada Tabel 0.5 dan diagram aktivitas pada Gambar 4.10.

Tabel 0.5 Spesifikasi Kasus Penggunaan UC-1

Kode	UC-1
Nama	Menambah Data Pasien atau *Ground Truth* Baru
Aktor	Administrator
Deskripsi	Admin memasukkan identitas pasien dan data sensor pasien sebagai *ground truth* maupun sebagai pasien baru
Tipe	Fungsional
Kondisi Awal	Data pasien baru atau *ground truth* belum tercatat
Kondisi Akhir	Sistem menambahkan data *ground truth* atau pasien baru pada tabel utama
Alur Kejadian Normal	
1. Sistem menampilkan halaman utama web 2. Administrator menekan tombol "NEW GROUND-TRUTH" 3. Sistem menampilkan form isian data pasien 4. Administrator mengisi data pasien dan menekan tombol "Add" 5. Sistem menyimpan data pasien tersebut dan menampilkan pada tabel utama	
Alur Kejadian Alternatif	
Tidak Ada.	



Gambar 4.10 Diagram Aktivitas UC-1

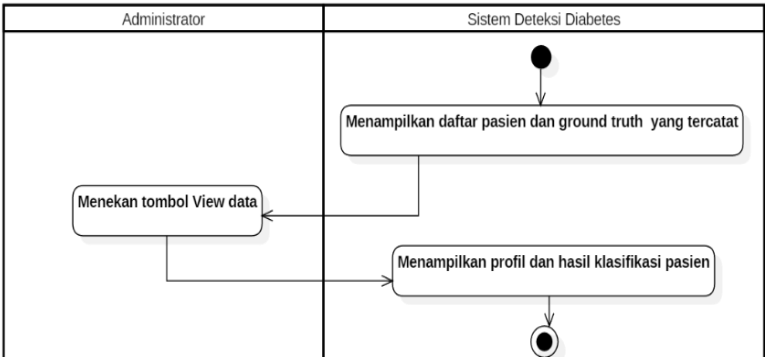
4.1.5.2. Melihat Data Profil Pasien serta Hasil Klasifikasi (UC-2)

Pada kasus penggunaan ini, administrator dapat melihat data profil pasien yang dipilih serta klasifikasi pasien oleh *classifier*. Spesifikasi kasus penggunaan dapat dilihat pada Tabel 0.6 dan diagram aktivitas pada Gambar 4.11.

Tabel 0.6 Spesifikasi Kasus Penggunaan UC-2

Kode	UC-2
Nama	Melihat data profil pasien serta hasil klasifikasi
Aktor	Administrator
Deskripsi	Admin dapat melihat profil pasien atau *ground truth* yang tercatat dalam sistem
Tipe	Fungsional

Kondisi Awal	Administrator tidak mengetahui hasil klasifikasi dan profil pasien
Kondisi Akhir	Sistem menampilkan data profil pasien serta hasil klasifikasi
Alur Kejadian Normal	
1. Sistem menampilkan halaman utama web berisi daftar seluruh pasien atau *ground truth* 2. Administrator menekan tombol "view" disebelah identitas singkat pasien yang ingin dilihat profilnya 3. Sistem menampilkan halaman profil pasien yang berisi identitas singkat serta hasil klasifikasi oleh *classifier*	
Alur Kejadian Alternatif	
Tidak Ada.	



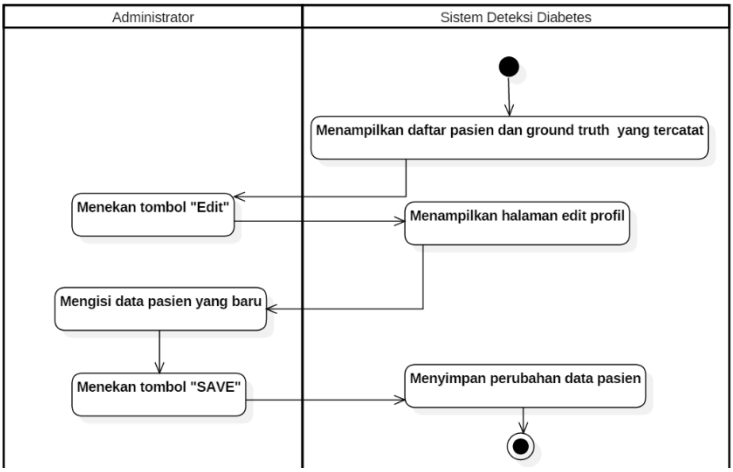
Gambar 4.11 Diagram Aktivitas UC-2

4.1.5.3. Mengubah Data *Ground Truth* atau Pasien Terdaftar (UC-3)

Pada kasus penggunaan ini, administrator dapat mengubah data pasien atau *ground truth* yang dipilih yang ada pada tabel utama. Spesifikasi kasus penggunaan dapat dilihat pada Tabel 0.7 dan diagram aktivitas pada Gambar 4.12.

Tabel 0.7 Spesifikasi Kasus Penggunaan UC-3

Kode	UC-3
Nama	Mengubah data *ground truth* atau pasien terdaftar
Aktor	Administrator
Deskripsi	Admin dapat mengubah data profil pasien atau *ground truth* yang tercatat dalam sistem
Tipe	Fungsional
Kondisi Awal	Data profil pasien belum *ter-update* pada basis data
Kondisi Akhir	Data profil pasien ter-*update* pada basis data
Alur Kejadian Normal	
1. Sistem menampilkan halaman utama web berisi daftar seluruh pasien atau *ground truth* 2. Administrator menekan tombol "edit" disebelah identitas singkat pasien yang ingin dilihat profilnya 3. Sistem menampilkan halaman edit profil pasien 4. Administrator mengisikan data identitas pasien terbaru 5. Administrator menekan tombol "SAVE" 6. Data pasien berubah pada basis data	
Alur Kejadian Alternatif	
Tidak Ada.	



Gambar 4.12 Diagram Aktivitas UC-3

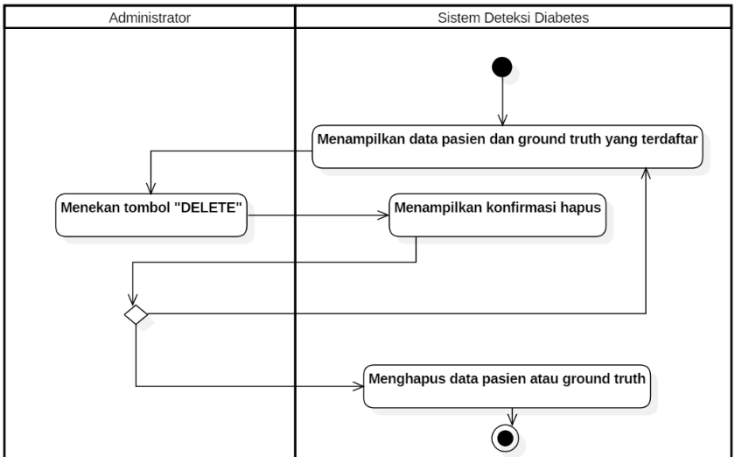
4.1.5.4. Menghapus Data *Ground Truth* atau Pasien Terdaftar (UC-4)

Pada kasus penggunaan ini, administrator dapat menghapus data pasien atau *ground truth* yang dipilih dari sistem. Spesifikasi kasus penggunaan dapat dilihat pada Tabel 0.8 dan diagram aktivitas dapat dilihat pada Gambar 4.13.

Tabel 0.8 Spesifikasi Kasus Penggunaan UC-4

Kode	UC-4
Nama	Menghapus data *ground truth* atau pasien terdaftar
Aktor	Administrator
Deskripsi	Admin dapat menghapus data profil pasien atau *ground truth* yang tercatat dalam sistem
Tipe	Fungsional

Kondisi Awal	Data profil pasien masih tercatat dalam basis data
Kondisi Akhir	Data profil pasien dihapus dari basis data
Alur Kejadian Normal	
1. Sistem menampilkan halaman utama web berisi daftar seluruh pasien atau *ground truth* 2. Administrator menekan tombol "delete" disebelah identitas singkat pasien yang ingin dilihat profilnya 3. Sistem menampilkan konfirmasi hapus "*You are about to delete. Do you want to procedd*?" 4. A. Administrator menekan "Delete" 5. Sistem menghapus data pasien dari basis data dan tabel utama	
Alur Kejadian Alternatif	
4.B. Administrator menekan tombol "Cancel"	



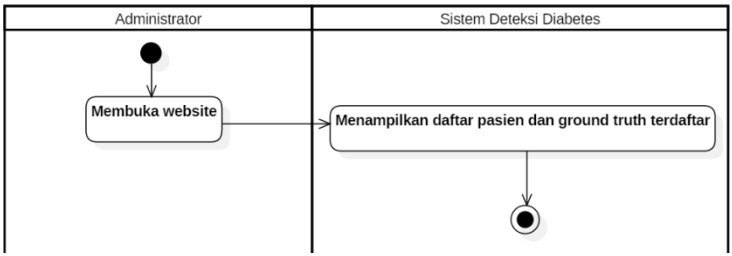
Gambar 4.13 Diagram Aktivitas UC-4

4.1.5.5. Melihat Seluruh Data Pasien dan *Ground Truth* Terdaftar (UC-5)

Pada kasus penggunaan ini, administrator dapat melihat seluruh data pasien dan *ground truth* yang tercatat dalam sistem. Spesifikasi kasus penggunaan dapat dilihat pada Tabel 0.9 dan Gambar 4.14.

Tabel 0.9 Spesifikasi Kasus Penggunaan UC-5

Kode	UC-5
Nama	Melihat seluruh data pasien *ground truth* terdaftar
Aktor	Administrator
Deskripsi	Admin dapat melihat seluruh data pasien dan *ground truth* yang terdaftar dalam sistem
Tipe	Fungsional
Kondisi Awal	Administrator tidak mengetahui pasien yang terdaftar pada sistem
Kondisi Akhir	Sistem menampilkan data pasien dan *ground truth* yang terdaftar dalam sistem
Alur Kejadian Normal	
1. Sistem menampilkan halaman utama web berisi daftar seluruh pasien atau *ground truth*	
Alur Kejadian Alternatif	
Tidak Ada.	



Gambar 4.14 Diagram Aktivitas UC-5

4.2. Perancangan Sistem

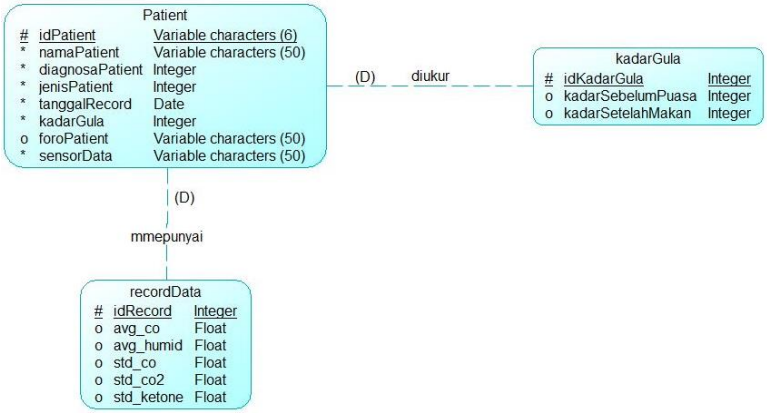
Tahap ini meliputi perancangan basis data, perancangan komponen sistem, dan perancangan alur proses penggunaan sistem yang diharapkan dapat memenuhi tujuan dari pengembangan aplikasi ini.

4.2.1. Perancangan Basis Data

Pada subbab ini dijelaskan mengenai perancangan basis data yang dalam hal ini digunakan untuk menyimpan data identitas pasien dan file respon sensor serta foto pasien. Basis data yang digunakan adalah basis data relasional (*Relational Database Management System*) dengan menggunakan MySQL 5.0. Rancangan basis data dalam bentuk konseptual dan fisik dapat dilihat pada sub-bab berikut.

4.2.1.1. *Conceptual Data Model*

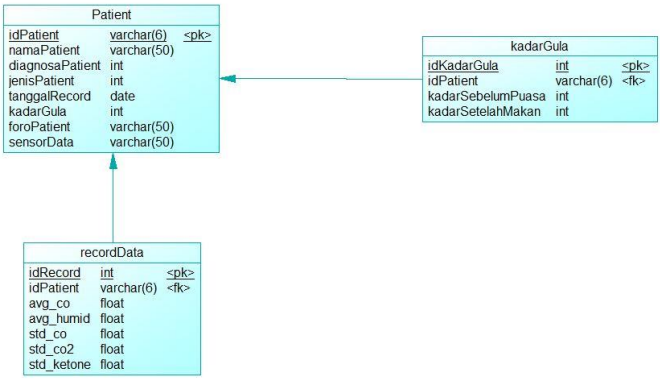
Rancangan basis data secara konseptual untuk menyimpan identitas pasien dan file respon sensor terhadap napas serta foto pasien dapat dilihat pada Gambar 4.15.



Gambar 4.15 *Conceptual Data Model* untuk Sistem Deteksi Diabetes

4.2.1.2. *Physical Data Model*

Rancangan basis data secara fisik yang siap diimplemnetasikan menjadi basis data dapat dilihat pada



Gambar 4.16 *Physical Data Model* untuk Sistem Deteksi Diabetes

4.2.1.3. Kamus Data

Terdapat tiga kamus data dari tiga tabel yang ada, yaitu:

1. Nama tabel : PATIENT
   Author : Administrator
   Primary Key : idPatient

Tabel 0.10 Kamus Data Tabel Patient

Nama Field	Deskripsi	Tipe & Panjang	Default	Keterangan
idPatient	Nomor registrasi data pasien	VARCHAR(6)	-	*Primary key*
namaPatient	Nama pasien	VARCHAR(50)	-	-
diagnosaPatient	Diagnosa pasien oleh dokter	INTEGER	1	Sehat (1) Ambang (2) Sakit (3)
jenisPatient	Jenis pasien yang dicatat	INTEGER	1	Acak (1) Puasa (2)
tanggalRecord	Tanggal data dimasukkan	DATE	TODAY	-
kadarGula	Kadar gula hasil periksa darah	INTEGER	1	-
fotoPatient	Nama file foto pasien	VARCHAR(50)	-	-
sensorData	Nama file sensor	VARCHAR(50)	-	-

	data			

2. Nama tabel : KADARGULA
   Author : Administrator
   Primary key : idKadarGula

Nama Field	Deskripsi	Tipe & Panjang	Default	Keterangan
idKadarGula	*Primary key* tabel kadarGula	INTEGER	-	*Primary key*, Auto_Increment
idPatient	ID pasien yang dicatat kadar gulanya	VARCHAR(6)	-	*Foreign key*
kadarSebelumPuasa	Kadar gula darah pasien sebelum puasa	INTEGER	-	-
kadarSetelahPuasa	Kadar gula darah pasien 2 jam setelah makan	INTEGER	-	-

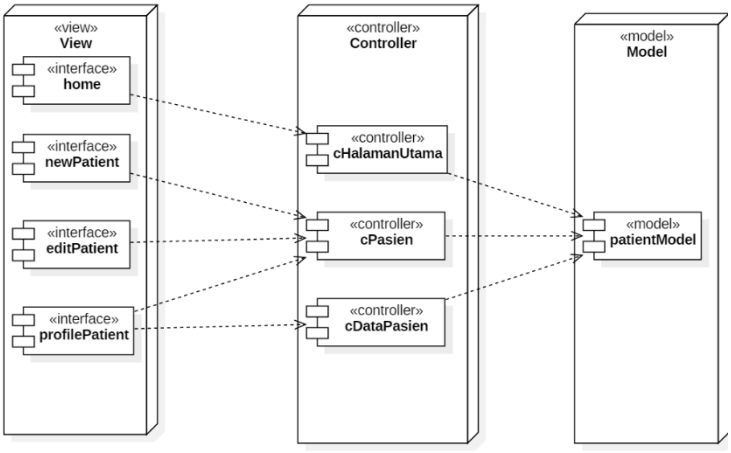
3. Nama Tabel : recordData
   Author : Administrator
   Primary key : idRecord

Nama Field	Deskripsi	Tipe & Panjang	Default	Keterangan
idRecord	*Primary key* tabel recordD	INTEGER	-	*Primary key*, Auto_Incr

	ata			ement
idPatient	ID pasien yang dicatat ekstraksi fitur sensornya	VARCHAR(6)	-	*Foreign key*
Avg_co	Nilai rata-rata respon CO	FLOAT	-	-
Avg_humid	Nilai rata-rata kelembapan	FLOAT	-	-
Std_ketone	Nilai standar deviasa ketone	FLOAT	-	-
Std_co2	Nilai standar deviasi co2	FLOAT	-	-
Std_voc	Nilai standar deviasi voc	FLOAT	-	-

4.2.2. Perancangan Komponen Sistem

Sistem dirancang menggunakan arsitektur Model, View, dan Controller (MVC) dengan bantuan CodeIgniter. Diagram komponen sistem dapat dilihat pada **Gambar 4.17.**

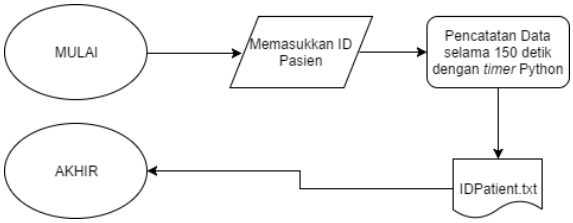


Gambar 4.17 Diagram Komponen MVC Sistem Deteksi Diabetes

4.2.3. Perancangan Proses

Pada tahap ini akan dijelaskan mengenai rancangan alur proses penggunaan aplikasi yang digunakan sebagai acuan dalam pembangunan aplikasi. Alur proses dimulai dari *input*, proses dan *output*.

4.2.3.1. *Input*



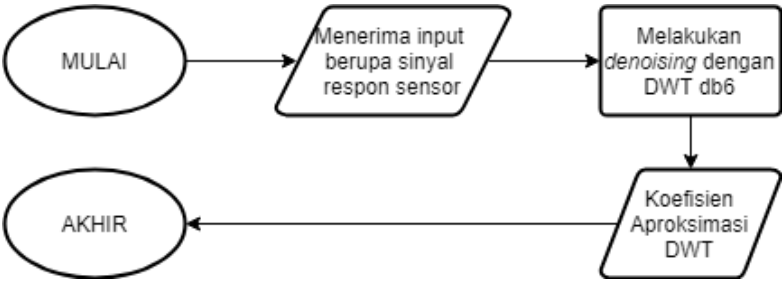
Gambar 4.18 Prosedur Pencatatan Data Napas Pasien dengan Timer Menggunakan Program Python

Pencatatan data menggunakan *e-Nose* dilakukan menggunakan program python yang memiliki *timer* seperi pada **Gambar 4.18** dengan langkah sebagai berikut:

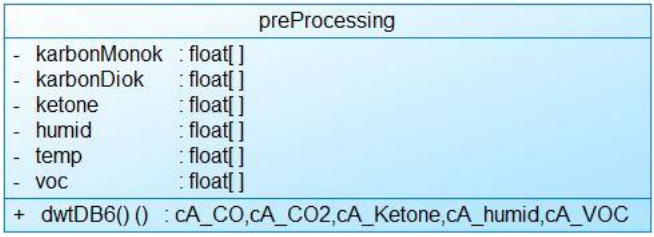
1. Memberi ID_Pasien, dengan aturan huruf pertama adalah jenis pasien yaitu: pasien acak (R) atau pasien puasa (F) diikuti dengan diagnosa pasien oleh dokter yaitu: sehat (H) atau diabetes (C) dan terakhir diikuti 3 digit nomor urut pasien. Contoh: Pak Tarmin adalah pasien gula darah acak yang di diagnosa oleh dokter sebagai orang diabetes dan Pak Tarmin adalah pasien acak dengan diagnosa diabetes yang di catat napasnya pada urutan ke dua. Maka ID_Pasien Pak Tarmin adalah RC001.
2. Memulai *timer* pencatatan data napas pasien dan program menampilkan kandungan gas dalam napas pasien secara *real time*.
3. Data kandungan gas dan respon sensor terhadap napas pasien tercatat dalam bentuk ID_Pasien.txt

4.2.3.1. *Preprocessing*

Pada tahap ini dilakukan proses *denosing* dengan menggunakan transformasi wavelet diskrit menggunakan *mother wavelet transform daubechies 6.* File yang telah didapatkan pada proses sebelumnya akan dibaca kemudian masing-masing respon sensor akan di *denoising* dengan dwt. Hasil yang didapatkan adalah koefisien aproksimasi dari transformasi wavelet karena diharapkan *noise* berada pada frekuensi yang lebih tinggi sehingga kita ambil frekuensi yang lebih rendah yang ada pada koefisien aproksimasi. Diagram alur dari proses ini dapat dilihat pada **Gambar 4.19** dan diagram kelas dari proses ini dapat dilihat pada **Gambar 4.20**.



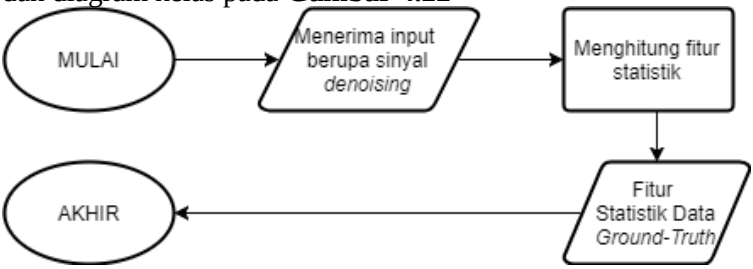
Gambar 4.19 Diagram Alur Proses '*Preprocessing*'



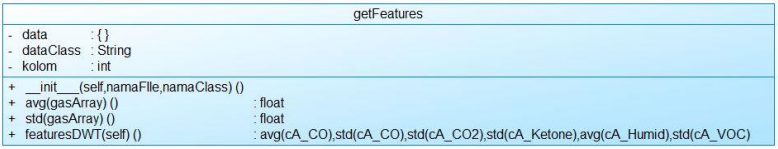
Gambar 4.20 Diagram Kelas Proses '*Preprocessing*'

4.2.3.2. Ekstraksi Fitur

Pada tahap ini dilakukan ekstraksi fitur, yaitu fitur statistik yang didapat setelah seleksi fitur pada tahap analisis. Seleksi fitur dilakukan terhadap sinyal *denoising* pada tahap sebelumnya, yaitu koefisien aproksimasi masing-masing sinyal sensor. Diagram alur proses ini dapat dilihar pada **Gambar 4.21** dan diagram kelas pada **Gambar 4.22**



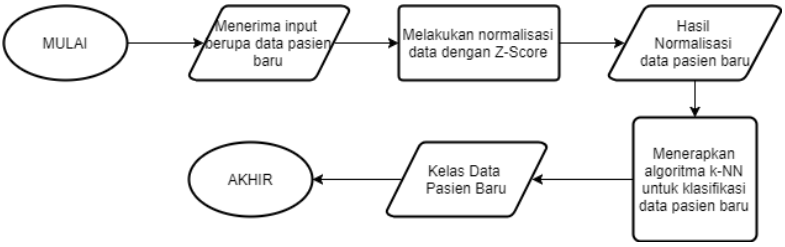
Gambar 4.21 Diagram Kelas Proses '*Ekstraksi Fitur*'



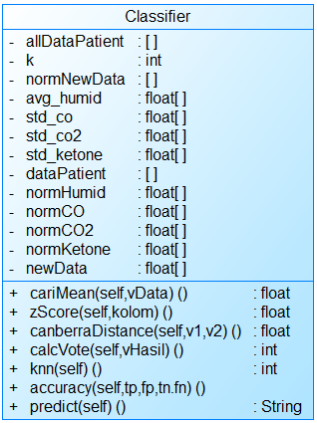
Gambar 4.22 Diagram Kelas Proses '*Ekstraksi Fitur*'

4.2.3.3. Klasifikasi

Pada tahap ini dilakukan klasifikasi data baru menggunakan model yang telah dibuat oleh classifier k-NN. *Classifier* yang digunakan untuk kasus ini adalah k-NN dengan menggunakan nilai k yang berbeda pada uji coba. Metode pengukuran jarak yang digunakan adalah *Canberra distance*. Model yang ada pada *ground truth* digunakan sebagai *training set* untuk melakukan prediksi data pasien baru. Pertama akan diterima inputan berupada data pasien baru, kemudian akan dilakukan normalisasi data dengan Z-Score. Setelah mendapatkan hasil normalisasi data maka algoritma *machine learning* akan membuat model dari tabel *ground truth* yang ada pada basis data. Setelah menerapkan algoritma klasifikasi makan dilakukan klasifikasi terhadap data pasien baru tersebut. Pada kelas diagram di proses ini juga terdapat fungsi untuk mencari nilai akurasi yang akan digunakan oleh proses evaluasi. Diagram alur dapat dilihat pada **Gambar 4.23** dan diagram kelas proses 'Klasifikasi' dapat dilihat pada **Gambar 4.24**.



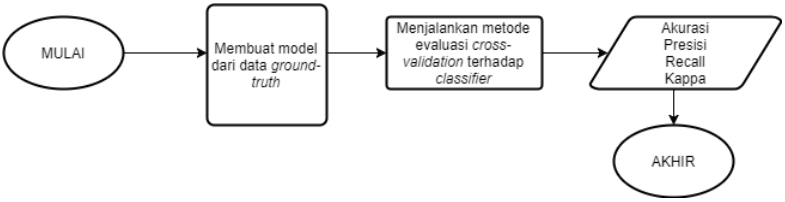
Gambar 4.23 Diagram Alur Proses '*Klasifikasi*'



Gambar 4.24 Diagram Kelas Proses '*Klasifikasi*'

4.2.3.4. Evaluasi

Pada tahap ini dilakukan evaluasi terhadap kinerja *classifier* yang digunakan dengan menggunakan beberapa *metric*. Metode evaluasi yang digunakan adalah *cross-validation* yaitu dengan membagi model yang dibuat menjadi beberapa ember dan jika data yang ada tidak cukup banyak maka dapat dilakukan metode *leave-one-out*. Setelah mendapatkan *confusion matrix* dari performa *classifier* maka dihitung nilai akurasi, presisi, recall dan kappa statistic untuk mengukur kinerja *classifier*. Diagram alur dapat dilihat pada **Gambar 4.25** dan fungsi *accuracy* pada diagram kelas pada **Gambar 4.24.**



Gambar 4.25 Diagram Alur Proses '*Evaluasi*'

BAB V
IMPLEMENTASI

Bab ini membahas implementasi dari analisis dan perancangan sistem yang telah dibahas pada Bab IV. Namun dalam penerapannya, rancangan tersebut dapat mengalami perubahan minor sewaktu-waktu apabila dibutuhkan.

5.1. Lingkungan Implementasi

Dalam implementasinya, lingkungan yang digunakan sama seperti yang dituliskan pada rancangan, yakni menggunakan beberapa perangkat pendukung sebagai berikut.

5.1.1. Lingkungan Implementasi Perangkat Keras

Perangkat keras yang digunakan dalam pengembangan aplikasi adalah sebagai berikut:

Jenis : Laptop
Tipe : HP x360 13
Prosesor : Intel ® Core™ i7-6500U CPU @ 2.50GHz (4 CPUs), ~2.6GHz
Memori : 8192MB RAM

5.1.2. Lingkungan Implementasi Perangkat Lunak

Perangkat lunak yang digunakan untuk implementasi aplikasi ini sebagai berikut:

- Arduino IDE: untuk memrogram *board Arduino* dan sensor
- PyCharm 2016.2.3: untuk memrogram *classifier*
- Bracket: untuk memrogram tampilan *web* sebagai GUI pengguna
- Rapidminer Studio: untuk melakukan uji coba *classifier* dari dataset

- WAMP Server: untuk *web server* lokal
- StarUML 2.8.0: untuk membuat UML
- SyBase PowerDesigner 16.5: untuk membuat rancangan basis data

dan Bahasa pemrograman:

- Bahasa Pemrograman Arduino: untuk *board Arduino*
- Python 2.7: untuk membuat *classifier*
- PHP, HTML, JavaScript, dan CSS: untuk membuat GUI berbasis *web*

5.2. Implementasi Basis Data

Sub-bab ini membahas tentang implementasi basis data yang telah dirancang dan dibahas pada Bab IV. Ada 3 implementasi tabel yang dapat dilihat pada Kode Sumber 0.1, Kode Sumber 0.2, Kode Sumber 0.3.

```
1.  create table PATIENT
2.  (
3.     IDPATIENT            varchar(6) not null,
4.     NAMAPATIENT          varchar(50) not null,
5.     DIAGNOSAPATIENT      int not null,
6.     JENISPATIENT         int not null,
7.     TANGGALRECORD        date not null,
8.     KADARGULA            int not null,
9.     FOROPATIENT          varchar(50),
10.    SENSORDATA           varchar(50) not null,
11.    primary key (IDPATIENT)
12. );
```

Kode Sumber 0.1 Implementasi Tabel PATIENT

```
1.  create table KADARGULA
2.  (
3.     IDKADARGULA          int not null,
4.     IDPATIENT            varchar(6),
5.     KADARSEBELUMPUASA    int,
6.     KADARSETELAHMAKAN    int,
```

```
7.     primary key (IDKADARGULA)
8.  );
9.
10. alter table KADARGULA add constraint FK_DIUKUR fore
    ign key (IDPATIENT)
11.       references PATIENT (IDPATIENT) on delete rest
    rict on update restrict;
```

Kode Sumber 0.2 Implementasi Tabel KADARGULA

```
1.  create table RECORDDATA
2.  (
3.     IDRECORD              int not null,
4.     IDPATIENT             varchar(6),
5.     AVG_CO                float,
6.     AVG_HUMID             float,
7.     STD_VOC               float,
8.     STD_CO2               float,
9.     STD_KETONE            float,
10.    primary key (IDRECORD)
11. );
12.
13. alter table RECORDDATA add constraint FK_MMEPUNYAI
    foreign key (IDPATIENT)
14.       references PATIENT (IDPATIENT) on delete rest
    rict on update restrict;
```

Kode Sumber 0.3 Implementasi Tabel RECORDDATA

5.3. Implementasi Proses

Pada subbab ini akan dibahas mengenai implementasi dari masing-masing proses pengolahan data. Proses yang dimaksud adalah detail dari proses yang menerima *input* kepada sistem yaitu implementasi perekaman data napas pasien menggunakan e-Nose hingga hasil dari masing-masing proses pengolahan data yaitu implementasi *preprocessing*, ekstraksi fitur, klasifikasi dan evaluasi. Detail dari masing-masing proses akan dijelaskan pada subbab berikut.

5.3.1. Implementasi Input

Input data merupakan proses perekaman data napas pasien menggunakan e-Nose. Napas pasien direkam selama ±150 detik dan menghasilkan file .txt dengan ±50 baris yang merekam setiap 3 detik respon. Untuk membuat *timer* perekaman data digunakan kode python seperti pada **Kode Sumber 5.4**.

```
1.  import threading
2.
3.  class recordTime(threading.Thread):
4.      def __init__(self, t, boxWindow):
5.          super(recordTime,self).__init__()
6.          self.timerTxt = t
7.          self.remaining = 0
8.          self.mgui = boxWindow
9.          self.runTime = 181
10.         self.run(self.runTime)
11.
12.     def run(self, remaining =  None):
13.             if remaining is not None:
14.                 self.remaining = remaining
15.
16.             if self.remaining <= 0:
17.                 self.timerTxt.configure(text='Data
    Recorded')
18.             else:
19.                 self.timerTxt['text'] =  self.remai
    ning
20.                 self.remaining = self.remaining -
     1
21.                 self.mgui.after(1000, self.run)
```

Kode Sumber 0.4 Implementasi Kelas 'recordTime'

Untuk memvisualisasikan data yang direkam oleh e-Nose maka data yang diterima akan langsung di plot secara *real time* oleh program, selain plot visualisasi, program juga menjalankan tugas untuk mencatat data respon sensor menjadi file .txt. Pada

Kode sumber 5.5 dapat dilihat cara memvisualisasikan data dan pencatatan pada file teks.

```
1.  import Tkinter as tk
2.  import matplotlib.pyplot as plt
3.  import classTimer
4.  import serial
5.  from drawnow import *
6.
7.  class plotFunction:
8.      def __init__(self, timerTxt, entryID, mgui):
9.          self.timerText = timerTxt
10.         self.pasienID = entryID
11.         self.dataSensor = serial.Serial('COM3', 960
    0)
12.         self.gui = mgui
13.         self.timer = None
14.
15.         #file
16.         self.fileName = tk.Entry.get(self.pasienID)
     + '.txt'
17.         self.folderPPM = 'recordData/ppm/' + self.f
    ileName
18.         self.folderRaw = 'recordData/raw/' + self.f
    ileName
19.
20.         self.co     = []
21.         self.co2    = []
22.         self.ketone = []
23.         self.temp   = []
24.         self.humid  = []
25.         self.voc    = []
26.
27.     def makeFigure(self):
28.         plt.figure('CO, CO2, Ketone in Breathe')
29.         plt.ylim(1.20, 20.0)
30.         plt.title('CO, CO2, Ketone PPM')
31.         plt.grid(True)
32.         plt.ylabel('CO & CO2 PPM')
33.         plt.plot(self.co, 'go-', label="CO")
34.         plt.plot(self.co2, 'ro-', label="CO2")
35.         plt.legend(loc="upper left")
```

```
36.
37.         plt2 = plt.twinx()
38.         plt.ylim(140, 160)
39.         plt2.plot(self.ketone, 'b^-
    ', label="Ketone")
40.         plt2.set_ylabel('Ketone PPM in breathe')
41.         plt2.ticklabel_format(useOffset=False)
42.         plt2.legend(loc="upper right")
43.
44.         plt.figure('Temp, Humid & VOC')
45.         plt.ylim(20, 40)
46.         plt.title('Temperature and Humidity in Brea
    th')
47.         plt.grid(True)
48.         plt.ylabel('Temperature in Celcius')
49.         plt.plot(self.temp, 'bo-
    ', label="Temperature")
50.         #plt.plot(self.voc, 'r-
    ', label="VOC") terlalu tinggi nilainya
51.         plt.legend(loc="upper left")
52.
53.         plt2 = plt.twinx()
54.         plt.ylim(40, 100)
55.         plt2.plot(self.humid, 'g-
    ', label="Humidity")
56.         plt2.set_ylabel('Humidity % in breathe')
57.         plt2.ticklabel_format(useOffset=False)
58.         plt2.legend(loc="upper right")
59.
60.     def writeFile(self, rawData, ppmData):
61.         with open(self.folderRaw, 'a') as raw_file:

62.             raw_file.write(rawData)
63.
64.         with open(self.folderPPM, 'a') as the_file:

65.             the_file.write(ppmData)
66.
67.     def startRecord(self):
68.         timer = classTimer.recordTime(self.timerTex
    t, self.gui)
69.         timer.start()
70.         count = 0
```

```
71.
72.         while timer.remaining > 0:
73.             while(self.dataSensor == 0):
74.                 print "Sensor calibrating..."
75.                 pass
76.
77.             dataRecordPasien = self.dataSensor.read
    line()
78.             dataArray = dataRecordPasien.split(',')

79.
80.             # raw
81.             raw_monox = float(dataArray[0])
82.             raw_diox = float(dataArray[1])
83.             raw_ketone = float(dataArray[2])
84.             mics = float(dataArray[8])
85.
86.             # ppm
87.             monox = float(dataArray[3])
88.             diox = float(dataArray[4])
89.             ketones = float(dataArray[5])
90.             temperature = float(dataArray[6])
91.             humidity = float(dataArray[7])
92.
93.             rawData = ("%f,%f,%f,%f,%f,%f\n") % (ra
    w_monox, raw_diox, raw_ketone, temperature, humidit
    y, mics)
94.             ppmData = ("%f,%f,%f,%f,%f,%f\n") % (mo
    nox, diox, ketones, temperature, humidity, mics)
95.
96.             self.writeFile(rawData, ppmData)
97.             self.co.append(monox)
98.             self.co2.append(diox)
99.             self.ketone.append(ketones)
100.                    self.temp.append(temperature)
101.                    self.humid.append(humidity)
102.                    self.voc.append(mics)
103.
104.                    drawnow(self.makeFigure)
105.                    plt.pause(.000001)
106.                    count  = count + 1
107.
108.                    if (count > 50):
```

```
109.                            self.co.pop(0)
110.                            self.co2.pop(0)
111.                            self.ketone.pop(0)
112.                            self.temp.pop(0)
113.                            self.humid.pop(0)
114.                            self.voc.pop(0)
115.
116.                print "Finish"
```

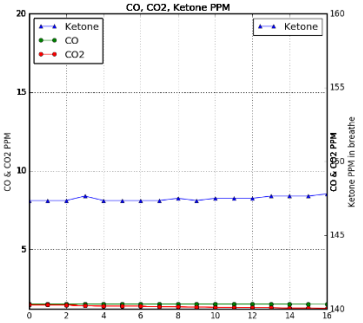
Kode Sumber 0.5 Implementasi Kelas 'plotFunction'

Untuk menjalankan fungsi-fungsi diatas maka juga perlu di implementasikan main yang berfungsi untuk membuat objek-objek diatas dan membuat *textbox* untuk menuliskan ID file seperti pada **Kode Sumber 5.6.**

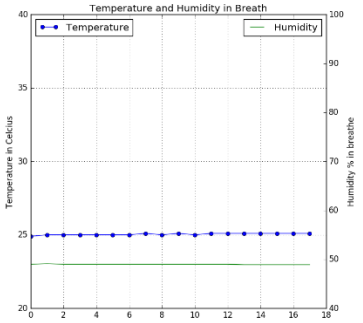
```
1.  import Tkinter as tk
2.  import plotData
3.
4.  def start():
5.      plotFunction = plotData.plotFunction(timerTxt,
    entryID, mgui)
6.      plotFunction.startRecord()
7.
8.  mgui = tk.Tk()
9.  mgui.title('Record Data Patient')
10. timerTxt = tk.Label(mgui, text='Let us Begin!', wid
    th=10, fg='red')
11. timerTxt.grid(row=1, column=2)
12. patientID = tk.Label(mgui, text="ID Patient").grid(
    row=0, column=1)
13. entryID = tk.Entry(mgui, bd=3)
14. entryID.grid(row=0, column=2)
15. btn = tk.Button(mgui, text="Start", width=10, comma
    nd=start).grid(row=2, column=2)
16. mgui.mainloop()
```

Kode Sumber 0.6 Implementasi main input

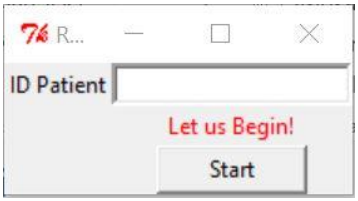
Berikut adalah hasil dari jalan program yaitu, kotak untuk memasukkan ID File seperti pada **Gambar 5.3**Gambar 5.3 Kotak input ID, dan visualisasi *real-time* plotting data seperti pada Error! Reference source not found. dan **Gambar 5.2**.



Gambar 5.1 *Real-time* plot ketone,co,co2



Gambar 5.2 *Real-time* plot temperature dan kelembapan



Gambar 5.3 Kotak input ID

5.3.2. Implementasi Preprocessing

Preprocessing dilakukan untuk *denoising* sinyal menggunakan transformasi wavelet diskrit yang terintegrasi dengan *website* ketika melakukan klasifikasi. Berikut dapat dilihat implementasi *preprocessing* menggunakan *library* PyWavelet. Sebagai nilai keluaran adalah koefisian aproksimasi dari DWT karena *noise* diharapkan berada pada frekuensi yang lebih tinggi. Sehingga koefisien aproksimasi berada pada *low pass* yang diharapkan keluaran sinyal sensor bersih. Berikut adalah implementasi kelas *preprocessing* seperti pada **kode sumber 5.7**.

```
1.  from pywt import dwt, wavedec
2.
3.  class preProcessing:
4.      def __init__(self, co, co2, ketone, h, t, voc):

5.          self.karbonMonok = co
6.          self.karbonDiok = co2
7.          self.ketone = ketone
8.          self.humid = h
9.          self.temp = t
10.         self.voc = voc
11.
12.     def dwtDB6(self):
13.         cA_CO, cD_CO = wavedec(self.karbonMonok,'db
    6', level=1)
14.         cA_CO2, cD_CO2 = wavedec(self.karbonDiok, '
    db6', level=1)
15.         cA_Ketone, cD_Ketone = wavedec(self.ketone,
     'db6', level=1)
16.         cA_Humid, cD_Humid = wavedec(self.humid, 'd
    b6', level=1)
17.         cA_VOC, cD_VOC = wavedec(self.voc, 'db6', l
    evel=1)
18.
```

```
19.         return cA_CO, cA_CO2, cA_Ketone, cA_Humid,
    cA_VOC
```

Kode Sumber 0.7 Implementasi kelas *preprocessing*

5.3.3. Implementasi Ekstraksi Fitur

Ekstraksi fitur dilakukan untuk mendapatkan fitur-fitur statistic dari keluaran sinyal *denoising* setelah analisis seleksi fitur, maka fitur yang paling dominan adalah nilai rata-rata CO, nilai standar deviasi CO, nilai standard deviasi CO2, nilai standar deviasi ketone, nilai rata-rata kelembapan dan nilai standard deviasi VOC. Implementasi kode sumber ekstraksi fitur dapat dilihat pada **Kode Sumber 5.8.**

```
1.  import numpy as np
2.  import preProcessing as dwt
3.
4.
5.  class getFeatures:
6.      def __init__(self, namaFile, namaClass):
7.          self.data = {}
8.          self.dataClass = namaClass #ambil class dar
    i nama file
9.
10.         with open(namaFile) as f:
11.             baris = f.readlines()
12.             f.close()
13.         formatData = baris[0].split(',')
14.
15.         self.kolom = len(formatData)
16.
17.         self.data = [[] for i in range(self.kolom)]

18.
19.         for line in baris[0:]:
20.             fitur = line.split(',')
21.             for kolom in range(self.kolom):
22.                 self.data[kolom].append(float(fitur
    [kolom]))
```

```
23.         #print len(self.data[0])
24.
25.     @staticmethod
26.     def avg(gasArray):
27.         return np.average(gasArray)
28.     @staticmethod
29.     def std(gasArray):
30.         return np.std(gasArray)
31.
32.     def featuresDWT(self):
33.         dwtFunc = dwt.preProcessing(self.data[0], s
    elf.data[1], self.data[2], self.data[3], self.data[
    4], self.data[5])
34.         cA_CO, cA_CO2, cA_Ketone, cA_Humid, cA_VOC
    = dwtFunc.dwtDB6()
35.
36.         return self.avg(cA_CO), self.std(cA_CO), se
    lf.std(cA_CO2), self.std(cA_Ketone), self.avg(cA_Hu
    mid), self.std(cA_VOC)
```

Kode Sumber 0.8 Implementasi kelas *preprocessing*

5.3.4. Implementasi Klasifikasi

Klasifikasi digunakan untuk memprediksi kelas untuk data masukan yang baru. Pada proses ini akan digunakan *classifier* k-nn yang dibuat menggunakan kode python seperi pada **Kode Sumber 5.9.**

```
1.  import MySQLdb
2.  import math
3.
4.  class Classifier:
5.      def __init__(self, k, newData):
6.          db = MySQLdb.connect("localhost", "root", "
    ", "dbpatient")
7.          cursor = db.cursor()
8.
9.          sql = "SELECT AVG_HUMID, STD_CO, STD_CO2, S
    TD_KETONE, KELAS, IDPATIENT FROM RECORDDATA WHERE I
    DPATIENT LIKE 'R%'"
```

```
10.
11.         try:
12.             cursor.execute(sql)
13.             result = cursor.fetchall()
14.
15.             db.close()
16.
17.             self.allDataPatient = []
18.             self.k = k
19.             normNewData = []
20.
21.             #print newData
22.
23.             avg_humid = []
24.             std_co = []
25.             std_co2 = []
26.             std_ketone = []
27.
28.             avg_humid.append(newData[0])
29.             std_co.append(newData[1])
30.             std_co2.append(newData[2])
31.             std_ketone.append(newData[3])
32.
33.             for row in result:
34.                 dataPatient = []
35.                 dataPatient.append(row[0])
36.                 avg_humid.append(row[0])
37.                 dataPatient.append(row[1])
38.                 std_co.append(row[1])
39.                 dataPatient.append(row[2])
40.                 std_co2.append(row[2])
41.                 dataPatient.append(row[3])
42.                 std_ketone.append(row[3])
43.                 dataPatient.append(int(row[4]))
44.                 dataPatient.append(row[5])
45.                 self.allDataPatient.append(dataPati
    ent)
46.
47.             normHumid = self.zScore(avg_humid)
48.             normCO = self.zScore(std_co)
49.             normCO2 = self.zScore(std_co2)
50.             normKetone = self.zScore(std_ketone)
51.
```

```
52.             self.newData = [normHumid.pop(0), normC
    O.pop(0), normCO2.pop(0), normKetone.pop(0), newDat
    a[4], newData[5]]
53.             #print self.newData
54.             for record in self.allDataPatient:
55.                 record[0] = normHumid.pop(0)
56.                 record[1] = normCO.pop(0)
57.                 record[2] = normCO2.pop(0)
58.                 record[3] = normKetone.pop(0)
59.
60.
61.
62.             self.predict()
63.         except:
64.             print "Error unable to fetch data"
65.
66.     def cariMean(self, vData):
67.         temp = list(vData)
68.         temp.sort()
69.         lenKolom = len(temp)
70.
71.         if (lenKolom % 2) == 1:
72.             return temp[lenKolom // 2]
73.         else:
74.             return (temp[lenKolom // 2] + temp[(len
    Kolom // 2) - 1]) / 2
75.
76.     def zScore(self, kolom):
77.         mean = self.cariMean(kolom)
78.         asd = math.sqrt((sum([abs(x -
     mean) for x in kolom])) ** 2 / len(kolom))
79.         hasil = [(x - mean) / asd for x in kolom]
80.
81.         return hasil
82.
83.     def canberraDistance(self, v1, v2):
84.         dist = 0
85.         lenVektor = len(v1) - 3
86.
87.         for atr in range(0, lenVektor):
88.             calcAtr = abs(v1[atr] -
     v2[atr]) / (abs(v1[atr]) + abs(v2[atr]))
89.             dist = dist + calcAtr
```

```
90.
91.         return dist
92.
93.     def calcVote(self, vHasil):
94.         #print vHasil
95.         countHealth = 0
96.         countDiabet = 0
97.         sumDHealth = 0
98.         sumDDiabet = 0
99.
100.                for member in vHasil[0:self.k]:
101.                    if member[1] == 1:
102.                        countHealth = countHealth +
    1
103.                        sumDHealth = sumDHealth + me
    mber[0]
104.                    elif member[1] == 3:
105.                        countDiabet = countDiabet +
    1
106.                        sumDDiabet = sumDDiabet + me
    mber[0]
107.
108.                #print countHealth
109.                #print countDiabet
110.
111.                if countHealth > countDiabet:
112.                    return 1
113.                elif countDiabet > countHealth:
114.                    return 3
115.                elif countHealth == countDiabet:
116.                    if sumDDiabet < sumDHealth:
117.                        return 3
118.                    else:
119.                        return 1
120.                else:
121.                    return 0
122.
123.            def knn(self):
124.                hasil = []
125.                for record in self.allDataPatient:
126.                    if (record[5] != self.newData[5]
    ):
```

```
127.                         dist = self.canberraDistance
    (self.newData, record)
128.                         hasil.append((dist, record[4
    ]))
129.
130.                 return self.calcVote(sorted(hasil, k
    ey=lambda x: x[0]))
131.
132.             def predict(self):
133.
134.                 try:
135.                     tp = 0
136.                     fp = 0
137.                     tn = 0
138.                     fn = 0
139.
140.                     predictClass = self.knn()
141.                     print predictClass
142.
143.                     if self.newData[4] == 1:
144.                         if self.newData[4] == predic
    tClass:
145.                             tp = tp + 1
146.                         else:
147.                             fn = fn + 1
148.                     elif self.newData[4] == 3:
149.                         if self.newData[4] == predic
    tClass:
150.                             tn = tn + 1
151.                         else:
152.                             fp = fp + 1
153.                 except:
154.                     print "Error unable to fetch dat
    a"
```

Kode Sumber 0.9 Implementasi kelas *Classifier*

5.3.5. Implementasi Evaluasi

Evaluasi kinerja *classifier* diukur menggunakan akurasi, presisi, recall dan kappa statistic yang diturunkan dari *confusion*

matrix. Dapat dilihat pada **Kode Sumber 5.10** implementasi perhitungan metric-metric tersebut.

```
1.  def accuracy(self, tp, fp, tn, fn):
2.          total = tp + tn + fp + fn
3.          predictHealth = fp + tp
4.          predictDiabetes = fn + tn
5.          actualDiabetes = fp + tn
6.          actualHealth = tp + fn
7.
8.          accuracy = float((tp + tn)) / float(total)

9.          precissionHealth = float(tp) / float((predi
    ctHealth))
10.         precissionDiabet = float(tn) / float((predi
    ctDiabetes))
11.         recallHealth = float(tp) / float((actualHea
    lth))
12.         recallDiabet = float(tn) / float((actualDia
    betes))
13.
14.         coeffDiabetes = float(precissionDiabet) / t
    otal
15.         coeffHealth = float(predictHealth) / total

16.         randomAccuracy = (coeffDiabetes * actualDia
    betes + coeffHealth * actualHealth) / total
17.
18.         kappa = ((accuracy -
     randomAccuracy)) / (1 - randomAccuracy)
19.
20.         print "Akurasi: ", accuracy * 100.0
21.         print "Precission of Health: ", precissionH
    ealth * 100.0
22.         print "Precission of Diabetes: ", precissio
    nDiabet * 100.0
23.         print "Recall of Health: ", recallHealth *
    100.0
24.         print "Recall of Diabet: ", recallDiabet *
    100.0
25.         print "Kappa: %.2f" % kappa
```

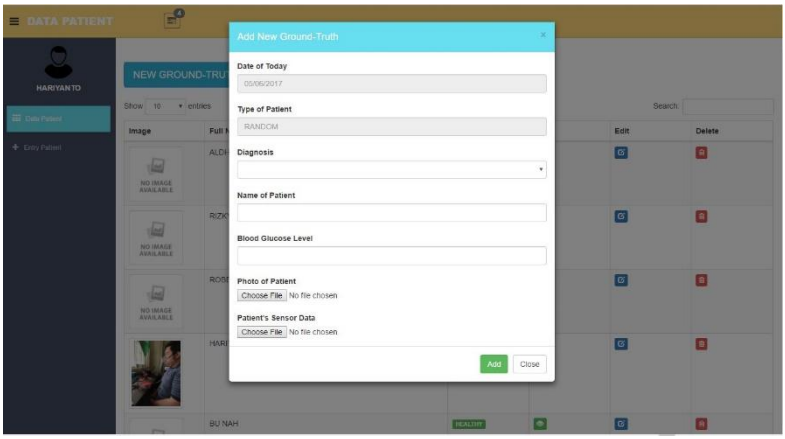
Kode Sumber 0.10 Implementasi fungsi *accuracy* pad akelas *classifier*

5.4. Implementasi Tampilan Antarmuka

Pada sub-bab ini akan dijelaskan mengenai implementasi antarmuka sistem web yang menjadi bagian terluar dan bagian yang berinteraksi langsung dengan pengguna.

5.4.1. Implementasi Halaman Menambah Data Pasien atau *Ground Truth* (UC-1)

Halaman ini terletak pada halaman utama dan muncul sebagai *pop-up* yang muncul ketika administrator menekan tombol "NEW GROUND-TRUTH". View ini adalah *home.php* yang dipanggil oleh *cDataPasien.php*. Implementsi tampilan dapat dilihat pada **Gambar 5.4.**



Gambar 5.4 Implementasi Halaman Menambah Data Pasien atau *Ground Truth*

```
1.  function addPatient2(){
2.              $namaPatient = strtoupper($this->input-
    >post('patientName'));
```

```
3.              $diagnosa = $this->convertDiagnose($this->input->post('diagnosa'));
4.              $tipe = $this->convertType('RANDOM');
5.              $bgl = $this->input->post('bloodGlucose');
6.              #echo "<script>alert($diagnosa)</script>";
7.              $idPatient = $this->getID($tipe, $diagnosa);
8.
9.              #echo "<script>alert('masuk')</script>";
10.
11.             $patientData = array(
12.                 'IDPATIENT' => $idPatient,
13.                 'NAMAPATIENT' => $namaPatient,
14.                 'DIAGNOSAPATIENT' => $diagnosa,
15.                 'JENISPATIENT' => $tipe,
16.                 'TANGGALRECORD' => date('Y-m-d', now()),
17.                 'KADARGULA' => $bgl,
18.                 'FOTOPATIENT' => $this->uploadFoto(),
19.                 'SENSORDATA' => $this->uploadSensor()
20.             );
21.
22.             #echo "<script>alert('masuk')</script>";
23.
24.             if($this->patientModel->insertPatient($patientData)){
25.                 $fileSensorPasien = $patientData['SENSORDATA'];
26.                 $path = "C:\wamp\www\dataPatient\CodeIgniter-3.1.4\assets\scripts\getStatisticFeatures\main.py";
27.                 $result = shell_exec("python $path $fileSensorPasien $diagnosa");
28.
29.                 $dataSensor = explode(',', $result);
```

```
30.                 $inputDataSensor = array(
31.                     'AVG_CO' => $dataSensor[0],
32.                     'STD_CO' => $dataSensor[1],
33.                     'STD_CO2' => $dataSensor[2],
34.                     'STD_KETONE' => $dataSensor[3],
35.                     'AVG_HUMID' => $dataSensor[4],
36.                     'STD_VOC' => $dataSensor[5],
37.                     'IDPATIENT' => $idPatient,
38.                     'KELAS' => $diagnosa
39.                 );
40.
41.                 if($this->patientModel->insertDataSensor($inputDataSensor)){
42.                     redirect(base_url());
43.                 }
44.                 else{
45.                     echo "<script>alert('Error Insert Data!');</script>";
46.                 }
47.                 //panggil fungsi python buat ambil statistik data
48.                 //insert statistik data ke db
49.
50.             }
51.     }
```

Kode Sumber 0.11 Fungsi untuk mengambil data inputan dari view pada *controller*

```
1.  public function insertPatient($data){
2.             if($this->db->insert("patient", $data)){
3.                 return true;
4.             }
5.         }
```

Kode Sumber 0.12 Model untuk *insert* data pasien ke basis data

```
1.  public function getID($tipe, $diagnosa){
```

```
2.       switch ($tipe){
3.           case 1:
4.               $type = "R";
5.               break;
6.           case 2:
7.               $type = "F";
8.               break;
9.           default:
10.              $type = "U";
11.      }
12.
13.      switch ($diagnosa){
14.          case 1:
15.              $diagnose = "H";
16.              break;
17.          case 2:
18.              $diagnose = "A";
19.              break;
20.          case 3:
21.              $diagnose = "C";
22.              break;
23.          default:
24.              $diagnose = "H";
25.      }
26.
27.      $idNomor = $this-
    >cekNumberRow($tipe, $diagnosa);
28.
29.      $idPatient = $type.$diagnose.$idNomor;
30.
31.      return $idPatient;
32.  }
```

Kode Sumber 0.13 Fungsi pada *Controller* untuk mandapat ID Pasien secara Otomatis

```
1.  protected function cekNumberRow($tipe, $diagnosa){

2.        $numberRow = $this->patientModel-
    >countRow($tipe, $diagnosa);
3.
4.        #echo $numberRow;
```

```
5.        return str_pad($numberRow+1, 3, "0", STR_PAD_
    LEFT);
6.    }
```

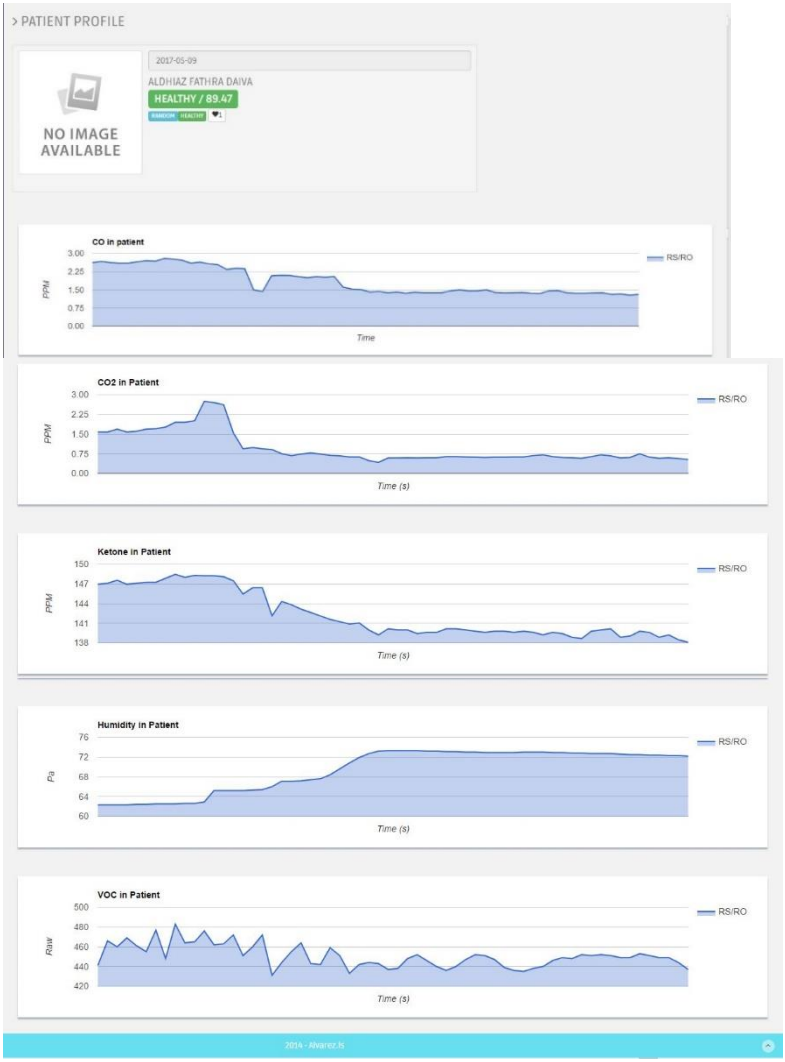
Kode Sumber 0.14 Fungsi pada *Controller* untuk menghitung jumlah *row*

```
1.  public function countRow($tipe, $diagnosa){
2.              $this->db->select('NAMAPATIENT');
3.              $this->db->from('patient');
4.              $this->db-
    >where("DIAGNOSAPATIENT = $diagnosa and JENISPATIEN
    T = $tipe");
5.              $query = $this->db->get();
6.              return $query->num_rows();
7.          }
```

Kode Sumber 0.15 Fungsi untuk menghitung jumlah baris pada tabel di model

5.4.2. Implementasi Halaman Profil Pasien serta Hasil Klasifikasi (UC-2)

Halaman ini adalah halaman untuk melihat profil pasien dan hasil klasifikasi. Profil pasien berupa foto pasien, nama pasien, jenis pasien, kadar gula darah tercatat, diagnosis pasien dan hasil klasifikasi oleh *classifier*. Terdapat juga grafik kandungan gas dalam napas pasien dalam ppm. Implementasi tampilan dapat dilihat pada **Gambar 5.5 dan Kode Sumber 0.16, Kode Sumber 0.17.**



Gambar 5.5 Implementasi Halaman Profil Pasien dan Hasil Klasifikasi

```
1.  public function getProfile(){
2.      $idpatient = $this->uri->segment('3');
```

```
3.      $pasien['profil'] = $this->patientModel-
    >getPatient($idpatient);
4.
5.      $head['title'] = "Profile | DATA PATIENT";
6.      $head['id'] = $idpatient;
7.      $dataSensorPasien = $this->patientModel-
    >getDataSensor($idpatient);
8.      $sensorData = $dataSensorPasien['AVG_HUMID'].",
    ".$dataSensorPasien['STD_CO'].",".$dataSensorPasien
    ['STD_CO2'].",".$dataSensorPasien['STD_KETONE'].","
    .$dataSensorPasien['KELAS'].",".$dataSensorPasien['
    IDPATIENT'];
9.      $path = "C:\wamp\www\dataPatient\CodeIgniter-
    3.1.4\assets\scripts\classification\main3.py";
10.     $path2 = "C:\wamp\www\dataPatient\CodeIgniter-
    3.1.4\assets\scripts\classification\main2.py";
11.     $pasien['predictClass'] = shell_exec("python $p
    ath $sensorData");
12.     #echo "<script>console.log( 'Debug Objects: " .
     $pasien['predictClass'] . "' );</script>";
13.     $pasien['evaluation'] = shell_exec("python $pat
    h2");
14.
15.     $this->load-
    >view('heading/headProfile', $head);
16.     $this->load->view('profilePatient', $pasien);
17.     $this->load->view('footing/footEntry');
18. }
```

Kode Sumber 0.16 Fungsi pada *controller* untuk mendapatkan data pasien yang ingin dilihat

```
1.  public function getPatient($idPatient){
2.          $this->db->select('*');
3.          $this->db->from('patient');
4.          $this->db-
    >where('IDPATIENT', $idPatient);
5.          $query = $this->db->get();
6.
7.          return $query->row();
8.      }
```

Kode Sumber 0.17 Fungsi pada model untuk mendapatkan profil pasien

```
1.  <?php
2.  defined('BASEPATH') OR exit('No direct script acces
    s allowed');
3.
4.  class cDataPasien extends CI_Controller{
5.      function __construct(){
6.          parent::__construct();
7.          $this->load->model('');
8.          $this->load->helper('url_helper');
9.          $this->load->library('session');
10.         $this->load->model('patientModel');
11.         $this->load->helper('date');
12.         $this->load->database();
13.     }
14.
15.     function getNamaFile($idPatient){
16.         $namaFile = $this->patientModel-
    >getFileName($idPatient);
17.
18.         return $namaFile;
19.     }
20.
21.     function getCOData(){
22.         $idpatient = $this->uri->segment('3');
23.         $namaFile = $this-
    >getNamaFile($idpatient);
24.         #echo "<script>console.log( 'Debug Objects:
     " . $namaFile . "' );</script>";
25.         $path = "C:\wamp\www\dataPatient\CodeIgnite
    r-3.1.4\assets\scripts\getDataCO.py";
26.         $result = shell_exec("python $path $namaFil
    e");
27.         #print_r($result);
28.         #echo "<script>console.log( 'Debug Objects:
     " . $result . "' );</script>";
29.         echo $result;
30.         #return $result;
31.     }
32.
```

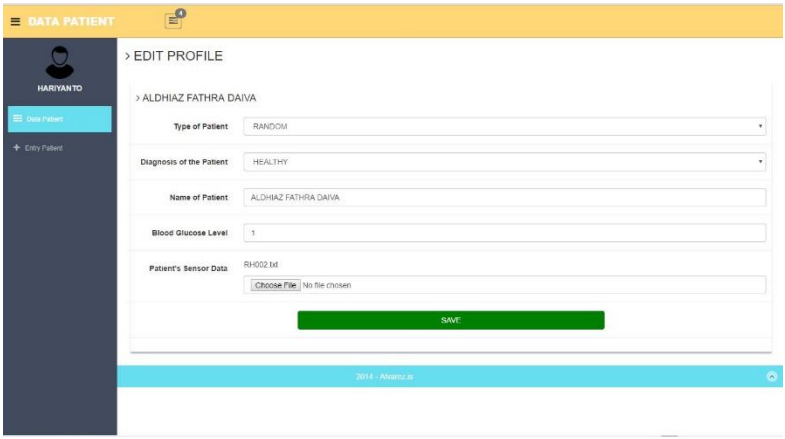
```
33.     function getCO2Data(){
34.         $idpatient = $this->uri->segment('3');
35.         $namaFile = $this-
    >getNamaFile($idpatient);
36.         $path = "C:\wamp\www\dataPatient\CodeIgnite
    r-3.1.4\assets\scripts\getDataCO2.py";
37.         $result = shell_exec("python $path $namaFil
    e");
38.
39.         echo $result;
40.     }
41.
42.     function getKetoneData(){
43.         $idpatient = $this->uri->segment('3');
44.         $namaFile = $this-
    >getNamaFile($idpatient);
45.         $path = "C:\wamp\www\dataPatient\CodeIgnite
    r-3.1.4\assets\scripts\getDataKetone.py";
46.         $result = shell_exec("python $path $namaFil
    e");
47.
48.         echo $result;
49.     }
50.
51.     function getHumidData(){
52.         $idpatient = $this->uri->segment('3');
53.         $namaFile = $this-
    >getNamaFile($idpatient);
54.         $path = "C:\wamp\www\dataPatient\CodeIgnite
    r-3.1.4\assets\scripts\getDataHumid.py";
55.         $result = shell_exec("python $path $namaFil
    e");
56.
57.         echo $result;
58.     }
59.
60.     function getVOCData(){
61.         $idpatient = $this->uri->segment('3');
62.         $namaFile = $this-
    >getNamaFile($idpatient);
63.         $path = "C:\wamp\www\dataPatient\CodeIgnite
    r-3.1.4\assets\scripts\getDataVOC.py";
```

```
64.         $result = shell_exec("python $path $namaFil
    e");
65.
66.         echo $result;
67.     }
68. }
```

Kode Sumber 0.18 Kelas untuk Mendapatkan Data Sensor menjadi Json

5.4.3. Implementasi Halaman Mengubah Data *Ground Truth* atau Pasien Terdaftar (UC-3)

Halaman ini adalah halaman untuk mengubah detail informasi pasien yang sudah tercatat pada basis data. Detail informasi termasuk adalah nama pasien, diagnosis pasien, kadar gula darah tercatat, dan file respon sensor yang tercatat pada basis data. Berikut dapat dilihat implementasi tampilan pada **Gambar 5.6** dan kode sumbernya pada **Kode Sumber 5.19** dan **Kode Sumber 5.20**.



Gambar 5.6 Implementasi halaman mengubah data *ground truth* atau pasien terdaftar

```
1.  function editPatient(){
2.      $idPatient = $this->uri->segment('3');
3.      $updateNamaPatient = strtoupper($this->input-
    >post('patientName'));
4.      $updateDiagnosa = $this->convertDiagnose($this-
    >input->post('diagnosa'));
5.      $updateTipe = $this->convertType($this->input-
    >post('patientType'));
6.      $updateBgl = $this->input-
    >post('bloodGlucose');
7.
8.      if(isset($_FILES['patientSensor'])){
9.          echo "<script>console.log( 'Debug Objects:
    masuk' );</script>";
10.     }else{
11.         echo "<script>console.log( 'Debug Objects:
    gak' );</script>";
12.     }
13.
14.     $updateData = array(
15.         'NAMAPATIENT' => $updateNamaPatient,
16.         'DIAGNOSAPATIENT' => $updateDiagnosa,
17.         'JENISPATIENT' => $updateTipe,
18.         'KADARGULA' => $updateBgl,
```

```
19.         'SENSORDATA' => $this->uploadSensor()
20.     );
21.
22.     if($this->patientModel-
    >updatePatient($idPatient, $updateData)){
23.             $fileSensorPasien = $updateData['SENSOR
    DATA'];
24.             $path = "C:\wamp\www\dataPatient\CodeIg
    niter-
    3.1.4\assets\scripts\getStatisticFeatures\main.py";

25.             $result = shell_exec("python $path $fil
    eSensorPasien $updateDiagnosa");
26.
27.             $dataSensor = explode(',', $result);
28.             echo "<script>console.log( 'Debug Objec
    ts: " . $dataSensor[0] . "' );</script>";
29.             $inputDataSensor = array(
30.                 'AVG_CO' => $dataSensor[0],
31.                 'STD_CO' => $dataSensor[1],
32.                 'STD_CO2' => $dataSensor[2],
33.                 'STD_KETONE' => $dataSensor[3],
34.                 'AVG_HUMID' => $dataSensor[4],
35.                 'STD_VOC' => $dataSensor[5],
36.                 'IDPATIENT' => $idPatient,
37.                 'KELAS' => $updateDiagnosa
38.             );
39.
40.             if($this->patientModel-
    >insertDataSensor($inputDataSensor)){
41.                 redirect(base_url("index.php/cPasie
    n/getProfile/$idPatient"));
42.             }
43.             else{
44.                 echo "<script>alert('Error Insert D
    ata!');</script>";
45.             }
46.     }
47. }
```

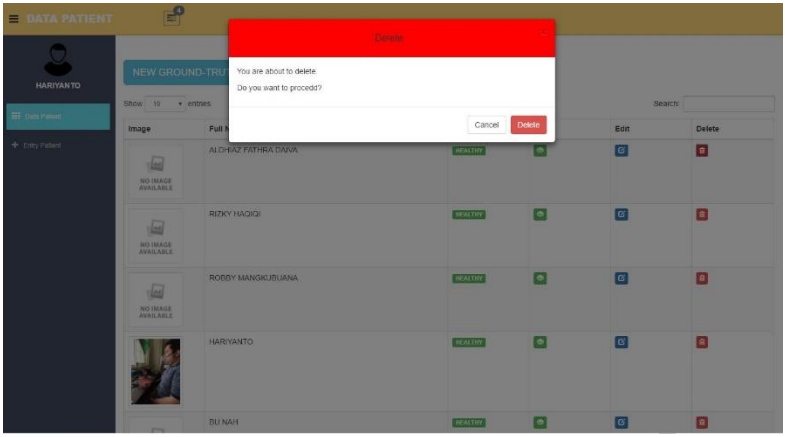
Kode Sumber 0.19 Fungsi untuk mendapatkan data pasien dan mengubah pada basis data di *controller*

```
1.  function updatePatient($idPatient, $updateData){
2.          $this->db->where("IDPATIENT", $idPatient);
3.          $this->db->update("patient", $updateData);
4.  
5.          return true;
6.      }
```

Kode Sumber 0.20 Fungsi pada model untuk *update* data

5.4.4. Implementasi Halaman Menghapus Data *Ground Truth* atau Pasien Terdaftar (UC-4)

Halaman ini terletak pada halaman utama dan digunakan untuk menghapus data pasien atau *ground truth* terdaftar. Tampilan yang diberikan berupa *pop-up* konfirmasi sebelum menghapus data pada basis data. *View* yang digunakan adalah *home.php* dan *controller* yang digunakan adalah cDataPasien.php dan model yang digunakan yaitu *patientModel.php*. Fungsi untuk menghapus pasien ada pada *controller* pada **Kode Sumber 5.21** dan fungsi untuk menghapus pada model ada pada **Kode Sumber 5.22**.



Gambar 5.7 Implementasi Tampilan Delete Pasien

```
1.  public function deletePatient(){
2.        $idPatient = $this->input-
    >post('idpatient',TRUE);
3.        $this->patientModel-
    >deletePatient($idPatient);
4.    }
```

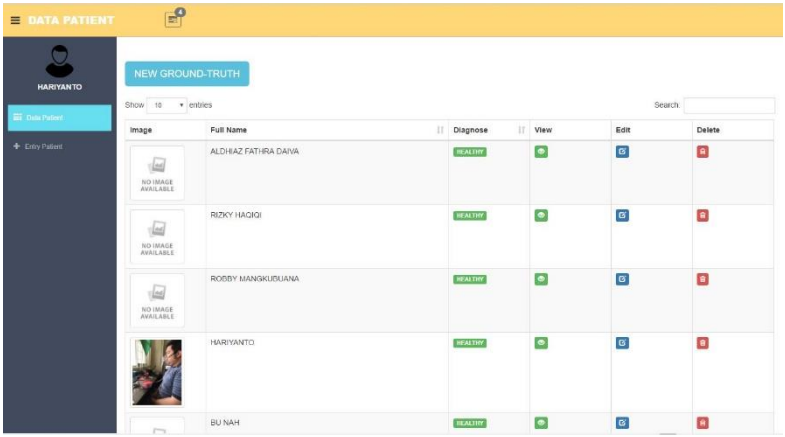
Kode Sumber 0.21 Fungsi untuk delete pasien pada controller

```
1.  public function deletePatient($idPatient){
2.          $this->db->where('IDPATIENT',$idPatient);
3.          if($this->db->delete('patient')){
4.              return true;
5.          }
6.      }
```

Kode Sumber 0.22 Fungsi pada model untuk menghapus data pasien dalam basis data

5.4.5. Implementasi Halaman Melihat Seluruh Data Pasien dan *Ground Truth* (UC-5)

Halaman ini adalah halaman utama dan menampilkan daftar pasien dan *ground truth* yang terdaftar pada sistem. Pada halaman ini terdapat tabel utama yang menampilkan seluruh data pasien terdaftar. *Controller* yang digunakan adalah *cHalamanUtama.php* dan *view* tetap pada *home.php* dan model digunakan untuk mengambil seluruh data tabel patient.



Gambar 5.8 Implementasi Halaman Utama

Pada **Kode Sumber 5.23** adalah fungsi *controller* untuk membuat dataTables pada halaman utama yang berfungsi untuk mengambil seluruh data pasien terdaftar dari model.

```
1.  public function getPatient2(){
2.        #console.log('masuk');
3.        $allPatient = $this->patientModel-
    >makeDataTables();
4.        #console.log('masuk');
5.        #console.log($allPatient);
6.        $data = array();
7.
8.        foreach($allPatient as $row)
9.        {
10.
11.           if($row->FOTOPATIENT == ''){
12.               $potoPatient = 'noPhoto.jpg';
13.           }
14.           else if(explode('.',$row-
    >FOTOPATIENT)[1] == ''){
15.               $potoPatient = 'noPhoto.jpg';
16.           }
17.           else{
18.               $potoPatient = $row->FOTOPATIENT;
```

```
19.           }
20.
21.           $sub_array = array();
22.           $sub_array[] = '<img src="'.base_url().'u
    ploads/images/'.$potoPatient.'" class="img-
    thumbnail" width="100" height="50" />';
23.           $sub_array[] = $row->NAMAPATIENT;
24.           if ($row->DIAGNOSAPATIENT == 1){
25.               $sub_array[] = "<span class='label la
    bel-success label-mini'>HEALTHY</span>";
26.           }
27.           else if ($row->DIAGNOSAPATIENT == 2){
28.               $sub_array[] = "<span class='label la
    bel-warning label-mini'>AMID</span>";
29.           }
30.           else if ($row->DIAGNOSAPATIENT == 3){
31.               $sub_array[] = "<span class='label la
    bel-danger label-mini'>CHRONIC</span>";
32.           }
33.
34.           $sub_array[] = "<a href='".base_url()."in
    dex.php/cPasien/getProfile/".$row-
    >IDPATIENT."'><button class='btn btn-success btn-
    xs' data-target='#myProfile' data-
    toggle='modal'><i class='glyphicon glyphicon-eye-
    open'></i></button></a>";
35.           $sub_array[] = "<a href='".base_url()."in
    dex.php/cPasien/showEditPage/".$row-
    >IDPATIENT."'><button class='btn btn-primary btn-
    xs'><i class='glyphicon glyphicon-
    edit'></i></button>";
36.           $sub_array[] = "<button class='btn btn-
    danger btn-xs delpatient' data-id='".$row-
    >IDPATIENT."'><i class='fa fa-trash-
    o'></i></button>";
37.           $data[] = $sub_array;
38.       }
39.
40.       $output = array(
41.           "draw" => intval($_POST["draw"]),
42.           "recordsTotal" => $this->patientModel-
    >get_all_data(),
```

```
43.           "recordsFiltered" => $this->patientModel-
    >get_filtered_data(),
44.           "data" => $data
45.       );
46.       echo json_encode($output);
47.   }
```

Kode Sumber 0.23 Fungsi pada *controller* untuk mengambil seluruh data pasien

Pada **Kode Sumber 5.24** adalah fungsi model untuk mendapatkan seluruh data pasien terdaftar diurutkan berdasar diagnose pasien (sehat/diabetes).

```
1.  public function makeDataTables(){
2.          $this->db-
    >order_by('DIAGNOSAPATIENT', 'asc');
3.          $query = $this->db->get('patient');
4.
5.          return $query->result();
6.       }
```

Kode Sumber 0.24 Fungsi pada *controller* untuk mengambil seluruh data pasien

Berikut pada **Kode Sumber 5.25** adalah kode untuk menampilkan tabel utama dan membaca data dari basis data.

```
1.  <section id="main-content">
2.          <section class="wrapper">
3.              <div class="row mt">
4.                  <div class="col-md-12">
5.                      <div class="content-panel">
6.                          <table class="table table-
    striped table-advance table-hover" id="myTable">
7.                           <h4><i class="fa fa-
    angle-right"></i> Patients Recorded Data</h4>
8.                           <hr>
```

```
                                <input type="text" id="myInput" onkeyup="myFunction()" placeholder="Search for names..">
                                <thead>
                                <tr>
                                    <th><i class="fa fa-bullhorn"></i> ID</th>
                                    <th class="hidden-phone"><i class="fa fa-question-circle"></i> Name</th>
                                    <th><i class="fa fa-edit"></i> Diagnose</th>
                                    <th></th>
                                </tr>
                                </thead>
                                <tbody>
                               <?php
                                foreach($records as $patient){
                                echo "<tr>";
                                echo "<td>";
                                echo "<a href='basic_table.html'>".$patient->IDPATIENT."</a>";
                                echo "</td>";
                                echo "<td>".$patient->NAMAPATIENT."</td>";
                                if ($patient->DIAGNOSAPATIENT == 1){
                                  echo "<td><span class='label label-success label-mini'>HEALTHY</span></td>";
                                }
                                else if ($patient->DIAGNOSAPATIENT == 2){
                                  echo "<td><span class='label label-warning label-mini'>AMID</span></td>";
                                }
                                else if ($patient->DIAGNOSAPATIENT == 3){
                                  echo "<td><span class='label label-danger label-mini'>CHRONIC</span></td>";
```

```
34.                                 }
35.                                 echo "<td>";
36.                                 echo "<a href='".base_
    url()."index.php/cPasien/getProfile/".$patient-
    >IDPATIENT."'><button class='btn btn-success btn-
    xs' data-target='#myProfile' data-
    toggle='modal'><i class='glyphicon glyphicon-eye-
    open'></i></button></a>";
37.                                 echo "<button class='b
    tn btn-primary btn-
    xs'><i class='glyphicon glyphicon-
    edit'></i></button>";
38.                                 echo "<button class='b
    tn btn-danger btn-xs delpatient' data-
    id='".$patient->IDPATIENT."'><i class='fa fa-trash-
    o'></i></button>";
39.                                 echo "</td>";
40.                                 echo "</tr>";
41.                                 }
42.                               ?>
43.                                 </tbody>
44.                             </table>
45.                         </div><!-- /content-panel --
    >
46.                     </div><!-- /col-md-12 -->
47.                 </div>
48.             </section>
49.         </section>
```

Kode Sumber 0.25 Implementasi halaman untuk menampilkan data tabel

[Halaman ini sengaja dikosongkan]

BAB VI
PENGUJIAN DAN EVALUASI

Bab ini membahas pengujian dan evaluasi pada aplikasi yang dikembangkan. Pengujian yang dilakukan adalah pengujian terhadap kebutuhan fungsionalitas sistem yang telah dijabarkan pada Bab IV dan terhadap tujuan dibuatnya aplikasi ini, yakni melakukan implementasi sesuai dengan modul-modul yang telah dijabarkan sebelumnya.

6.1. Lingkungan Pengujian

Lingkungan pengujian sistem pada pengerjaan Tugas Akhir ini dilakukan pada lingkungan server sebagai berikut:

Jenis	: Laptop
Tipe	: HP x360 13
Prosesor	: Intel ® Core™ i7-6500U CPU @ 2.50GHz (4 CPUs), ~2.6GHz
Memori	: 8192MB RAM

6.2. Skenario Uji Coba

Pada bagian ini akan dijelaskan mengenai scenario uji coba yang akan dilakukan. Ada 2 pengujian yaitu, uji coba kinerja *classifier* dan uji coba fungsionalitas.

6.2.1. Skenario Uji Coba *Classifier*

Uji coba kinerja *classifier* dilakukan dengan mengukur nilai akurasi dan kappa untuk setiap percobaan *classifier*. Parameter yang dirubah adalah seleksi fitur yang dilakukan, dan nilai k pada k-nn bernilai tetap yaitu 8. Uji coba yang dilakukan untuk mengukur kinerja *classifier* menggunakan metode *leave-one-out*.

6.2.1.1. Akurasi dengan k-NN tanpa Normalisasi

Berikut adalah nilai akurasi k-nn tanpa dilakukan normalisasi data, yaitu berubah sangat signifikan terhadap perbedaan atribut sehingga kinerja dapat dikatakan tidak *robust*.

Tabel 0.1 Akurasi *classifier* k-nn tanpa normalisasi

Classifier: k-NN		
K = 8		
Distance method: Canberra		
Normalize: NO		
Top N	**Accuracy**	**Kappa**
1	92.5 %	0.846
2	72.5 %	0.412
3	60.0 %	0.14
4	60.0 %	0.13
5	65.0 %	0.239
6	57.5 %	0.061

6.2.1.2. Akurasi dengan k-NN dengan Normalisasi

Untuk kinerja k-nn dengan jumlah atribut yang berbeda-beda setelah dilakukan normalisasi kinerjanya lebih konstan dan menjanjikan dapat dilihat dari nilai kappa yang lebih konstan daripada percobaan sebelumnya.

Tabel 0.2 Akurasi dengan *classifier* k-nn dengan normalisasi

Classifier: k-NN		
K = 8		
Distance method: Canberra		
Normalize: YES		
Top N	**Accuracy**	**Kappa**
1	95.0 %	0.898

2	95.0 %	0.898
3	90.0 %	0.794
4	90.0 %	0.796
5	85.0 %	0.685
6	80.0 %	0.583

6.2.1.3. Akurasi dengan SVM

Selain menggunakan k-NN dicoba juga menggunakan *classifier support vector machine* menggunakan kernel dot untuk klasifikasi data. Kinerja yang diberikan oleh *classifier* SVM sangat menjanjikan dengan nilai akurasi yang sangat konstan terhadap perubahan jumlah atribut dan nilai kappa yang sangat baik.

Tabel 0.3 Akurasi *classifier* dengan SVM

Classifier: Support vector machine		
Kernel: dot		
Normalize: YES		
Top N	**Accuracy**	**Kappa**
1	92.5 %	0.846
2	92.5 %	0.848
3	90.0 %	0.796
4	90.0 %	0.796
5	90.0 %	0.796
6	90.0 %	0.794

6.2.1.4. Akurasi dengan Neural Net

Digunakan juga *neural network* sebagai perbandingan *classifier* dalam mengklasifikasikan data. Kinerja yang diberikan juga sangat menjanjikan dibuktikan dengan nilai akurasi dan kappa yang tinggi dan konstan terhadap perubahan jumlah atribut.

Tabel 0.4 Akurasi *classifier* dengan Neural Net

Classifier: Neural network		
Normalize: YES		
Top N	**Accuracy**	**Kappa**
1	92.5 %	0.846
2	90.0 %	0.796
3	87.5 %	0.746
4	87.5 %	0.746
5	92.5 %	0.848
6	90.0 %	0.796

6.2.2. Skenario Uji Coba Fungsionalitas

Pada scenario uji coba fungsionalitas merupakan uji coba terhadap kebutuhan fungsional dengan menggunakan metode *black box*. Metode *black box* merupakan metode pengujian yang difokuskan pada pola masukan dan keluaran yang sesuai scenario.

6.2.2.1. Uji Coba Menambah Data Pasien atau *Ground Truth* Baru

Pada uji coba ini administrator akan mendaftarkan pasien atau *ground truth* baru pada sistem. Skenario uji coba dapat dilihat pada **Tabel 6.5**.

Tabel 0.5 Tabel Skenario Uji Coba Menambah Data Pasien Baru

ID	UC-1
Kasus Penggunaan	Menambah Data Pasien atau *Ground Truth* Baru
Sub Kasus	-
Nama	Pengujian mendaftarkan data pasien baru
Tujuan Uji Coba	Menguji fitur pendaftaran sistem
Skenario	Administrator melakukan pendaftaran pasien baru
Langkah Uji Coba	1. Admin membuka halaman beranda 2. Admin menekan tombol "NEW GROUND TRUTH DATA" 3. Admin mengisi identitas singkat dan *upload file* data napas 4. Admin memilih tombol 'Add'
Hasil yang Diharapkan	Data pasien baru terdaftar dan tertampil dalam tabel utama
Hasil yang didapat	Data pasien baru terdaftar dan tertampil dalam tabel utama
Hasil Uji Coba	Berhasil

6.2.2.2. Uji Coba Melihat Profil Pasien serta Hasil Klasifikasi

Pada uji coba ini, administrator akan melihat profil pasien yang berisi identitas singkat dan plot hasil napas pasien serta hasil klasifikasi oleh *classifier*. Skenario uji coba dapat dilihat pada **Tabel 6.6**

Tabel 0.6 Tabel Skenario Uji Coba Melihat Profil Pasien serta Hasil Klasifikasi

ID	UC-2
Kasus Penggunaan	Melihat Data Profil Pasien serta Hasil Klasifikasi
Sub Kasus	-
Nama	Pengujian melihat profil dan klasifikasi pasien
Tujuan Uji Coba	Menguji fitur klasifikasi pasien
Skenario	Administrator melihat profil pasien
Langkah Uji Coba	1. Admin membuka halaman beranda 2. Admin memilih pasien yang ingin dilihar profilnya 3. Admin menekan tombol ‘view’ di sebelah data pasien
Hasil yang Diharapkan	Sistem menampilkan halaman profil pasien yang berisi identitas singkat serta hasil klasifikasi oleh *classifier*
Hasil yang didapat	Sistem menampilkan halaman profil pasien yang berisi identitas singkat serta hasil klasifikasi oleh *classifier*
Hasil Uji Coba	Berhasil

6.2.2.3. Uji Coba Mengubah Data *Ground Truth* atau Pasien Terdaftar

Pada uji coba ini, administrator akan mengubah data pasien terdaftar. Data yang akan dirubah akan ditampilkan dahulu data yang tercata sekarang, kemudian admin akan melakukan input perubahan data yang baru kemudian setelah disimpan maka perubahan data juga terjadi pada basis data. Skenario uji coba dapat dilihat pada **Tabel 6.7**

Tabel 0.7 Tabel Skenario Uji Coba Mengubah Data Pasien Terdaftar

ID	UC-3
Kasus Penggunaan	Mengubah Data *Ground Truth* atau Pasien Terdaftar
Sub Kasus	-
Nama	Pengujian mengubah data pasien terdaftar
Tujuan Uji Coba	Menguji fitur mengubah data
Skenario	Administrator mengubah data pasien
Langkah Uji Coba	1. Admin membuka halaman beranda 2. Admin memilih pasien yang ingin dilihar profilnya 3. Admin menekan tombol ‘edit di sebelah data pasien
Hasil yang Diharapkan	Sistem mencatat perubahan data pasien
Hasil yang didapat	Sistem mencatat perubahan data pasien
Hasil Uji Coba	Berhasil

6.2.2.4. Uji Coba Menghapus Data *Ground Truth* atau Pasien Terdaftar

Pada uji coba ini, administrator akan menghapus data pasien terdaftar. Sebelum menghapus data pasien terdaftar, sistem akan menampilkan *pop-up* konfirmasi apakah yakin data akan dihapus. Skenario uji coba dapat dilihat pada **Tabel 6.8**

Tabel 0.8 Tabel Skenario Uji Coba Menghapus Data Pasien Terdaftar

ID	UC-4
Kasus Penggunaan	Menghapus Data *Ground Truth* atau Pasien Terdaftar
Sub Kasus	-
Nama	Pengujian menghapus data pasien terdaftar
Tujuan Uji Coba	Menguji fitur menghapus data
Skenario	Administrator menghapus data pasien
Langkah Uji Coba	1. Admin membuka halaman beranda 2. Admin memilih pasien yang ingin dihapus profilnya 3. Admin menekan tombol ‘hapus di sebelah data pasien
Hasil yang Diharapkan	Sistem menghapus data pasien
Hasil yang didapat	Sistem menghapus data pasien
Hasil Uji Coba	Berhasil

6.2.2.5. Uji Coba Melihat Seluruh Data Pasien dan *Ground Truth* Terdaftar

Pada uji coba ini, administrator akan melihat seluruh data pasien terdaftar. Skenario uji coba dapat dilihat pada **Tabel 6.9**

Tabel 0.9 Tabel Skenario Uji Coba Melihat Seluruh Data Pasien Terdaftar

ID	UC-5
Kasus Penggunaan	Melihat Seluruh Data Pasien dan *Ground Truth* Terdaftar
Sub Kasus	-
Nama	Pengujian melihat seluruh data pasien terdaftar
Tujuan Uji Coba	Menguji fitur melihat seluruh data pasien terdaftar
Skenario	Administrator melihat seluruh data pasien
Langkah Uji Coba	1. Admin membuka halaman beranda
Hasil yang Diharapkan	Sistem menampilkan seluruh data pasien terdaftar
Hasil yang didapat	Sistem menampilkan seluruh data pasien terdaftar
Hasil Uji Coba	Berhasil

6.3. Evaluasi Uji Coba

Pada bagian ini akan dijelaskan hasil evaluasi dari uji coba yang telah dilakukan pada sub-bab sebelumnya. Evaluasi yang diberikan meliputi evaluasi kinerja *classifier* dan evaluasi uji coba fungsionalitas.

6.3.1. Evaluasi Uji Coba *Classifier*

Hasil uji coba *classifier* mendapatkan hasil dari perhitungan nilai akurasi dan kappa dimana hasil yang paling baik adalah nilai akurasi yang tinggi dan nilai kappa > 0.8. Kinerja SVM dan Neural Net konstan dan menjajikan begitu juga kinerja K-NN dengan Normalisasi.

6.3.2. Evaluasi Uji Coba Fungsionalitas

Hasil uji coba yang dilakukan oleh admin, scenario yang diminta telah berhasil dijalankan oleh sistem, sehingga dapat disimpulkan bahwa fungsionalitas dari aplikasi bekerja sesuai dengan apa yang diharapkan.

BAB VII
KESIMPULAN DAN SARAN

Pada bab ini akan diberikan kesimpulan yang diperoleh selama pengerjaan Tugas Akhir dan saran mengenai pengembangan yang dapat dilakukan terhadap Tugas Akhir ini di masa yang akan datang.

7.1. Kesimpulan

Dari hasil pengamatan selama proses perancangan, implementasi, dan pengujian perangkat lunak yang dilakukan, dapat diambil kesimpulan sebagai berikut:

1. Atribut yang paling berpengaruh yang dihasilkan oleh seleksi fitur adalah nilai rata-rata CO, nilai rata-rata kelembapan, standar deviasi CO_2, standar deviasi ketone, standar deviasi VOC.
2. Atribut temperature tidak digunakan karena dianggap lebih berpengaruh kepada suhu lingkungan.
3. *Classifier* yang diimplementasikan untuk sistem adalah *classifier* k-NN dikarenakan implementasi yang mudah untuk sistem web dan memberikan performa yang cukup menjanjikan.
4. *Classifier* yang digunakan memiliki performa yang sangat baik ditunjukkan oleh nilai *kappa statistic* yang berada pada rentang *near perfect performance*.

7.2. Saran

Berikut merupakan beberapa saran untuk pengembangan sistem di masa yang akan datang. Saran-saran ini didasarkan pada hasil perancangan, implementasi dan pengujian yang telah dilakukan.

1. Pengujian lebih baik dilakukan dengan metode *leave-one-out* karena data yang ada tidak begitu banyak dan cenderung cepat untuk diimplementasikan.
2. Data yang dihimpun kurang banyak sebaiknya lebih ditambah banyak lagi dan lebih banyak rentangnya, salah satunya untuk rentang *prediabetes*.
3. Bisa dikembangkan lebih lanjut untuk mengukur estimasi gula darah karena menurut penelitian sebelumnya ada korelasi antara konsentrasi gas dalam napas dan kadar gula darah.

DAFTAR PUSTAKA

[1] K. Yan, D. Zhang, D. Wu, H. Wei, and G. Lu, "Design of a breath analysis system for diabetes screening and blood glucose level prediction," *IEEE Trans. Biomed. Eng.*, vol. 61, no. 11, pp. 2787–2795, 2014.

[2] Anonim, "Satuan Ukuran Gula Darah mmol/L dan mg/dL," 2009. [Online]. Available: http://indodiabetes.com/satuan-ukuran-gula-darah-mmoll-dan-mgdl.html. [Accessed: 15-Dec-2016].

[3] S. R., "PENYEBAB DAN PENANGANAN DIABETES BAU MULUT," 2016. [Online]. Available: https://diabetics1.com/2016/03/penyebab-bau-mulut-penderita-diabetes.html. [Accessed: 15-Dec-2016].

[4] P. R. Galassetti *et al.*, "Breath Ethanol and acetone as indicators of serum glucose levels: an initial report," *Diabetes Technol.Ther.*, vol. VII, no. 1, pp. 115–123, 2005.

[5] P. Paredi, W. Biernacki, G. Invernizzi, S. A. Kharitonov, and P. J. Barnes, "Exhaled Carbon Monoxide Levels Elevated in Diabetes and Correlated With Glucose Concentration in Blood: A New Test for Monitoring the Disease?," *Chest*, vol. VII, no. 1, pp. 115–123, 2005.

[6] M. Philips, R. N. Cataneo, T. Cheema, and J. Greenberg, "Increased breath biomarkers of oxidative stress in diabetes melitus," *Clin. Chim. Acta*, vol. 1, pp. 189–194, 2004.

[7] B. Novak *et al.*, "Exhaled methyl nitrate as a noninvasive marker of hyperglycemia in type 1 diabetes," *Proc. Natl. Acad. Sci.*, vol. 40, pp. 15613–15618, 2007.

[8] H. Gunawan, *Prinsip-prinsip Elektronik edisi ke-2*, 2nd

ed. Jakarta: Erlangga, 2010.

[9] R. Tem, U. Tem, and S. Tem, "Technical Mq-2 Gas Sensor," vol. 1, p. 2.

[10] R. Tem, "Technical MQ-135 Gas Sensor," vol. 1, pp. 3–4.

[11] R. Tem, "Mq-138 Gas Sensor," vol. 1, pp. 3–4.

[12] D. R. Wijaya, R. Sarno, and E. Zulaika, "Gas concentration analysis of resistive gas sensor array," *2016 Int. Symp. Electron. Smart Devices*, pp. 337–342, 2016.

[13] M. Rouse, "data preprocessing." [Online]. Available: http://searchsqlserver.techtarget.com/definition/data-preprocessing. [Accessed: 03-Jun-2017].

[14] H. Kim *et al.*, "Electronic-nose for detecting environmental pollutants: Signal processing and analog front-end design," *Analog Integr. Circuits Signal Process.*, vol. 70, no. 1, pp. 15–32, 2012.

[15] X. Guo *et al.*, "A novel feature extraction approach using window function capturing and QPSO-SVM for enhancing electronic nose performance," *Sensors (Switzerland)*, vol. 15, no. 7, pp. 15198–15217, 2015.

[16] R. Bott, "About Feature Scaling and Normalization - and the effect of standardization for machine learning algorithms," *Igarss 2014*, 2014. [Online]. Available: http://sebastianraschka.com/Articles/2014_about_feature_scaling.html. [Accessed: 01-Jul-2017].

[17] A. DLI, "Stationary Signals," 2009. [Online]. Available: http://www.azimadli.com/vibman/stationarysignals.htm. [Accessed: 01-Jul-2017].

[18] D. R. Wijaya, R. Sarno, and E. Zulaika, "Sensor array optimization for mobile electronic nose: Wavelet transform and filter based feature selection approach," *Int. Rev. Comput. Softw.*, vol. 11, no. 8, pp. 659–671, 2016.

[19] D. Guo, D. Zhang, L. Zhang, and G. Lu, "Non-invasive blood glucose monitoring for diabetics by means of breath signal analysis," *Sensors Actuators, B Chem.*, vol. 173, pp. 106–113, 2012.

[20] Math.NET, "Distance Metrics." [Online]. Available: https://numerics.mathdotnet.com/distance.html. [Accessed: 01-Jul-2017].

[21] A. Fauzan, "Penerapan Cross Validation Sebagai Metode Pengujian Sistem," 2015. [Online]. Available: http://www.charisfauzan.net/2015/02/penerapan-cross-validation-sebagai.html. [Accessed: 03-Jul-2017].

[22] U. of R. Department of Computer Sciences, "Confusion Matrix." [Online]. Available: http://www2.cs.uregina.ca/~dbd/cs831/notes/confusion_matrix/%0Aconfusion_matrix.html%0A. [Accessed: 03-Jul-2017].

[23] R. Zacharski, "A Programmer's Guide to Data Mining Chapter 5," in *A Programmer's Guide to Data Mining*, pp. 5–20.

[24] J. Landis and G. Koch, "Kappa Statistic," in *The measurement of observer agreement for categorical data*, 1977, pp. 159–74.

[Halaman ini sengaja dikosongkan]

BIODATA PENULIS

Hariyanto, lahir di Surabaya pada 21 April 1995. Menempuh Pendidikan formal di SDK Santa Angela Surabaya (2001-2007), SMPK Angelus Custos 1 Surabaya (2007-2010), SMAK Frateran Surabaya (2010-2013), dan Pendidikan S1 di Teknik Informatika ITS (2013-2017). Selama masa kuliah, penulis aktif dalam organisasi Himpunan Mahasiswa Teknik Computer-Informatika (HMTC), Schematics, dan ITS Expo. Diantaranya menjadi staf Departemen Kesejahteraan Mahasiswa, Staf dan Staf Ahli *National Seminar of Techology Schematics*, dan Staf Ahli Akomodasi dan Transportasi ITS Expo. Penulis menyukai riset dan pernah mendapatkan danah hibah untuk PKM-KC pada tahun 2015 dengan judul "INDOHEALTHS: *Smart Medical Assistant* yang Terhubungn dengan Pelayanan Kesehatan". Bidang minat yang ditekuni penulis adalah Komputasi Cerdas dan Visi (KCV) dan Manajemen Informasi (MI), diantaranya *machine learning, data mining, big data, image processing* dan *data science*. Penulis dapat dihubungi melalui: **hariyanto.tan95@gmail.com**