TUGAS AKHIR - KI141502

KLASIFIKASI JENIS GANGGUAN PADA *BASE TRANSCEIVER STATION* (BTS) MENGGUNAKAN ALGORITMA *K-NEAREST NEIGHBOR*, *NAÏVE BAYES* DAN *GENETIC ALGORITHM*

WIDA DWITIAYASA
NRP 5114100155

Dosen Pembimbing I
Arya Yudhi Wijaya, S.Kom., M.Kom.

Dosen Pembimbing II
Bilqis Amaliah, S.Kom., M.Kom.

DEPARTEMEN INFORMATIKA
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember
Surabaya 2018

[Halaman ini sengaja dikosongkan]

TUGAS AKHIR - KI141502

KLASIFIKASI JENIS GANGGUAN PADA *BASE TRANSCEIVER STATION* (BTS) MENGGUNAKAN ALGORITMA *K-NEAREST NEIGHBOR*, *NAÏVE BAYES* DAN *GENETIC ALGORITHM*

WIDA DWITIAYASA
NRP 5114100155

Dosen Pembimbing I
Arya Yudhi Wijaya, S.Kom., M.Kom.

Dosen Pembimbing II
Bilqis Amaliah, S.Kom., M.Kom.

Departemen Informatika
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember
Surabaya 2018

[Halaman ini sengaja dikosongkan]

UNDERGRADUATE THESES – KI141502

CLASSIFICATION TYPE OF INTERFERENCE ON BASE TRANSCEIVER STATION (BTS) USING K-NEAREST NEIGHBORS, NAÏVE BAYES AND GENETIC ALGORITHM

WIDA DWITIAYASA
NRP 5114100155

Supervisor I
Arya Yudhi Wijaya, S.Kom., M.Kom.

Supervisor II
Bilqis Amaliah, S.Kom., M.Kom.

Department of Informatics
Faculty of Information Technology and Communication
Sepuluh Nopember Institute of Technology
Surabaya 2018

[Halaman ini sengaja dikosongkan]

LEMBAR PENGESAHAN

KLASIFIKASI JENIS GANGGUAN PADA *BASE TRANSCEIVER STATION* (BTS) MENGGUNAKAN ALGORITMA *K-NEAREST NEIGHBOR, NAÏVE BAYES* DAN *GENETIC ALGORITHM*

TUGAS AKHIR

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

Oleh :
WIDA DWITIAYASA
NRP : 5114 100 155

Disetujui oleh Dosen Pembimbing Tugas Akhir :

Arya Yudhi Wijaya, S.Kom., M.Kom.	.....
NIP: 19840904 201012 1 002	(pembimbing 1)
Bilqis Amaliah, S.Kom., M.Kom.	.....
NIP: 19750914 200112 2 002	(pembimbing 2)

SURABAYA
JANUARI 2018

[Halaman ini sengaja dikosongkan]

KLASIFIKASI JENIS GANGGUAN PADA *BASE TRANSCEIVER STATION* (BTS) MENGGUNAKAN ALGORITMA *K-NEAREST NEIGHBOR, NAÏVE BAYES* DAN *GENETIC ALGORITHM*

Nama Mahasiswa	**: WIDA DWITIAYASA**
NRP	**: 5114100155**
Jurusan	**: Teknik Informatika FTIF-ITS**
Dosen Pembimbing 1	**: Arya Yudhi Wijaya, S.Kom., M.Kom.**
Dosen Pembimbing 2	**: Bilqis Amaliah. S.Kom., M.Kom.**

ABSTRAK

Permintaan dan penggunaan layanan data, *payload* maupun SMS yang semakin meningkat mengakibatkan sering terjadi kerusakan pada sistem *Base Transceiver Station* (BTS). Namun demikian deteksi kerusakan tersebut masih dilakukan secara manual, sehingga dinilai kurang optimal dalam segi waktu, subyektifitas, dan tingkat keakuratan.

Untuk mengatasi masalah tersebut, tugas akhir ini mencoba melakukan implementasi pengklasifikasian jenis kerusakan pada BTS secara otomatis menggunakan algoritma *K-Nearest Neighbor*, *Naive Bayes*, dan *Genetic Algorithm*. Dataset yang digunakan dalam proses uji coba adalah data BTS dari salah satu operator seluler di Indonesia. Visualisasi dari penerapan algoritma diatas menggunakan perangkat lunak Bot Telegram.

Hasil uji coba menunjukkan tingkat akurasi terbaik sebesar 95.45% sehingga algoritma ini dapat digunakan dalam pengklasifikasian jenis kerusakan pada BTS.

Kata kunci: K-Nearest Neighbors, Naïve Bayes, Genetic Algorithm, BTS, Klasifikasi

[Halaman ini sengaja dikosongkan]

CLASSIFICATION TYPE OF INTERFERENCE ON BASE TRANSCEIVER STATION (BTS) USING K-NEAREST NEIGHBORS, NAÏVE BAYES AND GENETIC ALGORITHM

Student's Name	**: WIDA DWITIAYASA**
Student's ID	**: 5114100155**
Department	**: Teknik Informatika FTIF-ITS**
First Advisor	**: Arya Yudhi Wijaya, S.Kom., M.Kom.**
Second Advisor	**: Bilqis Amaliah, S.Kom., M.Kom.**

ABSTRACT

Development of technology make communication easier. This is indicated by the increasing of cellular features that can answer user needs such as data services, payloads, and SMS. However, increasing of demand and usage causes frequent damage to the system. This condition impacts on the unavailability of user requirements as expected.

To overcome this problem, this final project tries to implement the classification of damage type in Base Transceiver Station (BTS) automatically by using K-Nearest Neighbor algorithm, Naive Bayes, and supported by Genetic Algorithm method. The dataset used in this experiment is data BTS from one of the cellular operators in Indonesia. Visualization of the algorithm implementation uses Telegram Bot software.

The test results show the best accuracy is 95.45% so this algorithm can be used in classifying the type of damage to the BTS.

Keywords: K-Nearest Neighbors, Genetic Algorithm, BTS, Classification

[Halaman ini sengaja dikosongkan]

KATA PENGANTAR

Astungkara, segala puji dan syukur bagi Ida Sang Hyang Widhi Wasa, yang telah melimpahkan nikmat, dan rejeki-Nya sehingga penulis dapat menyelesaikan Tugas Akhir yang berjudul

"KLASIFIKASI JENIS GANGGUAN PADA *BASE TRANSCEIVER STATION* (BTS) MENGGUNAKAN ALGORITMA *K-NEAREST NEIGHBOR, NAÏVE BAYES* DAN *GENETIC ALGORITHM.*"

Pengerjaan Tugas Akhir ini merupakan suatu kesempatan yang berharga bagi penulis. Dengan pengerjaan Tugas Akhir, penulis dapat memperdalam, meningkatkan, serta menerapkan apa yang telah didapatkan penulis selama menempuh perkuliahan di Teknik Informatika ITS.

Terselesaikannya Tugas Akhir ini tidak lepas dari bantuan dan dukungan dari berbagai pihak. Dan dalam kesempatan ini penulis mengucapkan rasa syukur dan terima kasih kepada:

1. Ida Sang Hyang Widhi Wasa, karena atas izin-Nya lah penulis dapat menyelesaikan Tugas Akhir dengan baik.
2. Kedua orang tua, Made Mertayasa serta Ni Ketut Supartiani, dan kedua saudara Yoga Partamayasa serta Bayu Trianayasa, terima kasih atas doa dan bantuan moral dan material selama penulis belajar di Teknik Informatika ITS.

3. Bapak Dr. Darlis Herumurti, S.Kom., M.Kom., selaku ketua jurusan Teknik Informatika ITS
4. Bapak Radityo Anggoro, S.Kom., M.Sc. selaku Koordinator Tugas Akhir di Teknik Informatika ITS.
5. Bapak Radityo Anggoro, S.Kom., M.Sc. selaku Dosen Perwalian penulis di Teknik Informatika ITS yang telah memberikan semangat dan motivasi kepada penulis untuk menyelesaikan Tugas Akhir.
6. Bapak Victor Hariadi, S.Si.,M.Kom selaku Kepala Rumpun Mata Kuliah Dasar Terapan Komputasi yang telah memberikan semangat dan motivasi kepada penulis untuk menyelesaikan Tugas Akhir.
7. Bapak Arya Yudhi Wijaya, S.Kom, M.Kom selaku pembimbing I Tugas Akhir yang telah memberikan banyak waktu untuk berdiskusi dan memberi semangat serta motivasi kepada penulis untuk menyelesaikan Tugas Akhir.
8. Ibu Bilqis Amaliah, S.Kom, M.Kom selaku pembimbing II Tugas Akhir yang telah memberikan banyak waktu untuk berdiskusi dan memberi semangat serta motivasi kepada penulis untuk menyelesaikan Tugas Akhir.
9. Bapak dan Ibu Dosen di Jurusan Teknik Informatika yang telah memberikan ilmu selama penulis kuliah di Teknik Informatika.
10. Seluruh Staf dan karyawan Teknik Informatika yang telah memberikan bantuan selama penulis kuliah di Teknik Informatika.
11. Rekan-rekan di laboratorium Dasar Terapan Komputer mas Yuda, mas Ipul, mas Ichang, mas Andre, mas Ardhana, mas Hendro, mas Reza, dll. Yang telah mendukung pengerjaan Tugas Akhir.
12. Seluruh rekan-rekan TC 2014 yang saya banggakan.

13. Rekan-rekan Kontrakan Sutorejo Timur XI Rika, Tara, Tari, Chestha, Widya, Cyndi dan Anggita, terimakasih banyak atas segala dukungan dan motivasi yang diberikan selama penulis menempuh kuliah di Teknik Informatika ITS.
14. Rekan-rekan Benalu Lucha, Uyun, Afiif, terimakasih banyak atas segala dukungan dan motivasi yang diberikan selama penulis menempuh kuliah di Teknik Informatika ITS.
15. Rekan-rekan TPKH-ITS, terimakasih banyak telah memberikan penulis banyak pelajaran dan pengalaman. Terutama pada angkatan 2014 yang saya cintai.
16. Rekan-rekan kerja, Yusuf, Valdy, Aufar, Dzaky, Nobby, Widhi, Fahmi dan Hari, terimakasih atas dukungan dan motivasi yang diberikan selama penulis menempuh kuliah di Teknik Informatika ITS.

Penulis memohon maaf apabila terdapat kekurangan dalam penulisan Tugas Akhir ini. Kritik dan saran penulis harapkan untuk perbaikan dan pembelajaran di kemudian hari. Semoga Tugas Akhir ini dapat memberikan Manfaat yang sebesar besarnya.

Surabaya, Januari 2018

Wida Dwitiayasa

[Halaman ini sengaja dikosongkan]

DAFTAR ISI

[Halaman ini sengaja dikosongkan]

DAFTAR GAMBAR

[Halaman ini sengaja dikosongkan]

DAFTAR TABEL

[Halaman ini sengaja dikosongkan]

DAFTAR KODE SUMBER

[Halaman ini sengaja dikosongkan]

BAB I
PENDAHULUAN

Pada bab ini dibahas mengenai latar belakang, rumusan masalah, batasan masalah, tujuan, manfaat, metodologi, dan sistematika laporan tugas akhir. Diharapkan dari penjelasan dalam bab ini gambaran tugas akhir secara umum dapat dipahami.

1.1. Latar Belakang

Sebagai makhluk sosial, komunikasi merupakan hal yang penting dan sering dianggap sebagai kebutuhan yang harus dipenuhi oleh setiap manusia. Dengan berkembangnya teknologi informasi dewasa ini, komunikasi dapat dilakukan dengan mudah. Jarak dan waktu bukan lagi menjadi halangan yang besar di dunia yang serba *online* seperti sekarang. Dengan demikian tentunya menyebabkan tingginya trafik kebutuhan layanan data/payload, *voice* maupun *SMS*. Hal tersebut mendorong banyak pengembang operator seluler untuk mengembangkan fitur – fitur yang mampu menjawab kebutuhan pengguna. Salah satunya berupa pengembangan teknologi jaringan dari 1G, 2G, 3G hingga kini memasuki layanan 4G. Namun meski demikian, teknologi-teknologi canggih ini tetap memiliki batasan kemampuan yang tidak jarang akan menimbulkan kerusakan (*error*) pada sistem tersebut. Informasi tentang kerusakan system sejauh ini masih dianalisa secara manual sehingga menyebabkan keterlambatan penanganan yang mengakibatkan tidak tercapainya kebutuhan pengguna sebagaimana yang diharapkan.

Pada tugas akhir ini, penulis berupaya melakukan otomatisasi pengklasifikasian jenis kerusakan yang terjadi pada *Base Transceiver Station* (BTS) tersebut. Salah satu metode klasifikasi yang umum digunakan adalah *K-Nearest*

Neighbors (KNN) dan *Naïve Bayes*. KNN merupakan salah satu algoritma klasifikasi yang diproses berdasarkan tipe dari data model. Setiap data model *t*(*testing)* yang diberikan, tipe kelasnya ditentukan dengan mencari sebanyak *K* data model yang memiliki kemiripan tertinggi dengan *t* dan kemudian tipe kelas *t* diprediksi dengan menghitung mayoritas *votes* dari *K* data model. Akan tetapi algoritma KNN memiliki beberapa kelemahan dimana salah satunya adalah perlunya menentukan nilai dari parameter *K* (jumlah tetangga terdekat) secara manual sebagai tahap awal menggunakan metode ini. Namun untuk mengatasi kelemahan tersebut, pada tugas akhir ini KNN akan didukung oleh salah satu metode *clustering* yaitu *Genetic Algorithm* (GA). GA berperan dalam membantu KNN melakukan penentuan nilai *K* secara terus menerus hingga metode KNN mengembalikan nilai yang paling optimal. Hal ini merupakan salah satu kelebihan dari GA yaitu memiliki kemampuan untuk mencari nilai optimal secara pararel melalui proses kerjasama antara berbagai unit yang disebut kromosom individu. Selanjutnya hasil keluaran dari KNN akan dikombinasikan menggunakan algoritma *Naïve Bayes,* dimana algoritma ini merupakan salah satu algoritma klasifikasi sederhana yang dihitung berdasarkan nilai probabilitas dengan asumsi independesi fitur yang kuat sehingga diharapkan dapat meningkatkan performa serta kualitas output dari KNN.

Dengan adanya proses sesuai yang disebutkan diatas, diharapkan pengklasifikasian jenis *trouble* pada BTS akan menjadi lebih efisien dan akurat.

1.2. Rumusan Permasalahan

Rumusan masalah yang diangkat dalam tugas akhir ini dapat dipaparkan sebagai berikut:

1. Bagaimana efek implementasi GA dalam mendukung algoritma KNN?
2. Bagaimana efek pengukuran performa dari kombinasi GA, KNN dan *Naïve Bayes*?
3. Bagaimana efek dalam melakukan evaluasi data untuk menetukan kelompok data uji menggunakan algroitma *K-fold validation*?
4. Bagaimana efek dalam melakukan implementasi metode normalisasi?
5. Bagaimana efek metode yang digunakan dalam menghitung jarak pada algoritma KNN?
6. Bagaimana hasil kesimpulan uji coba seluruh dataset dengan menggunakan seluruh skenario pengujian tanpa menggunakan normalisasi data?
7. Bagaimana hasil kesimpulan uji coba seluruh dataset dengan menggunakan seluruh skenario pengujian dengan menggunakan normalisasi data?

1.3. Batasan Masalah

Permasalahan yang dibahas dalam tugas akhir ini memiliki beberapa batasan, yaitu sebagai berikut:

1. Menggunakan bahasa pemrograman *Python 2.7.*
2. Dataset yang digunakan diambil dari BTS salah satu perusahaan telekomunikasi di Indonesia.
3. Metode utama yang digunakan adalah kombinasi KNN dan *Naïve Bayes* dengan dukungan GA. KNN digunakan sebagai metode pembanding. *Naïve Bayes* digunakan pula sebagai metode pembanding.
4. Metode perhitungan jarak yang dibandingkan adalah *eucidean distance, manhattan distance* dan *Chebyshev distance.*

5. Metode persilangan yang digunakan pada GA berupa *Single-point crossover* dan *gene mutation*.
6. Metode normalisasi data menggunakan *min-max algorithm* .
7. Evaluasi dilakukan terhadap *accuracy* dataset menggunakan metode *Confusion Matrix*.
8. Validasi dataset dilakukan dengan *K-fold validation* sebanyak 5 fold.
9. Jumlah iterasi yang digunakan sebanyak 100 iterasi.
10. *Deployment* aplikasi dilakukan dalam bentuk bot Telegram.

1.4. Tujuan

Tugas Akhir ini mempunyai beberapa tujuan, yaitu sebagai berikut:

1. Mengimplementasikan metode GA untuk mendukung KNN dalam mencari nilai *K* untuk menyelesaikan permasalahan klasifikasi jenis *trouble* BTS serta mengkombinasikan metode *Naïve Bayes* demi meningkatkan performa dan kualitas hasil output.
2. Membangun sistem pemberitahuan *trouble* secara otomatis melalui salah satu aplikasi *chatting*.

1.5. Manfaat

Pengerjaan tugas akhir ini dilakukan dengan harapan dapat memberikan kontribusi pada salah satu perusahaan operator seluler di Indonesia dalam mendeteksi permasalahan pada BTS.

1.6. Metodologi

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

1. Penyusunan proposal Tugas Akhir.

   Tahap awal yang dilakukan dalam pengerjaan Tugas Akhir ini adalah penyusunan proposal Tugas Akhir. Di dalam proposal diajukan suatu gagasan pembuatan sistem untuk melakukan klasifikasi jenis *trouble* pada BTS menggunakan metode GA dan KNN.

2. Studi literatur

   Pada tahap ini dilakukan pencarian, pengumpulan, penyaringan, pemahaman, dan pembelajaran literatur yang berhubungan dengan metode *Min-Max Normalization,* GA, KNN, *Naïve Bayes*, *Confusion Matrix*, dan *K-fold validation*. Literatur yang digunakan meliputi: paper, jurnal, dan dokumentasi internet.

3. Perancangan perangkat lunak

   Tahap ini meliputi perancangan sistem berdasarkan studi literatur dan pembelajaran konsep teknologi dari perangkat lunak yang ada. Tahap ini mendefinisikan alur dari implementasi. Langkah-langkah yang dikerjakan juga didefinisikan pada tahap ini. Pada tahapan ini dibuat *prototype* sistem, yang merupakan rancangan dasar dari sistem yang akan dibuat. Kemudian dilakukan desain sistem dan desain proses-proses yang ada.

4. Implementasi perangkat lunak

   Implementasi merupakan tahap membangun rancangan program yang telah dibuat. Pada tahapan ini merealisasikan rancangan yang terdapat pada tahapan sebelumnya, sehingga menjadi sebuah program yang sesuai dengan apa yang telah direncanakan.

5. Pengujian dan evaluasi

   Pada tahap ini dilakukan uji coba pada data yang telah dikumpulkan. Pengujian dan evaluasi akan dilakukan dengan menggunakan bahasa *Python 2.7*. Tahapan ini dimaksudkan untuk mengevaluasi kesesuaian data dan program serta mencari masalah yang mungkin timbul dan mengadakan perbaikan jika terdapat kesalahan.

6. Penyusunan buku Tugas Akhir.

   Pada tahapan ini disusun buku yang memuat dokumentasi mengenai pembuatan serta hasil dari implementasi perangkat lunak yang telah dibuat.

1.7. Sistematika Penulisan Laporan Tugas Akhir

Buku Tugas Akhir ini bertujuan untuk mendapatkan gambaran dari proses pengerjaan tugas akhir. 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 penjelasan mengenai latar belakang masalah, tujuan, dan manfaat dari pembuatan Tugas Akhir. Selain itu rumusan permasalahan, batasan masalah, dan sistematika penulisan juga merupakan bagian dari bab ini.

Bab II Dasar Teori

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

Bab III Perancangan Perangkat Lunak

Bab ini berisi penjelasan mengenai desain dan perancangan sistem yamg direpresentasikan dalam bentuk *pseudocode* dan *flowchart*.

Bab IV Implementasi

Bab ini merupakan pembangunan sistem menggunakan *Spyder 3* sesuai permasalahan dan batasan yang telah dijabarkan pada Bab I serta merupakan implementasi dari desain yang telah dibuat pada bab sebelumnya.

Bab V Uji Coba dan Evaluasi

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

Bab VI Kesimpulan dan Saran

Bab ini merupakan bab terakhir yang menyampaikan kesimpulan dari hasil uji coba yang diakukan serta saran untuk hal-hal yang masih dapat dikerjakan dengan lebih baik dan dapat dikembangkan lebih lanjut maupun masalah-masalah yang dialami pada proses pengerjaan Tugas Akhir.

[Halaman ini sengaja dikosongkan]

BAB II
DASAR TEORI

Bab ini berisi penjelasan teori-teori yang berkaitan dengan algoritma yang diajukan dalam pengimplementasian sistem tugas akhir. Penjelasan ini bertujuan untuk memberikan gambaran secara umum terhadap sistem yang dibangun dan berguna sebagai teori yang mendasari pengembangan perangkat lunak.

2.1 Base Transceiver Station (BTS)

BTS adalah suatu perangkat dalam jaringan telekomunikasi seluler yang berbentuk sebuah tower dengan antenna pemancar dan penerima yang berfungsi sebagai penguat sinyal daya, sehingga dapat menghubungkan jaringan operator telekomunikasi seluler dengan pelanggannya [1]. Dengan tingginya tingkat permintaan akses dari pengguna, tidak jarang apabila suatu BTS berada dalam keadaan *downtime* atau sejumlah satuan waktu dimana BTS tersebut tidak dapat beroperasi dengan baik yang disebabkan oleh adanya kerusakan (*failure*). Terdapat 3 jenis kerusakan yang dimaksud yaitu *Bad Voice, Low Throughput* dan *Low Coverage*.

2.1.1 *Bad Voice*

Bad voice atau *drop call* adalah kegagalan panggilan yang terjadi setelah panggilan berakhir tanpa pemutusan secara normal. Akan tetapi tidak semua *drop call* itu menyebabkan terputusnya sambungan melainkan dapat hanya

terjadi interfrensi koneksi tanpa terputusnya sambungan komunikasi [2]. Kerusakan ini terjadi diakibatkan oleh:

2.1.1.1.1.1.1.1 RTWP (*Received Total Wideband Power*)

RTWP merupakan total daya yang diterima oleh jaringan. Nilai ini dapat dijadikan suatu indikator/parameter sebagai acuan suatu site mengalami interferensi *uplink* atau tidak, serta dapat membantu analisis dan solusi penanganan interferensi *uplink* pada suatu site yang bersangkutan [3]. Dalam tugas akhir ini, acuan nilai RTWP adalah **<=-100 dbm** [4] sesuai dengan standard yang dimiliki oleh salah satu perusahaan telekomunikasi di Indonesia.

2.1.1.1.1.1.1.2 ULBLER (*Uplink Block Error Rate*)

ULBLER merupakan perbandingan jumlah blok yang salah atau *error* dengan jumlah keseluruhan blok yang diterima pada sebuah rangkaian digital [5]. Dalam tugas akhir ini, acuan nilai ULBLER adalah **0-10%** [6] sesuai dengan standard yang dimiliki oleh salah satu perusahaan telekomunikasi di Indonesia.

2.1.2 *Low Throughput*

Throughput merupakan jumlah data digital per satuan waktu yang dikirimkan melalui baik *link* fisik maupun *logic*, atau yang mengalir dari suatu node ke node lain. Sebagai contoh, *throughput* bisa merupakan jumlah data yang disampaikan ke sebuah terminal jaringan tertentu atau sebuah host. *Throughput* biasanya diukur dalam bit per detik (bps), kadang-kadang dalam banyak paket per detik [7]. Sehingga *throughput* dapat dianalogikan sebagai besar pemakaian *bandwith*. Rendahnya nilai *throughput* ini terjadi diakibatkan oleh :

a. PRB (*Physical Resource Block*)

PRB merupakan unit terkecil dari *bandwith* yang dibagi-bagi oleh *scheduler* pada BTS. Dalam tugas akhir ini, acuan nilai PRB adalah **<=70%** [8] sesuai dengan standard yang dimiliki oleh salah satu perusahaan telekomunikasi di Indonesia.

2.1.3 *Low Coverage*

Low Coverage adalah jenis kerusakan jaringan berupa tersedianya signal akses yang tidak berkualitas, sehingga signal yang ada tersebut tidak dapat digunakan sebagaimana yang diharapkan. Kerusakan ini terjadi diakibatkan oleh :

a. Ec/No (*Energy Chip to Noise*)

Ec/No merupakan rasio perbandingan antara energi yang dihasilkan dari sinyal pilot dengan total energi yang diterima. Nilai ini juga menunjukan level daya minimum (*threshold*) dimana user masih dapat melakukan suatu panggilan [5]. Dalam tugas akhir ini, acuan nilai Ec/No adalah **0 hingga (-8)dB** [9] sesuai dengan standard yang dimiliki oleh salah satu perusahaan telekomunikasi di Indonesia.

b. RSCP (*Received Signal Code Power*)

RSCP merupakan kuat sinyal penerimaan yang menyatakan besarnya daya pada satu kode yang diterima oleh UE (*User Equipment*) yang merupakan salah satu parameter penentu nilai Ec/No . Nilai ini merupakan suatu nilai yang menunjukkan level kekuatan sinyal. Dalam tugas akhir ini, acuan nilai RSCP adalah **0 hingga (-95)db** [9]sesuai dengan standard yang dimiliki oleh salah satu perusahaan telekomunikasi di Indonesia.

2.2 *Dataset Base Transceiver Station (BTS)*

Dataset BTS diambil dari salah satu perusahaan telekomunikasi di Indonesia. Terdiri atas 6 atribut dengan keseluruhan atribut bertipe data numerik yang merepresentasikan aktifitas 557 BTS di jawa timur pada pukul 12.00. Dataset ini mempunyai 557 *record* dengan 228 BTS terindikasi normal, 304 BTS terindikasi *bad voice*, 1 BTS terindikasi *low throughput*, 24 BTS terindikasi *low coverage*. Potongan dataset dapat dilihat pada Lampiran.

2.3 *Min-Max Algorithm*

Algoritma Min-Max [10] merupakan salah satu metode dalam melakukan normalisasi data. Normalisasi data adalah sebuah metode untuk mengatasi ketidakseimbangan nilai yang terdapat pada suatu dataset maka nilai-nilai tersebut harus diubah kedalam suatu rentang yang sama. Cara kerja algoritma ini adalah dengan melakukan penskalaan data pada rentang tertentu, dimana rentang yang digunakan umumnya adalah 0-1. Rumus untuk algoritma ini adalah sebagai berikut :

$$X_n = \frac{X_0 - X_{min}}{X_{max} - X_{min}} \qquad (2.1)$$

2.4 *K-Fold Validation*

K-Fold Validation [11] adalah sebuah metode validasi untuk mengukur atau mengestimasi performa dari sebuah model data. Caranya adalah dengan memotong dataset menjadi k bagian, lalu setiap subset ke – k digunakan sebagai

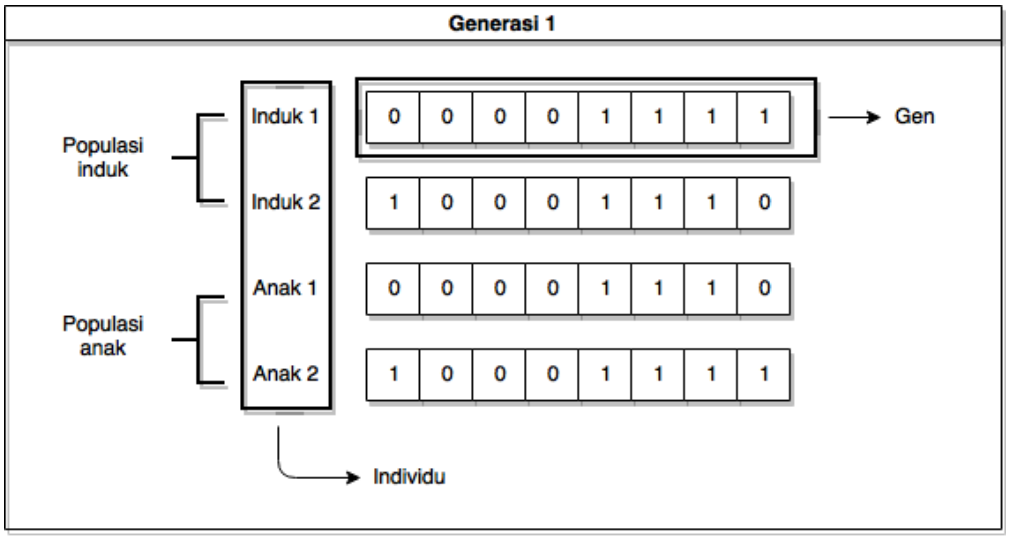
data *testing* sedangkan sisanya digunakan sebagai data *training* dengan model pembelajaran yang telah kita pilih. Sehingga setiap data akan pernah menjadi data *training* dan juga pernah menjadi data *testing*. Subset yang menghasilkan nilai paling baik itu lah yang kemudian akan dipilih menjadi tetapan *model training* dan *testing*. Pada tugas akhir ini, akan digunakan sebanyak 5-fold validation. Ilustrasi metode *K-Fold Validation* ditunjukkan pada Gambar 2.1.

Gambar 2.1 Ilustrasi K-Fold Validation

2.5 *Genetic Algorithm*

GA merupakan algoritma yang memanfaatkan proses seleksi alamiah yang dikenal sebagai proses evolusi. Dalam proses evolusi, individu secara terus-menerus mengalami perubahan gen untuk menyesuaikan dengan lingkungan hidupnya, maka pada akhirnya hanya individu-individu yang kuatlah yang mampu bertahan. GA merupakan metode algoritma yang "buta" karena GA tidak mengetahui kapan

dirinya telah mencapai solusi optimal sehingga GA akan terus melakukan proses hingga mencapai batas iterasi tertentu.



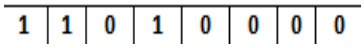
Gambar 2.2Ilustrasi istilah pada GA

Terdapat beberapa istilah dalam GA yang telah diilustrasikan pada Gambar 2.2, yakni:

1. Gen
   Gen adalah bagian dari tiap individu/kromosom yang berisi parameter sesuai permasalahan yang akan dicari.
2. Individu/kromosom
   Kromosom adalah serangkaian gen yang merepresentasikan solusi sesuai dengan permasalahan yang akan dicari.
3. Populasi
   Populasi adalah kumpulan dari beberapa kromosom yang akan diproses pada setiap generasi.
4. Generasi
   Generasi adalah kumpulan populasi yang diperoleh dari operasi GA.

GA mengandung beberapa operasi penting didalamnya, yaitu *encoding*, *crossover*, mutasi dan evaluasi. Tahap awal GA berupa pemilihan populasi solusi potensial yang selanjutnya disebut pemilihan *parent*. Berikut akan dijelaskan masing-masing operasi yang ada pada GA.

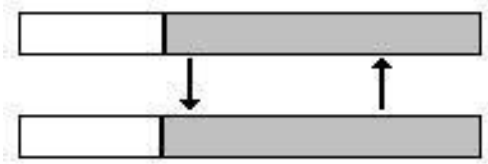
2.5.1 *Encoding*



Gambar 2.3 Ilustrasi Bit-String Encoding

Dalam GA, kromosom yang akan diproses merupakan representasi dari permasalahan yang akan dicari solusi optimalnya. Untuk mempermudah operasi, dilakukan *encoding* dalam kromosom. Terdapat banyak sekali tipe *encoding*, salah satunya adalah *bit-string encoding*, yaitu kromosom direpresentasikan menjadi bilangan biner (0 atau 1) sepanjang 8 bit. Ilustrasi *bit-string encoding* ditunjukkan pada Gambar 2.3

2.5.2 Crossover



Gambar 2.4 Proses crossover dengan *single-point crossover*

Crossover adalah operator algoritma genetika yang membutuhkan parameter dua kromosom. Dua buah kromosom tersebut merupakan kromosom induk. Operator ini akan

menghasilkan dua buah kromosom baru yang disebut kromosom anak. Terdapat beberapa jenis *crossover* yang sering digunakan dalam algoritma genetika, salah satunya adalah *Single-point crossover* [12]. Langkah kerja pertama operator ini adalah dengan menentukan *crossover point* (posisi pemotongan gen). Kromosom anak pertama berisi gen pertama sampai gen *crossover point* dari kromosom induk pertama ditambah dengan gen dari *crossover point* sampai gen terakhir dari kromosom induk kedua. Begitu pula sebaliknya terhadap kromosom anak kedua berisi gen pertama sampai gen *crossover point* dari induk kedua ditambah dengan gen dari *crossover point* sampai gen terakhir dari kromosom induk pertama. Adapun metode *crossover* ini dapat diilustrasikan pada Gambar 2.4

Dari ilustrasi pada Gambar 2.4, maka contoh penerapan metode *single-point crossover* adalah sebagai berikut :
Induk 1 : 1 2 3 4 | 5 6 7 8
Induk 2 : A B C D | E F G H
Setelah proses *crossover*, 2 turunan yang dapat dihasilkan dari kedua parent diatas adalah :
Anak 1 : 1 2 3 4 | E F G H
Anak 2 : A B C D | 5 6 7 8

2.5.3 Mutasi

SEBELUM MUTASI	1	1	0	1	0	0	0	0
SETELAH MUTASI	1	1	1	1	0	0	0	0

Gambar 2.5 Proses Mutasi [13]

Mutasi memungkinkan memunculkan individu-individu baru yang bukan berasal dari kawin silang. Pada tugas akhir ini, operator mutasi digunakan ketika hasil *crossover*

kedua induk memberikan hasil yang sama dengan salah satu induk, selain itu digunakan pula ketika hasil persilangan kedua induk tersebut merupakan angka nol. Banyak sekali teknik mutasi, salah satunya adalah *bit inversion* dimana gen yang dikenakan mutasi apabila bernilai 0 akan diubah menjadi 1 dan sebaliknya apabila bernilai 1 akan diubah menjadi 0. Gen tersebut juga dipilih secara acak. Sebagai contoh penerapan metode mutasi dapat diilustrasikan pada Gambar 2.5

2.5.4 Evaluasi

Evaluasi pada GA adalah menghitung *fitness value* dari suatu individu/kromosom. Terdapat bermacam-macam metode yang dapat digunakan untuk mencari nilai *fitness value* pada GA, tergantung dengan permasalahan yang akan diselesaikan. Pada tugas akhir kali ini, perhitungan nilai *fitness value* per individu didapatkan menggunakan nilai akurasi dari metode utama yaitu kombinasi KNN dan *Naïve Bayes*.

2.6 *K-Nearest Neighbor*

Algoritma KNN adalah algoritma *supervised learning* dimana label dari hasil yang diharapkan telah diketahui sebelumnya. Hasil yang diharapkan ini disebut *test class*. Hasil ini kemudian akan diuji dengan hasil prediksi yang didapatkan dari *train class K* tetangga terdekatnya. Dalam KNN untuk mengetahui kromosom mana yang merupakan tetangga terdekat, KNN bekerja berdasarkan jarak minimum dari data baru atau data *testing* ke data *training samples* untuk menentukan *K* tetangga yang paling dekat. Pada tugas akhir ini, pencarian jarak minimum menggunakan 3 rumus pehitungan jarak yaitu sebagai berikut :

a. *Euclidean Distance* [14]

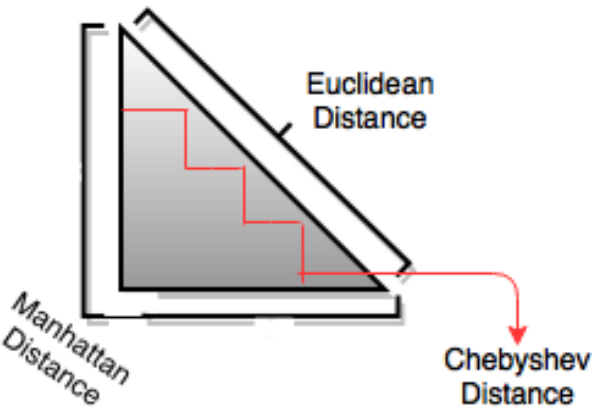
$$D_{xy} = \sqrt{\sum_{k=1}^{m} (x_{ik} - x_{jk})^2} \quad (2.2)$$

b. *Manhattan Distance* [15]

$$D_{xy} = \sum_{k=1}^{m} | x_{ik} - x_{jk} | \quad (2.3)$$

c. *Chebyshev Distance* [15]

$$D_{xy} = max_k | x_{ik} - x_{jk} | \quad (2.4)$$



Gambar 2.6 Ilustrasi metode perhitungan jarak

Jika diilustrasikan pada sebuah segitiga siku-siku, *Eucldean distance* merupakan perhitungan jarak yang paling

minimum, hanya saja tidak berlaku pada seluruh sistem seperti halnya pada sistem *google maps*, dimana terdapat pemukiman penduduk di rute *Euclidean Distance* yang menyebabkan tidak dapat terpilihnya rute ini sebagai saran mencapai tujuan. *Manhattan Distance* atau lebih dikenal dengan *City Block* merupakan rute terjauh pada ilustrasi tersebut, hanya saja jenis perhitungan ini dapat digunakan pada *google maps* walaupun bukan merupakan saran rute terdekat untuk mencapai tujuan. Sedangkan *Chebyshev Distance* merupakan kombinasi perhitungan antara *Euclidean Distance* dan *Manhattan Distance*, sehingga merupakan perhitungan jarak yang tepat pada kasus sistem *google maps*. Ilustrasi yang dimaksud dapat dilihat pada Gambar 2.6.

2.7 *Naïve Bayes*

Algoritma Naïve Bayes merupakan salah satu algoritma klasifikasi yang melakukan proses berdasarkan nilai perhitungan probabilitas. Dalam algoritma ini, diasumsikan bahwa semua atribut bersifat independen atau tidak saling ketergantungan sehingga ketidakseimbangan nilai data tidak menjadi masalah disini. Konsep dasar yang digunakan *Naïve Bayes* adalah teorema Bayes. Teorema Bayes adalah teorema yang digunakan dalam statistika untuk menghitung peluang setiap kelas dari data input yang diberikan dan menentukan kelas mana yang memiliki nilai probabilitas tertinggi. Untuk klasifikasi data kontinyu digunakan rumus Densitas Gauss seperti berikut [16]:

$$P(X_i = x_i | Y_j = y_j) = \frac{1}{\sqrt{2\pi\sigma_{ij}}} e^{\frac{(x_i - \mu_{ij})^2}{2\sigma^2 ij}} \quad (2.5)$$

2.8 *Confusion Matrix*

Confusion Matrix adalah suatu metode yang biasanya digunakan untuk melakukan perhitungan performa *output* pada konsep *data mining*. Salah satu *output* yang dihasilkan oleh metode ini adalah berupa hasil perhitungan *accuracy*. *Accuracy* adalah perbandingan kasus yang diidentifikasi benar dengan jumlah seluruh kasus yang ada. Penjelasan tersebut dapat dituliskan sebagai berikut[17] :

$$Accuracy\ (ACC) = \frac{\sum True\ positive\ +\ \sum True\ negative}{\sum Total\ population} \qquad (2.6)$$

2.9 *Bot Telegram*

Telegram merupakan sebuah aplikasi pesan instan multiplatform berbasis cloud yang berfokus pada kecepatan dan kemanan proses yang berlangsung. Telegram dirancang untuk memudahkan pengguna saling berkirim pesan teks, audio, video, gambar, file, dan lain sebagainya. Aplikasi ini diprakarsai oleh dua bersaudara asal Rusia yaitu Nikolai Durov dan Pavel Durov pada tahun 2013. Telegram juga menyediakan layanan *API* kepada pengembang independen, salah satunya *bot*. *Bot* sering juga disebut sebagai robot internet karena fungsinya yang dapat melakukan berbagai pekerjaan secara otomatis sesuai keinginan pengembang. *Bot* dapat digunakan untuk mencari di mesin pencari, cuaca, torrent, nilai tukar, alih bahasa, bahkan program yang sangat rumit sekalipun. Pada tugas akhir ini *Bot* Telegram digunakan untuk memberikan informasi secara otomatis ketika terjadi kerusakan pada BTS.

BAB III
PERANCANGAN PERANGKAT LUNAK

Pada bab ini akan dijelaskan mengenai perancangan sistem perangkat lunak yang akan dibuat. Perancangan akan dibagi menjadi tiga proses utama, yaitu:

1. *Preprocessing* untuk mensolusikan ketidakseimbangan nilai data dengan menggunakan metode *min-max normalization.*

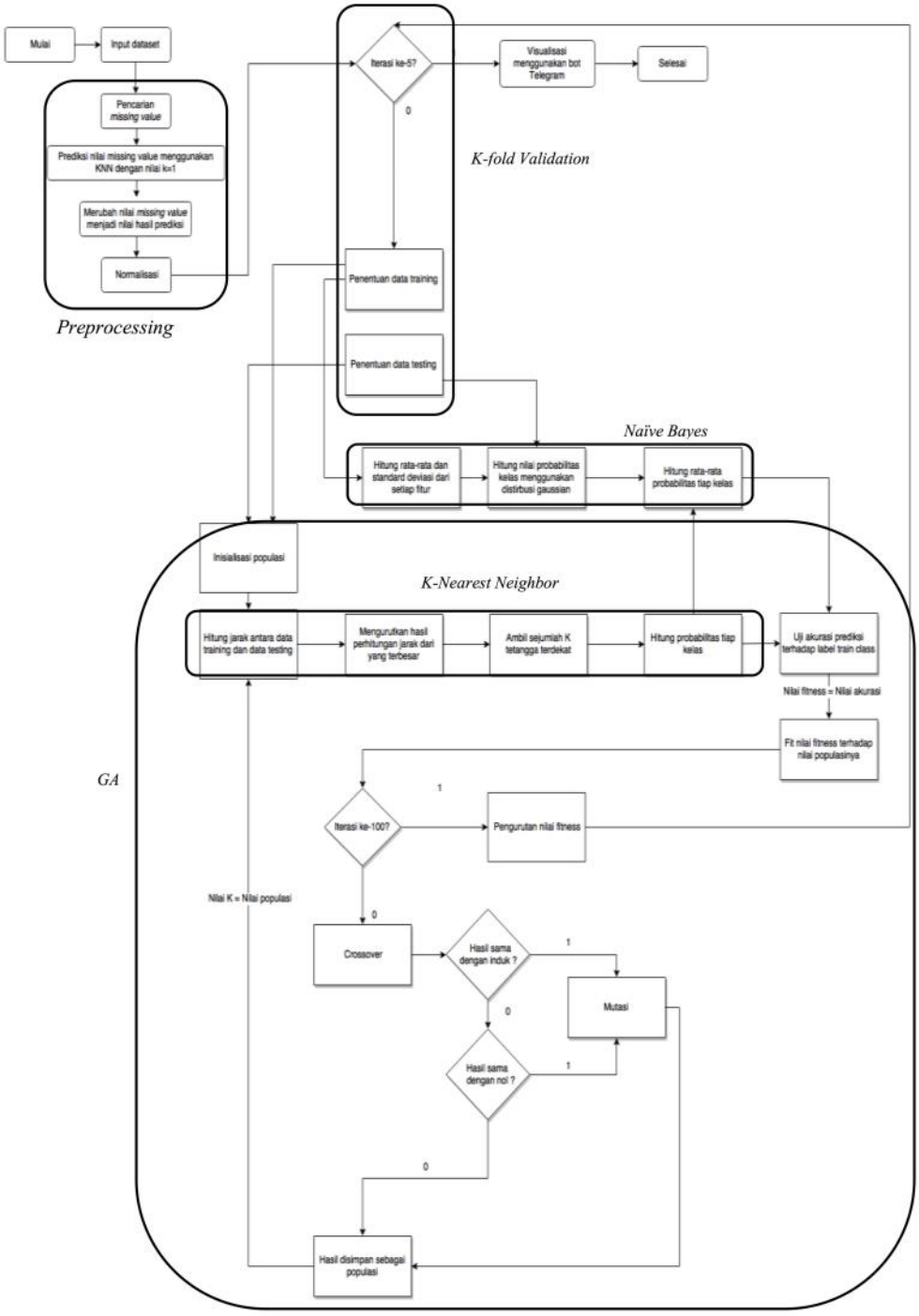
2. Pengimplementasian GA untuk mendukung algoritma KNN serta implementasi algortima *Naive Bayes* untuk meningkatkan nilai performa *output* dimana didapatkan melalui implementasi dari perhitungan akurasi menggunakan *Confusion Matrix* yang diikuti dengan pengukuran performa dataset menggunakan algoritma *K-Fold Validation*.

3. Penerapan visualisasi menggunakan bot Telegram.
   Pada bab ini akan dijelaskan gambaran umum program utama dalam *flowchart* serta rancangan setiap metode dalam bentuk *pseudocode*.

3.1 Desain Metode Secara Umum

Pada Tugas Akhir ini, diagram alur kerja sistem secara umum akan disajikan dalam bentuk flowchart yang dapat dilihat pada Gambar 3.1.

[Halaman ini sengaja dikosongkan]



Gambar 3.1 Ilustrasi alur program

[Halaman ini sengaja dikosongkan]

Metode ini memiliki dua tahap inti yaitu tahap membangun model terbaik menggunakan kombinasi algoritma KNN dan algoritma *Naïve Bayes*, dimana kedua tahap inti tersebut didukung oleh beberapa metode lainnya seperti salah satunya GA. Dimulai dari permasalahan dataset berupa tidak seimbangnya nilai data yang dapat menyebabkan berkurangnya kualitas output yang dihasilkan. Permasalahan ini dapat disolusikan menggunakan tahap normalisasi dimana pada tugas akhir ini menggunakan metode *Min-Max Normalization.* Sehingga data akan diubah ke dalam batasan 0-1. Kemudian selanjutnya hasil dari tahap tersebut dikelompokkan menjadi beberapa bagian data untuk mendapatkan kumpulan data yang menjadi *training* dan *testing* dengan performa terbaik menggunakan algoritma *K-Fold Validation*. Selanjutnya adalah tahap inti yaitu kombinasi antara KNN dan *Naïve Bayes* dalam melakukan klasifikasi sesuai kelas atau prediksi kelas pada setiap data *testing*. Sebelumnya GA digunakan untuk membantu KNN dalam menemukan nilai *K* yang paling optimal. Tahap terakhir adalah perhitungan akurasi dari setiap kelompok data menggunaka metode akurasi pada *confusion matrix* yang kemudian jika akurasinya tinggi maka akan divisualisasikan menggunakan bot Telegram.

3.1.1 Preprocessing

Dalam mengawali proses pengerjaan program, maka harus disiapkan terlebih dahulu data yang akan digunakan dalam proses pengerjaan. *Preprocessing* merupakan tahapan yang penting untuk dilakukan karena salah satu permasalahan yang sering kali terdapat pada suatu dataset ialah bahwa tidak seimbangnya nilai pada dataset tersebut. Maka itu pada tugas akhir kali ini, tahap *preprocessing* yang dilakukan adalah

menyeimbangkan nilai data menggunakan metode *min-max normalization*.

3.1.1.1 Normalisasi

Tahap normalisasi yang diterapkan adalah dengan menggunakan algoritma *Min-Max* sebagaimana perhitungannya sudah dijelaskan pada persamaan di bab sebelumnya. Tahap ini bertujuan untuk menyeimbangkan nilai data menjadi batasan 0-1. *Pseudocode Min-Max Normalization* dapat dilihat pada Gambar 3.2.

Masukan	Data tanpa batasan tertentu
Keluaran	Data dengan batasan 0-1

```
1.  data_atribut;
2.  for i = 1 to n column of data_atribut
3.      find minimal and maximal value
3.      for j = 1 to n row of data_atribut
4.          calculate new data_atribut[i][j] using
            min-max normalization
5.      end for
6.  end for
```

Gambar 3 2 Pseudocode Fungsi Normalisasi

3.1.2 Processing

Pada bagian ini dijelaskan pseudocode dari algoritma yang digunakan yaitu *K-fold Validation,* GA, KNN, dan *Naïve Bayes*. Penerapan keseluruhan algoritma ini menggunakan bahasa pemrograman *Python 2.7*.

3.1.2.1 K-Fold Validation

Fungsi *K-Fold Validation* digunakan untuk menentukan kelompok data *training* dan *testing* yang memiliki performa

terbaik. Dalam fungsi ini data akan dibagi menjadi 5 kelompok sama besar yang selanjutnya salah satu kelompok data sebagai data *testing* dan sisanya sebagai data *training*. Begitu seterusnya hingga seluruh kelompok data sempat menjadi data *testing* dan *training*. *Pseudocode K-Fold Validation* dapat dilihat pada Gambar 3.3.

Masukan	Matriks data keseluruhan, tipe kelas, jumlah data per kelas, dan iterasi fold
Keluaran	Data *training*, *testing* dan batasan tiap fold

```
1.  data_changed = all_data;
2.  for i = 1 to n fold
3.      for j = 1 to n class
4.          append test(i) by 20% data from
            data_changed(j)
5.          data_changed(j) = data_changed(j) – test
6.          for x = 1 to n instances of real_data(j)
7.              if instances(x) not in test(i) :
8.                  append train(i) by
                    instances(x)
9                   end if
10.         end for
11.     end for
12. end for
```

Gambar 3 3 Pseudocode K-Fold Validation

3.1.2.2 Genetic Algorithm

Pada bagian ini dijelaskan pseudocode dari metode GA. Metode GA terdiri dari 5 tahapan yaitu tahap *generate kromosom* yang melewati tahap *encoding*, *crossover*, *mutasi*, dan tahap evaluasi *fitness value* menggunakan algoritma KNN. Pada tugas akhir ini tahapan-tahapan tersebut dikelompokkan

menjadi hanya 2 fungsi yaitu fungsi inisialisasi dan fungsi utama.

3.1.2.2.1 Fungsi inisialisasi

Fungsi ini mencakup tahap *generate kromosom* sebagai tahap pertama dalam algoritma genetika dimana dalam proses representasinya menggunakan dukungan tahap *encoding* sebagaimana yang telah dijelaskan pada bab sebelumnya. Kromosom direpresentasikan dalam bilangan biner sebanyak 8 bit. Pada fungsi ini akan di *generate* 8 buah kromosom awal yang selanjutnya disebut sebagai *parent*. Sebelum diubah dalam bentuk bit biner, 8 buah kromosom ini merupakan bilangan desimal yang didapatkan dari hasil random antara 1 hingga batas maksimal parameter nilai *K* yaitu 25% dari jumlah total data. Nilai batas maksimal inilah yang akan menjadi input pada fungsi ini. Pseudocode fungsi inisialisasi dapat dilihat pada Gambar 3.4.

Masukan	Batas maksimal parameter nilai K
Keluaran	Representasi kromosom

```
1.  max; array of K;
2.  for i = 1 to n parents
3.      temp = get a random value from 1 to max
4.      append temp to array of K
5.  end for
```

Gambar 3 4 Pseudocode Fungsi Inisialisasi

3.1.2.2.2 Fungsi utama Genetic Algorithm

Fungsi ini mencakup seluruh tahapan algoritma genetika yang belum diproses pada fungsi inisialisasi. Tahap awal yang dilakukan setelah fungsi inisialisasi berjalan adalah tahap evaluasi dimana pada tahap ini dilakukan perhitungan nilai *fitness value* setiap kromosom yang telah di *generate*. Di

dalam tahap inilah letak implementasi algoritma KNN dan *Naïve Bayes*. Data *testing* akan dihitung jaraknya (atau nilai *dissimilarity*) terhadap keseluruhan data *training*. Nilai probabilitas tiap kelas dari sejumlah *K* nilai jarak terkecil atau yang memiliki kemiripan paling besar akan menjadi *output* dari implementasi algoritma KNN. Jarak data *testing* dengan data *training* dihitung menggunakan beberapa rumus perhitungan jarak sebagaimana yang telah disebutkan pada bab sebelumnya. Kemudian setiap fitur atau atribut pada data *training* dihitung nilai *mean* dan standard deviasi nya, dimana kedua nilai ini akan menjadi parameter perhitungan probabilitas tiap kelas pada data *testing* yang diberikan. Hasil perhitungan probabilitas tersebut merupakan *output* dari implementasi algoritma *Naïve Bayes*. Selanjutnya hitung rata-rata probabilitas tiap kelas dari *output* algoritma KNN dan *ouput* algoritma *Naïve Bayes* tersebut. Kelas dengan nilai probabilitas tertinggi merupakan kelas prediksi untuk data *testing* terpilih. Kelompok *Output* yang diberikan diolah kembali menggunakan rumus perhitungan akurasi pada *confusion matrix* sebagaimana yang telah dijelaskan pada bab sebelumnya. Nilai ini yang kemudian menjadi *fitness value* dari kromosom tersebut. Setelah tahap evaluasi berjalan, dilakukan tahap *crossover* dan atau mutasi dalam pembentukan generasi selanjutnya. Tahap *crossover* adalah tahap yang digunakan untuk menghasilkan keturunan baru melalui prosedur kawin silang (*crossover*). Dalam fungsi ini, akan dipilih dua individu secara acak dari kumpulan *parent* yang berupa *output* dari fungsi 3.4. Tipe kawin silang yang digunakan adalah *single-point crossover* sebagaimana yang telah dijelaskan pada bab sebelumnya. Sedangkan tahap mutasi adalah tahap yang digunakan untuk menghasilkan keturunan baru melalui mutase gen pada individu. Tidak

semua proses melewati tahap ini. Mutasi hanya dilakukan apabila *output* dari proses *crossover* termasuk dalam 3 jenis kriteria antara lain menghasilkan nilai yang sama dengan *parent*, menghasilkan angka nol, atau hasil yang didapat diluar batas maksimal nilai *K* dimana telah dijelaskan pada fungsi 3.4. Tipe mutasi yang dilakukan adalah *bit inversion* sebagaimana yang telah dijelaskan pada bab sebelumnya. Pseudocode fungsi utama GA dapat dilihat pada Gambar 3.11.

3.1.2.3 *K-Nearest Neighbor* (KNN)

Masukan	Matriks data *training*, data *testing*, dan K
Keluaran	Prediksi kelas pada data *testing*

```
1.  training_data; testing_data; k;
2.  for i = 1 to n testing_data
3.      for j = 1 to n training_data
4.          compute distance between
            testing_data(i) with training data(j) using Euclidean
            Distance, Manhattan Distance and Chebyshev
            Distance
5.      end for
6.      find K smallest distance
7.      compute probability of each class by votes of K instances
8.  end for
```

Gambar 3 5 Pseudocode Fungsi *K-Nearest Neighbor*

Fungsi KNN adalah salah satu fungsi pada tugas akhir ini yang digunakan untuk melakukan perhitungan nilai probabilitas tiap kelas dari suatu data *testing* yang terpilih. Dalam fungsi ini akan dilakukan perhitungan jarak kemiripan antara data *testing* terhadap data *training* menggunakan beberapa persamaan yang telah didefinisikan pada bab sebelumnya. Kemudian hasil dari masing masing perhitungan jarak di pasangkan terhadap kelas aslinya yaitu kelas pada data

training yang sedang terlibat dalam perhitungan. Setelah tahap ini, maka terbentuklah matriks 2 dimensi. Matriks ini diurutkan secara *Ascending* atau dari jarak terkecil hingga terbesar guna mengetahui model data yang memiliki tingkat kemiripan tertinggi terhadap data yang sedang di *testing*. Selanjutnya diambil *K* data teratas dimana *K* adalah hasil output dari metode GA. Selanjutnya dihitung nilai probabilitas kemunculan tiap kelas pada *K* data tersebut. Kelas dengan probabilitas paling tinggi merupakan prediksi kelas dari data *testing* terpilih saat ini. Akan tetapi pada tahap inti tugas akhir ini cukup pada tahap menghitung nilai probabilitas tiap kelas sehingga tidak melakukan prediksi kelas pada data *testing*, karena nilai probabilitas tersebut selanjutnya akan dikombinasikan menggunakan metode *Naïve Bayes*. *Pseudocode* fungsi KNN [19] dapat dilihat pada Gambar 3.12.

3.1.2.4 Naïve Bayes

Pada bagian ini akan dijelaskan pseudocode dari metode *Naïve Bayes*. Metode ini selanjutnya akan dikombinasikan bersama algoritma KNN untuk mengklasifikasi suatu data *testing* pada kelas tertentu sesuai dengan hasil yang terprediksi. Pada tugas akhir ini, metode *Naïve Bayes* akan dibagi menjadi 2 fungsi yaitu fungsi untuk memproses data *testing* dan fungsi untuk memproses data *training*.

Masukan	Matriks data *training*, list tipe kelas dataset
Keluaran	Model data *training*

```
1.  training_atr;
2.  for i = 1 to n training_atr
3.      calculate main
4.      calculate standard deviation
5.  end for
```

Gambar 3 6 Pseudocode Fungsi *Gaussian Training*

3.1.2.4.1 Fungsi Gaussian Training

Dalam fungsi ini akan dicari model dari setiap atribut pada data *training* yaitu berupa nilai *mean* atau rata-rata dan nilai standard deviasi. Kedua nilai ini akan menjadi parameter perhitungan yang dibutuhkan untuk memproses suatu data *testing*. *Pseudocode* fungsi ini dapat dilihat pada Gambar 3.5.

3.1.2.4.2 Fungsi Gaussian Testing

Dalam fungsi ini data *testing* yang terpilih akan dihitung nilai probabilitasnya pada tiap kelas menggunakan rumus perhitungan *Densitas Gauss* seperti yang telah dijelaskan pada bab sebelumnya. Parameter dari rumus perhitungan ini ialah hasil *output* dari fungsi *Gaussian Training*. Kemudian nilai probabilitas masing-masing kelas tersebut diurutkan secara *Ascending* dan kelas yang memiliki probabilitas paling tinngi maka itulah yang merupakan prediksi dari kelas data *testing* tersebut. Namun pada tugas akhir ini, algoritma *naive bayes* hanya sampai pada menghitung nilai probabilitas tiap kelas, dimana selanjutnya nilai ini akan dikombinasikan dengan probabilitas tiap kelas yang diperoleh dari algoritma KNN sehingga diharapkan hasil prediksi yang dihasilkan menjadi lebih akurat. *Pseudocode* fungsi *Naïve Bayes* dapat dilihat pada Gambar 3.6.

Masukan	Matriks atribut data *testing*, list tipe kelas dataset
Keluaran	Probabilitas kelas pada data *testing*

```
1.  testing_atr; class_type;
2.  for i = 1 to n testing_atr
3.      for j = 1 to n class_type
4.          Do Densitas Gauss
5.      end for
6.  end for
```

Gambar 3 7 Pseudocode Fungsi *Gaussian Testing*

3.1.2.5 Akurasi

Akurasi adalah metode yang digunakan untuk menghitung besar keakuratan hasil prediksi kelas data *testing* terhadap kelas yang ada pada data *training*. Fungsi ini berguna dalam membantu GA untuk menentukan nilai *K* yang paling optimal yaitu dengan menjadi nilai *fitness* pada setiap kromosom yang terbentuk. *Pseudocode* fungsi akurasi dapat dilihat pada Gambar 3.7.

Masukan	Matriks kelas data *training*
Keluaran	Matriks kelas data *testing*

```
1.  testing_class; training_class;
2.  j = 0
3.  for i = 1 to n testing_class
4.      if testing_class(i) = training_class(j) :
5.          labeled as true positive
6.      else :
7.          labeled as true negative
8.      end if
9.      j++
10. end for
11. sum = sum of true positive and true negative
12. result = divide sum with total data
13. akurasi = result * 100
```

Gambar 3 8 Pseudocode Fungsi Akurasi

3.1.3 Program Utama (main)

Program utama atau sering disebut dengan fungsi *main* berisi tentang proses memanggil kembali fungsi-fungsi yang telah didefinisikan sebelumnya yaitu fungsi Normalisasi, *K-Fold Validation,* GA, KNN, *Naïve Bayes* dan fungsi Akurasi. Terdapat juga beberapa kode program pendukung yaitu

membaca serta menulis dataset dalam format text. Input dari program utama adalah dataset BTS dari salah satu perusahaan telekomunikasi di Indonesia. *Pseudocode* fungsi *main* dapat dilihat pada Gambar 3.8.

Masukan	Output dari semua fungsi
Keluaran	Data model dengan akurasi terbaik

```
1.  foldvalidation;
2.  Do read file
3.  for i = 1 to n foldvalidation :
4.      Do imputation method
5.      Do normalization data
6.      Do fold validation process
7.      Do GA, KNN and Naive Bayes function
8.  end for
9.  Write file in txt
```

Gambar 3 9 Pseudocode Fungsi Utama

3.2 Visualisasi

Masukan	-
Keluaran	IP

```
1.  Define rules of bot
2.  Listen IP
3.  Write IP in txt file
```

Gambar 3 10 Pseudocode mendapatkan IP

Masukan	IP, Data *Training*, Data *Testing*
Keluaran	Notifikasi gangguan kepada IP yang terdaftar

```
1.  testing_data; training_data;
2.  Do read file
3.  Do imputation method
4.  Do normalization data
5.  Do GA,KNN, and Naive Bayes using sklearn
6.  Send notifications
```

Gambar 3 11 Pseudocode eksekusi bot

Penerapan keseluruhan algoritma adalah menggunakan bahasa pemrograman *Python 2.7*. Begitu pula terhadap penerapan proses visualisasi. Pada tugas akhir ini akan diintegrasikan *Python* dengan fitur *Bot Telegram*. Fitur bot telegram pada tugas akhir ini berfungsi untuk mengirimkan pesan gangguan langsung kepada pihak yang bertugas dalam melakukan perbaikan pada gangguan tersebut, sehingga dapat menghindari adanya salah paham dan keterlambatan penanganan gangguan. Pada tugas akhir ini, proses visualisasi akan dibagi menjadi 2 program yaitu program untuk mendapatkan IP akun telegram pihak yang bersangkutan dan program untuk mengeksekusi semua metode serta bot Telegram. *Pseudocode* kedua program tersebut dapat dilihat pada Gambar 3.9. dan 3.10.

Masukan	Data *training*, kromosom, data *testing*
Keluaran	Dua individu baru

```
1.  testing_data; training_data; K; mom; dad;
2.  child 1 = 4 first bit from mom + 4 last bit from dad
3.  child 2 = 4 last bit from mom + 4 first bit from mom
4.  if child = parent or child out of range or child = 0 :
5.      random index to get bit that will be mutated
6.      child = mutation result
7.  end if
8.  for i = 1 to n testing_data
9.      for j = 1 to n training_data
10.         Do K-Nearest Neighbor Algorithm with K value =
            child 1 and K value = child 2
11.     end for
12.     Do Accuration Function
13.     Fitness value(i) = accuracy result
14. end for
```

Gambar 3 12 Pseudocode Fungsi Utama Genetic Algorithm

[Halaman ini sengaja dikosongkan]

BAB IV
IMPLEMENTASI

Pada bab ini akan dibahas mengenai implementasi yang dilakukan berdasarkan rancangan yang telah dijabarkan pada bab sebelumnya. Implementasi kode program dilakukan sepenuhnya menggunakan bahasa *Python*. Sebelum penjelasan implementasi akan ditunjukkan terlebih dahulu lingkungan untuk melakukan implementasi.

4.1 Lingkungan implementasi

Lingkungan implementasi yang akan digunakan untuk melakukan implementasi adalah *Spyder* yang diinstal pada sistem operasi MacOS X El Capitan.

4.2 Implementasi

Pada sub bab ini akan dijelaskan parameter yang digunakan dan implementasi setiap sub bab yang terdapat pada bab sebelumnya yaitu bab perancangan program. Pada bagian implementasi ini juga akan dijelaskan mengenai fungsi-fungsi yang digunakan dalam program tugas akhir ini dan disertai dengan kode sumber masing-masing fungsi utama.

4.2.1 Parameter yang Digunakan

Pada tugas akhir ini, penulis akan memilih 1 individu terbaik berdasarkan nilai *fitness* dari setiap *fold*. Selain itu, penulis juga menggunakan tipe kawin silang *single-point crossover* dan tipe mutase *bit-inversion*. Parameter yang digunakan dapat dilihat pada Tabel 4.1.

Tabel 4.1 Parameter yang digunakan

Parameter GA	Nilai	Parameter K-NN	Nilai	Parameter K-fold validation	Nilai
Jumlah populasi	100	Jumlah tetangga terdekat (library)	1	Jumlah kelompok data	5
Jumlah kromosom	4				
Jumlah gen (bit-string)	8	Jumlah tetangga terdekat (stratch)	GA		
Individu yang dipilih (elitism)	1				

Dalam menggunakan metode GA, dibutuhkan beberapa parameter yang harus ditentukan diawal program, yaitu :

1. Jumlah populasi, yaitu banyak generasi yang akan terbentuk selama proses GA. Dalam satu generasi terdiri dari 2 *parents* dan 2 *child*. Pada Tugas Akhir ini, jumlah populasi yang akan terbentuk ialah 100 generasi.
2. Jumlah kromosom, yaitu banyaknya kromosom atau individu dalam satu generasi. Pada Tugas Akhir ini, jumlah kromosom yang akan terbentuk ialah 4 individu.
3. Jumlah gen, yaitu panjang bit dari satu kromosom/individu. Pada Tugas Akhir ini jumlah gen yang ditentukan adalah 8 *bit-string*.
4. Individu yang dipilih, yaitu jumlah individu yang ditentukan akan bertahan sampai pada akhir generasi GA. Pada Tugas Akhir ini individu yang dipilih adalah 1.

Sedangkan dalam menggunakan metode KNN dan *K-Fold Validation* hanya membutuhkan 1 parameter yang harus ditentukan diawal program yaitu besar nilai *K*. Untuk KNN yang menggunakan library, ditentukan nilai *K*=1 dan untuk KNN tanpa menggunakan library, ditentukan nilai *K* oleh output dari metode GA. Kemudian untuk *K* pada *K-Fold Validation* ditentukan nilai *K*=5.

4.2.2 Implementasi Fungsi Readfile

Fungsi readfile() adalah fungsi yang dibuat untuk membaca suatu file dataset berformat txt. Input dari fungsi ini berupa nama file yang dimaskud dan *ouptut* dari fungsi ini berupa isi dari dataset yang sudah siap digunakan.

```
1.  with open(namafile) as f:
2.    for index,line in enumerate(f):
3.       kelas = line.strip("\n").split(",")[-1]
4.       perline = line.strip().split(",")
5.       data.append(perline)
6.  return [data,tipekelas,asd]
```

Kode Sumber 4 1 Implementasi Fungsi Readfile()

Kode Sumber 4.1 berfungsi untuk membaca isi file txt, yang kemudian direpresentasikan ke dalam variabel bertipe array. Berdasarkan implementasi kode diatas, dari suatu dataset akan didapatkan data, tipe kelas, dan total jumlah data pada tiap kelas.

4.2.4 Implementasi Metode Normalisasi

Metode normalisasi adalah metode yang digunakan untuk menstabilkan nilai pada suatu dataset ke dalam range tertentu sesuai rumus perhitungan yang digunakan.

```
1.  def run(data):
2.      data=np.array(data)
3.      data_atr=data[:,:-1]
4.      data_atr=data_atr.astype(float)
5.      data_cls=data[:,-1]
6.      data_cls=data_cls.tolist()
7.      for n in range(len(data_atr[1])):
8.          minimal = data_atr[:,n].min()
9.          data_atr[:,n]=data_atr[:,n]-minimal
10.         maksimal = data_atr[:,n].max()
11.         data_atr[:,n]=data_atr[:,n]/(maksimal-
    minimal)
12.     data_atr=data_atr.astype('str')
13.     data_atr=data_atr.tolist()
14.     for n in range(len(data_atr)):
15.         data_atr[n].append(data_cls[n])
16.     data=data_atr
17.     return data
```

Kode Sumber 4 2 Implementasi Metode Normalisasi

Pada kode sumber 4.2, diimplementasikan rumus perhitungan *Min-Max* seperti yang telah dijelaskan pada bab sebelumnya, sehingga data akan berada dalam range 0-1.

4.2.5 Implementasi Metode K-Fold Validation

Metode *K-Fold Validation* digunakan untuk menemukan model kelompok data terbaik sebagai *training* dan *testing*. Pada tugas akhir ini digunakan sebanyak 5 fold validation. Ilustrasi metode ini telah dijelaskan pada bab sebelumnya.

Kode Sumber 4.3 berfungsi untuk mendapatkan *list* kelas apa saja yang terkandung pada dataset, serta mendapatkan *list* batas pemotongan data pada setiap *fold* dari kelas tersebut.

```
1.  for line in data:
2.      perline = line
3.      kelas = line[-1]
4.      for j in range((100/20)+1):
5.          potong = 0.2*jumlahDataPerKelas[tipekelas.in
    dex(kelas)]
6.          x = round(potong*j)
7.          vold.append(int(x))
8.       if kelas not in temp:
9.          temp.append(kelas)
10.         wida = [kelas, vold]
11.         volds.append(wida)
```

Kode Sumber 4 3 Implementasi Metode K-Fold Validation (1)

```
1.  if(v==0):
2.   try:
3.     batasbawah = volds[tipekelas.index(kelas)][1][v]
       batasatas = volds[tipekelas.index(kelas)][1][v+1]
     except:
4.     batasbawah = volds[0][1][v]
5.     batasatas = volds[0][1][v+1]
6.   if counter[tipekelas.index(kelas)] < batasatas:
7.     test.append(perline)
8.     counter[tipekelas.index(kelas)] = counter[tipekel
    as.index(kelas)] + 1
9.   else:
10.    train.append(perline)
```

Kode Sumber 4 4 Implementasi Metode K-Fold Validation (2)

Kode Sumber 4.4 berfungsi untuk mendapatkan kumpulan data *testing* dan data *training* pada *fold* pertama sesuai dengan batasan yang telah didapatkan dari Kode Sumber 4.3.

Kode Sumber 4.5 berfungsi untuk mendapatkan kumpulan data *testing* dan data *training* pada *fold* terakhir sesuai dengan batasan yang telah didapatkan dari Kode Sumber 4.3.

```
1.  elif(v==len(vold)-1):
2.          batasbawah = volds[tipekelas.index(kelas)][1]
            [v-1]
3.          batasatas = volds[tipekelas.index(kelas)][1][
            v]
4.  if counter[tipekelas.index(kelas)]<batasatas and cou
    nter[tipekelas.index(kelas)]>=batasbawah:
5.          test.append(perline)
6.          counter[tipekelas.index(kelas)] = counter[tip
            ekelas.index(kelas)] + 1
7.  else:
8.          train.append(perline)
9.          counter[tipekelas.index(kelas)] = counter[tip
            ekelas.index(kelas)] + 1
```

Kode Sumber 4 5 Implementasi Metode K-Fold Validation (3)

```
1.  batasatas = volds[tipekelas.index(kelas)][1][v]
2.  batasbawah = volds[tipekelas.index(kelas)][1][v+1]
3.  if counter[tipekelas.index(kelas)]>=batasatas and co
    unter[tipekelas.index(kelas)]<batasbawah:
4.     test.append(perline)
5.     counter[tipekelas.index(kelas)] = counter[tipekel
    as.index(kelas)] + 1
6.  else:
7.     train.append(perline)
8.     counter[tipekelas.index(kelas)] = counter[tipekel
    as.index(kelas)] + 1
9.  train = np.array(train)
10. test = np.array(test)
11. return [test,train,tests,trains,volds,temp]
```

Kode Sumber 4 6 Implementasi Metode K-Fold Validation (4)

Kode Sumber 4.6 berfungsi untuk mendapatkan kumpulan data *testing* dan data *training* pada tiap *fold* terkecuali *fold* pertama dan terakhir sesuai dengan batasan yang telah didapatkan dari Kode Sumber 4.3.

4.2.6 Implementasi Metode Genetic Algorithm

GA digunakan untuk membantu KNN dalam menemukan nilai *K* yang paling optimal sehingga tidak dilakukan input manual seperti KNN yang murni. Terdapat 2 fungsi pada implementasi metode ini yaitu init_parents() dan run() yang akan dijelaskan kemudian.

```
1.  def init_parents(seperempat):
2.      random.seed(9001)
3.      ortu = random.randint(1,seperempat)
4.      for i in range(8):
5.          while(ortu in k):
6.              ortu = random.randint(1,seperempat)
7.          k.append(ortu)
```

Kode Sumber 4 7 Implementasi Metode Genetic Algorithm (1)

Kode Sumber 4.7 berfungsi untuk mendapatkan 8 nilai inisialisasi parents yang dibutuhkan oleh GA. Nilai tersebut ditentukan untuk berada dalam kisaran 1 sampai 20% dari total data yang ada.

```
1.  def run(k, train, test, seperempat, bapak, ibu, fitn
    ess_e, fitness_m, fitness_c, tipeKelas):
2.      random.seed(9001)
3.      test_class = test[:,-1]
4.      rand=random.randint(1,len(k)-1)
5.      dad=k[rand]
```

Kode Sumber 4 8 Implementasi Metode Genetic Algorithm (2)

Kode Sumber 4.8 berfungsi untuk memilih satu nilai dari hasil *output* Kode Sumber 4.7 untuk kemudian dijadikan sebagai nilai dari *parents* pertama. Tidak ada metode tertentu

yang digunakan dalam pemilihan nilai tersebut, melainkan hanya menggunakan bantuan variable *random*. Akan dilakukan proses yang sama untuk mendapatkan nilai dari *parents* kedua.

```
1.  #euclidean
2.  prediksikelas=KNN.run(dad,train,test,tipeKelas)
3.  fitnessdad = evaluasi.akurasi(prediksikelas, test_cl
    ass)
4.  fitness_e.append([dad,fitnessdad])
5.
6.  #manhattan
7.  prediksikelas=KNN.run(dad,train,test,tipeKelas,jarak
    ="manhattan",returnvalue="GA")
8.  fitnessdad = evaluasi.akurasi(prediksikelas, test_cl
    ass)
9.  fitness_m.append([dad,fitnessdad])
10.
11. #chebyshev
12. prediksikelas=KNN.run(dad,train,test,tipeKelas,jarak
    ="chebyshev",returnvalue="GA")
13. fitnessdad = evaluasi.akurasi(prediksikelas, test_cl
    ass)
14. fitness_c.append([dad,fitnessdad])
```

Kode Sumber 4 9 Implementasi Metode Genetic Algorithm (3)

Kode Sumber 4.9 berfungsi untuk mendapatkan nilai *fitness* dari hasil output Kode Sumber 4.8. Nilai ini yang kemudian digunakan sebagai parameter pemilihan individu yang akan bertahan. Untuk mendapatkan nilai *fitness*, dibutuhkan bantuan KNN dan Akurasi. Metode perhitungan jarak yang digunakan pada KNN ialah *Euclidean distance*, *Manhattan*, dan *Chebyshev distance*. Ketiga jenis metode perhitungan jarak ini akan dibandingkan hasilnya melalui nilai *fitness* tertinggi yang didapatkan oleh setiap metode pada akhir iterasi.

```
1.      dadbin = bin(dad)[2:]
2.      mombin = bin(mom)[2:]
3.      if(len(dadbin)%2==1):
4.          dadbin=dadbin.zfill(len(dadbin)+1)
5.      if(len(mombin)%2==1):
6.          mombin=mombin.zfill(len(mombin)+1)
7.
8.      if(len(dadbin)>len(mombin)):
9.          mombin=mombin.zfill(len(dadbin))
10.     else:
11.         dadbin=dadbin.zfill(len(mombin))
```

Kode Sumber 4 10 Implementasi Metode Genetic Algorithm (4)

Kode Sumber 4.10 berfungsi untuk merubah 2 nilai *parents* terpilih menjadi tipe *binary string*. Pada tahap selanjutnya yaitu *crossover*, kedua nilai parents ini akan saling mengalami kawin silang dimana membutuhkan kesamaan ukuran *array* pada keduanya. Hal tersebut diimplementasikan pula pada kode sumber ini.

```
1.  child1=dadbin[0:len(dadbin)/2]+mombin[len(mombin)/2]
2.  child2=mombin[0:len(mombin)/2]+dadbin[len(dadbin)/2]
3.  while(child1==dadbin or child1==mombin or int(child1
    ,2)==0):
4.      rand=random.randint(0,len(child1)-1)
5.      if(child1[rand]=='0'):
6.          child1=child1[0:rand]+'1'+child1[rand+1:]
7.      else:
8.          child1=child1[0:rand]+'0'+child1[rand+1:]
```

Kode Sumber 4 11 Implementasi Metode Genetic Algorithm (5)

Kode Sumber 4.11 merupakan implementasi dari proses *crossover* atau kawin silang pada GA, dimana pada tugas akhir ini menggunakan metode *single-point crossover*. Cara kerja

metode ini telah dijelaskan pada bab sebelumnya. Pada kode sumber ini diimplementasikan pula proses mutase pada GA. Proses ini akan dilakukan ketika hasil *crossover* dari kedua *parents* menghasilkan *binary* yang sama dengan parents tersebut dan atau menghasilkan *binary* yang merepresentasikan angka nol.

```
1.   if(child1 > seperempat):
2.           child1=child1%seperempat
3.       if(child2 > seperempat):
4.           child2=child2%seperempat
5.       child.append(child1)
6.       child.append(child2)
7.
8.       k.append(child1)
9.       k.append(child2)
10.
11.      return [k,bapak,ibu,fitness_e,fitness_m,fitness_
    c,train,test]
```

Kode Sumber 4 12 Implementasi Metode Genetic Algorithm (6)

Kode Sumber 4.12 merupakan implementasi proses mutase lainnya. Proses mutase, tidak hanya dilakukan ketika child menghasilkan binary yang sama terhadap parents melainkan juga dilakukan ketika nilai output dari crossover menghasilkan bilangan decimal diluar dari batasan yang ditentukan yaitu 1-20% dari dataset. Metode yang dilakukan pada tahap mutase ini adalah bit inversion. Cara kerja metode ini telah dijelaskan pada bab sebelumnya.

4.2.7 Implementasi Metode K-Nearest Neighbor

KNN digunakan untuk mendukung proses yang terjadi pada GA. Dukungan yang diberikan berupa pemberian nilai *fitness* dari kromosom baru yang dihasilkan. Terdapat 3

perhitungan jarak yang digunakan pada metode ini antara lain *Euclidean Distance, Manhattan Distance,* dan *Chebyshev Distance.*

```
datatest = float(test_atribut[baris_test][kolom])
datatrain = float(train_atribut[baris_train][kolom])
dikurangin = np.subtract(datatest, datatrain)
dikuadratin = np.power(dikurangin,2)
dimutlak = np.absolute(dikurangin)
mutlak.append(dimutlak)
kuadrat.append(dikuadratin)
if(jarak=="manhattan"):
   sums = sum(mutlak)
   manhattan.append([sums,train_class[baris_train][0]])
elif(jarak=="chebyshev"):
     maximum = sorted(mutlak)
     maxdistance.append([maximum,train_class[baris_train][0]])
else:
     sums = sum(kuadrat)
     e_distance_pertest=np.sqrt(sums)
     e_distance.append([e_distance_pertest,train_class[baris_train]])
```

Kode Sumber 4 13 Implementasi Metode K-Nearest Neighbor (1)

Kode Sumber 4.13 merupakan implementasi dari perhitungan 3 metode jarak tersebut diatas. Rumus perhitungan ketiga metode ini telah dijelaskan pada bab sebelumnya.

Kode Sumber 4.14 merupakan implementasi proses pada KNN setelah melakukan perhitungan jarak, yaitu mengurutkan seluruh hasil perhitungan jarak antara satu data testing terhadap seluruh data training secara *Ascending*. Kemudian diambil sebanyak *K* data teratas dan dilakukan voting pada

nilai kelasnya. Kelas dengan voting terbanyak merupakan kelas prediksi untuk data testing yang sedang diujikan tersebut. Secara keseluruhan pada tugas akhir ini, implementasi metode KNN memiliki 2 kriteria *output* yaitu berupa nilai probabilitas setiap kelasnya pada data testing atau nilai prediksi kelas pada data testing tersebut. KNN akan memberikan *output* berupa probabilitas ketika ia digunakan pada proses selain GA, sedangkan pada GA ia akan memberikan output berupa nilai prediksi kelas.

```
1.  if(jarak=="manhattan"):
2.     sort = sorted(manhattan)
3.     paratetangga = list(sort[0:k])
4.  elif(jarak=="chebyshev"):
5.     sort = sorted(maxdistance)
6.     paratetangga = list(sort[0:k])
7.  else:
8.     sort = sorted(e_distance)
9.     paratetangga = list(sort[0:k])
10. for y in range(len(tipekelas)):
11.    jumlah=sum(x.count(tipekelas[y]) for x in paratet
    angga)
12.    persen=(jumlah/float(k))*100.0
13.    var2=[tipekelas[y],persen/100.0]
14.    var=[tipekelas[y],persen]
15.    persenK.append(var)
16.    prob.append(var2)
17.    sortpersen=sorted(persenK, key=lambda persenK:per
    senK[1], reverse=True)
18.    probabilitas.append(prob)
19.    prediksikelas.append(sortpersen[0][0])
20.    prediksikelas=np.array(prediksikelas)
21.    prediksikelas=prediksikelas.astype(int)
22.    if(returnvalue!="GA"):
23.         return probabilitas
24.    else:
25.         return prediksikelas
```

Kode Sumber 4 14 Implementasi Metode K-Nearest Neighbor (2)

```
1.  def lib(k,train,test,tipeKelas,jarak="euclidean",ret
    urnvalue="GA"):
2.      klasifikasi = knn(n_neighbors=k,metric=jarak)
3.      trained_model = klasifikasi.fit(train_atribut,tr
    ain_class)
4.      probabilitas_knn = trained_model.predict_proba(t
    est_atribut)
5.     prediksikelas = klasifikasi.predict(test_atribut)
6.     prediksikelas = np.array(prediksikelas)
7.     prediksikelas = prediksikelas.astype(int)
8.     if(returnvalue!= "GA"):
9.         return probabilitas_knn
10.    else:
11.        return prediksikelas
```

Kode Sumber 4 15 Implementasi Metode K-Nearest Neighbor (3)

Kode Sumber 4.15 merupakan implementasi metode KNN menggunakan *library* dari sklearn. Terdapat 2 tipe output pada kode sumber ini, kriteria *output* yang diberikan sama seperti Kode Sumber 4.14.

4.2.8 Implementasi Metode *Naive Bayes*

Naive Bayes merupakan salah satu metode dalam bidang klasifikasi. Pada tugas akhir ini, penulis akan mencoba untuk menggabungkan metode *Naive Bayes* dengan metode KNN. Terdapat 2 fungsi dalam pengimplementasian metode ini yaitu fungsi gaussian_train() untuk mendapatkan model dari data *training* dan fungsi gaussian_test() untuk mendapatkan nilai probabilitas masing-masing kelas dari suatu data *testing* dengan melakukan perhitungan *Densitas Gauss* didalamnya.

Kode Sumber 4.16 merupakan implementasi fungsi gaussian_train() yaitu berupa proses mendapatkan nilai model untuk setiap fitur pada data training. Nilai model yang

dimaksud berupa rata-rata dan standard deviasi. Nilai ini akan digunakan untuk perhitungan pada tahap selanjutnya. Karena proses ini dilakukan pada setiap kelompok kelas, maka dideteksi beberapa kasus yang memungkinkan untuk terjadi *erorr* pada proses penentuan nilai model, yaitu ketika dataset hanya memiliki 1 data pada kelas tertentu. Ketika data tersebut menjadi data training, rata-rata pada setiap fiturnya merupakan dirinya sendiri sedangkan nilai standard deviasinya diberi nilai default 0. Namun ketika data tersebut menjadi data testing, rata-rata pada setiap fitur dan standard deviasinya diberi nilai default 0.

```
1.  prob=float(idx)/float(train.shape[0])
2.  if(train_atributs.shape[0]==1):
3.     deviasi=np.zeros(train_atributs.shape[1]-1)
4.     rata2=train_atributs[0][:-1]
5.  elif(train_atributs.shape[0]==0):
6.     deviasi=np.zeros(trains.shape[1])
7.     rata2=np.zeros(trains.shape[1])
8.  else:
9.     rata2=np.mean(train_atributs[:,:-1],axis=0)
10.    deviasi=np.std(train_atributs[:,:-1],axis=0)
11. perkolom=zip(rata2,deviasi)
12. perkelas.append([z,perkolom,prob])
13. return perkelas
```

Kode Sumber 4 16 Implementasi Metode Naive Bayes (1)

Kode Sumber 4.17 merupakan implementasi fungsi gaussian_test() yaitu berupa perhitungan *Densitas Gauss* atau perhitungan distribusi normal pada setiap fitur. Rumus perhitungan ini telah dijelaskan pada bab sebelumnya. Cara mendapatkan nilai probabilitas kelas yang sedang diproses adalah dengan melakukan perkalian pada seluruh hasil perhitungan *densitas gauss* yang dimiliki oleh setiap fitur.

```
1.  dikurangin=float(test_atributs[baris][kolom])-mean
2.  dikuadratin=np.power(dikurangin,2)
3.  if(dev==0):
4.     hasilrumus=0
5.  else:
6.     penyebutpangkat=2*(np.power(dev,2))
7.     pangkat=(dikuadratin/penyebutpangkat)*(-1)
8.     exp=np.exp(pangkat)
9.     akar=np.sqrt((2*3.14))
10.    penyebut=akar*dev
11.    hasilrumus=exp/penyebut
12. probabilitas=probabilitas*hasilrumus
13. p_perkelas=float(probabilitas*perkelas[kelas][2])
```

Kode Sumber 4 17 Implementasi Metode Naive Bayes (2)

```
1.  if(math.isnan(p_perkelas)):
2.     tmp.append([kelas,0])
3.     probabilitas_perkelas.append(0)
4.  else:
5.     tmp.append([kelas,p_perkelas])
6.     probabilitas_perkelas.append(p_perkelas)
7.     probabilitas_perbaris.append(tmp)
8.     prediksi.append(tipekelas[probabilitas_perkelas.a
    rgmax()])
9.     if(execute!="bayes"):
10.         return probabilitas_perbaris
11.    else:
12.         return prediksi
```

Kode Sumber 4 18 Implementasi Metode Naive Bayes (3)

Kode Sumber 4.18 berfungsi untuk mengembalikan nilai akhir dari hasil perhitungan metode *Naïve Bayes*. Terdapat 2 tipe output, yaitu barupa nilai probabilitas atau nilai prediksi kelas. *Naïve Bayes* akan memberikan *output* berupa probabilitas ketika ia akan dikombinasikan dengan metode KNN selain itu ia akan memberikan output berupa prediksi kelas dari data testing yang diberikan.

```
1.  gnb = GaussianNB()
2.  gnb.fit(train_atribut,train_class)
3.  probabilitas_bayes=gnb.predict_proba(test_atribut)
4.  return probabilitas_bayes
```

Kode Sumber 4 19 Implementasi Metode Naive Bayes (4)

Kode Sumber 4.19 merupakan implementasi Metode *Naïve Bayes* menggunakan *library* **sklearn,** dimana kode sumber ini hanya akan memberikan *output* berupa nilai probabilitas setiap kelas pada suatu data *testing*.

4.2.9 Implementasi Metode Akurasi

Metode akurasi digunakan untuk menghitung keakuratan hasil prediksi data *testing* terhadap data *training*.

```
1.  def akurasi(prediksikelas,test_class):
2.      test_class=np.array(test_class)
3.      test_class=test_class.astype(int)
4.      hasil = prediksikelas==test_class
5.      jumlah_tptn = sum(hasil)
6.      banyak_hasil = len(hasil)
7.      akurasi=(float(jumlah_tptn)/float(banyak_hasil)
8.      return akurasi*100
```

Kode Sumber 4 20 Implementasi Pehitungan Akurasi

Kode Sumber 4.20 merupakan implementasi dari metode akurasi. Rumus perhitungan metode ini telah dijelaskan pada bab sebelumnya.

4.2.10 Implementasi Program Utama

Pada program utama (*main*) akan digabungkan semua implementasi metode yang telah dijelaskan sebelumnya. *Output* dari program utama ialah berupa model data yang

memiliki tingkat keakuratan paling tinggi untuk kemudian siap dijadikan data *training* selanjutnya. Model data ini disimpan dalam file bertipe txt.

```
1.  data = normalisasi.run(data)
2.  for v in range(k_fold):
3.      [test,train,tests,trains,volds,temp] = splitData.run
    (v,data,tipeKelas,jumlahDataPerKelas,tests,trains,volds,
    temp)
4.      for i in range(100):
5.          [k,bapak,ibu,fitness_e,fitness_m,fitness_c,train
    ,test] = GA.run(k,train, test, seperempat, bapak, ibu, f
    itness_e, fitness_m, fitness_c, tipeKelas)
6.      #euclidean
7.      sortsatuvold_e=sorted(fitness_e, key=lambda fitness_
    e:fitness_e[1], reverse=True)
8.      terbaiksatuvold=[v,sortsatuvold_e[0][0],sortsatuvold
    _e[0][1]]
9.      semuavold_e.append(terbaiksatuvold)
10.     kterbaike=sortsatuvold_e[0][0]
11.     probabilitas_knn_e=KNN.run(kterbaike,train,test,tipe
    Kelas,jarak="euclidean",returnvalue="pro")
12.     #manhattan
13.     sortsatuvold_m=sorted(fitness_m, key=lambda fitness_
    m:fitness_m[1], reverse=True)
14.     terbaiksatuvold=[v,sortsatuvold_m[0][0],sortsatuvold
    _m[0][1]]
15.     semuavold_m.append(terbaiksatuvold)
16.     kterbaikm=sortsatuvold_m[0][0]
17.     probabilitas_knn_m=KNN.run(kterbaikm,train,test,tipe
    Kelas,jarak="manhattan",returnvalue="pro")
18.     #chebyshev
19.     sortsatuvold_c=sorted(fitness_c, key=lambda fitness_
    c:fitness_c[1], reverse=True)
20.     terbaiksatuvold=[v,sortsatuvold_c[0][0],sortsatuvold
    _c[0][1]]
21.     semuavold_c.append(terbaiksatuvold)
22.     kterbaikc=sortsatuvold_c[0][0]
23.     probabilitas_knn_c=KNN.run(kterbaikc,train,test,tipe
    Kelas,jarak="chebyshev",returnvalue="pro")
```

Kode Sumber 4 21 Implementasi Program Utama (1)

Kode Sumber 4.21 merupakan implementasi pada program utama dalam menjalankan metode normalisasi, GA, dan *KNN* pada 3 rumus perhitungan jarak.

```
1.  #bayes
2.  model_gauss_perkolom=nb.gaussian_train(train,tipeKelas)
3.  probabilitas_bayes=nb.gaussian_test(model_gauss_perkolom
    ,tipeKelas,test_atribut,execute="knnbayes")
4.  prediksi_bayes=nb.gaussian_test(model_gauss_perkolom,tip
    eKelas,test_atribut)
5.  akurasi_bayes=evaluasi.akurasi(prediksi_bayes,test_class
    )
6.  akurasipervold=[v,akurasi_bayes]
7.  akurasivold_bayes.append(akurasipervold)
8.  for idx,kelass in enumerate(tipeKelas):
9.      #euclidean
10.     hasiljumlah=probabilitas_knn_e[t][idx][1]+probabilit
    as_bayes[t][idx][1]
11.     avg=hasiljumlah/2
12.     temp2=[kelass,avg]
13.     proe.append(temp2)
14.     #manhattan
15.     hasiljumlah=probabilitas_knn_m[t][idx][1]+probabilit
    as_bayes[t][idx][1]
16.     avg=hasiljumlah/2
17.     temp2=[kelass,avg]
18.             prom.append(temp2)
19.     #chebyshev
20.     hasiljumlah=probabilitas_knn_c[t][idx][1]+probabilit
    as_bayes[t][idx][1]
21.     avg=hasiljumlah/2
22.     temp2=[kelass,avg]
23.     proc.append(temp2)
24. printing.run(start_time,testvold,sortsemuavold_e,sortsem
    uavold_m,sortsemuavold_c,sortsemuavold_bayes,sortsemuavo
    ld_bayesKNN_e,sortsemuavold_bayesKNN_m,sortsemuavold_bay
    esKNN_c)
```

Kode Sumber 4 22 Implementasi Program Utama (2)

Kode Sumber 4.22 merupakan implementasi pada program utama dalam menjalankan metode *Naive Bayes* serta dalam melakukan kombinasi hasil probabilitas *Naive Bayes* terhadap hasil probabilitas dari KNN. Kemudian hasil keseluruhan skenario akan ditampilkan pada *iPython console*.

4.2.11 Implementasi Visualisasi Program

Implementasi visualisasi program pada tugas akhir ini ialah dengan menggunakan fitur bot pada aplikasi Telegram seperti yang telah dijelaskan pada bab sebelumnya.

```
1.  def start1(bot, update):
2.       id = update.message.chat_id
3.       with open("admin1.txt","wb") as textfile:
4.           textfile.write(str(id))
5.
6.  def start2(bot, update):
7.       id = update.message.chat_
8.       with open("admin2.txt","wb") as textfile:
9.           textfile.write(str(id))
10.
11. def start3(bot, update):
12.      id = update.message.chat_
13.      with open("admin3.txt","wb") as textfile:
14.          textfile.write(str(id))
```

Kode Sumber 4 23 Implementasi Visualisasi Program (1)

Kode Sumber 4.23 merupakan implementasi tahap pembuatan perintah-perintah yang ada pada bot. Dalam tugas akhir ini, dibuat 3 perintah yaitu start_1 untuk admin yang bertanggung jawab melakukan perbaikan pada kerusakan BTS berupa bad voice, start_2 untuk admin yang bertanggung jawab dalam melakukan perbaikan pada

kerusakan BTS berupa low coverage,dan start_3 untuk admin yang bertanggung jawab dalam melakukan perbaikan pada kerusakan BTS berupa low throughput. Setiap id admin yang menjalankan perintah diatas akan disalin kedalam file txt sesuai dengan tipe kerusakan yang di-*handle*.

```
1.  def main():
2.       updater = Updater(token='316988090:AAEzHGqQ3Bnh
    _ZFw14zGc00TBjJZzJnTzD4')
3.       dispatcher = updater.dispatcher
4.       dispatcher.add_handler(CommandHandler('start_1'
    ,start1))
5.       dispatcher.add_handler(CommandHandler('start_2'
    ,start2))
6.       dispatcher.add_handler(CommandHandler('start_3'
    ,start3))
7.       updater.start_polling()
8.       updater.idle()
9.  if __name__ == '__main__':
10.      main()
```

Kode Sumber 4 24 Implementasi Visualisasi Program (2)

Kode Sumber 4.24 merupakan implementasi dalam pembuatan bot pada Telegram. Proses ini membutuhkan unique token yang dapat diperoleh dari *BotFather* setelah melakukan pendaftaran bot di Telegram.

Kode Sumber 4.25 berfungsi untuk mengirimkan hasil prediksi kerusakan BTS dari bot ke id akun telegram yang telah didapatkan dari kode sumber 4.23. Pada tahap ini sudah didapatkan model file terbaik yang siap digunakan sebagai data training. Selanjutnya diimplementasikan kembali metode-

metode pada program utama untuk melakukan prediksi kerusakan.

```
1.  start_time = time.time()
2.
3.  def kirim(bot,trouble,namatrouble):
4.      namafile="admin"+str(trouble)+".txt"
5.      namafile="{}".format(namafile)
6.      with open(namafile) as f:
7.          admin_id = f.read()
8.          bot.sendMessage(chat_id=admin_id,text=(namat
    rouble))
9.
10. bot = telegram.Bot('316988090:AAEzHGqQ3Bnh_ZFw14zGc0
    0TBjJZzJnTzD4')
11.
12. #penamaan BTS
13. def random_char(y,jenistrouble):
14.     temp = ''.join(random.choice(string.ascii_letter
    s) for x in range(y))
15.     trouble1="BTS "+str(temp)+" is bad voice detecte
    d"
16.     trouble2="BTS "+str(temp)+" is low coverage dete
    cted"
17.     trouble3="BTS "+str(temp)+" is low throughput de
    tected"
18.     return eval(jenistrouble)
```

Kode Sumber 4 25 Implementasi Visualisasi Program (3)

[Halaman ini sengaja dikosongkan]

BAB V
UJI COBA DAN EVALUASI

Pada bab ini akan dijelaskan hasil uji coba yang dilakukan pada sistem yang telah dikerjakan serta analisa dari uji coba yang telah dilakukan. Pembahasan pengujian meliputi lingkungan uji coba, scenario uji coba yang meliputi uji kebenaran dan uji kinerja serta analisa setiap pengujian.

5.1 Lingkungan Pengujian

Lingkungan uji coba menjelaskan lingkungan yang digunakan untuk menguji implementasi metode gabungan GA, KNN dan *Naïve Bayes* pada Tugas Akhir ini. Lingkungan uji coba meliputi perangkat keras dan perangkat lunak yang dijelaskan sebagai berikut:

1. Perangkat Keras
   Prosesor: Intel® Core™ i3-3240 CPU @ 3.40 GHz
   Memori: 8.00 GB.
   Sistem Operasi: 64-bit.
2. Perangkat Lunak
   Sistem Operasi: Windows 10 Enterprise.
   Perangkat Pengembang: *Spyder*.

5.2 Data Uji Coba

Tahap uji coba pada tugas akhir ini menggunakan 1 dataset dari salah satu perusahaan Telekomunikasi di Indonesia. Data akan dibagi menjadi 2 menjadi data *training* dan data *testing*. Pembagian data dalam tahap ini menggunakan metode 5-*fold validation,* dimana data keseluruhan dibagi menjadi 5 bagian. Sebagaimana visualisasi 5-*fold validation* dapat dilihat pada Bab 2 Gambar 2.1.

Pada fold pertama, 20% bagian pertama dari dataset merupakan data *testing*, sedangkan sisanya menjadi data *training*. Setelah itu, akan dilakukan proses pemilihan nilai *K* oleh GA yang akan digunakan pada metode KNN. Data *testing* akan dihitung nilai probabilitas kelasnya menggunakan kombinasi antara metode KNN dan *Naïve Bayes*. Lalu, nilai probabilitas kelas yang tertinggi merupakan label kelas dari data *testing* tersebut. Pada fold kedua, 20% bagian kedua dari dataset merupakan data *testing*, sedangkan sisanya menjadi data *training*. Begitu seterusnya hingga semua bagian dataset pernah menjadi data *testing*. Hasil dari tahap *fold validation* ini adalah sejumlah data yang tepat hasil prediksinya yang kemudian dibagi dengan seluruh data untuk mendapatkan akurasi.

5.3 Skenario dan Evaluasi Pengujian

Uji coba ini dilakukan untuk menguji apakah fungsionalitas program telah diimplementasikan dengan benar dan berjalan sebagaimana mestinya. Uji coba akan didasarkan pada beberapa skenario untuk menguji kesesuaian dan kinerja aplikasi.

Skenario pengujian terbagi menjadi 2 kelompok utama yaitu pengujian algoritma menggunakan normalisasi data dan tanpa normalisasi data. Setiap kelompok terdiri dari 7 skenario pengujian seperti berikut:

1. Skenario perhitungan akurasi menggunakan metode *Naïve Bayes*.
2. Skenario perhitungan akurasi menggunakan metode gabungan GA dan KNN dengan rumus perhitungan jarak *Euclidean Distance.*

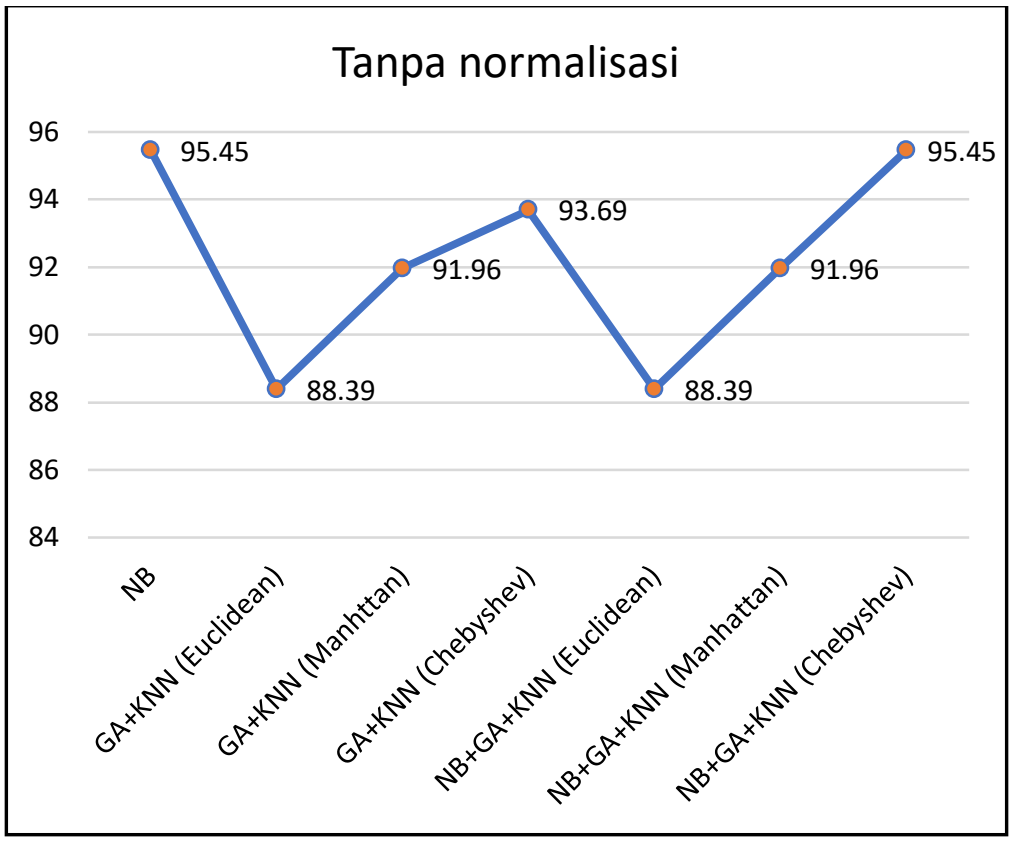
3. Skenario perhitungan akurasi menggunakan metode gabungan GA dan KNN dengan rumus perhitungan jarak *Manhattan Distance.*
4. Skenario perhitungan akurasi menggunakan metode gabungan GA dan KNN dengan rumus perhitungan jarak *Chebyshev Distance*.
5. Skenario perhitungan akurasi menggunakan metode gabungan GA, KNN dan *Naïve Bayes* dengan rumus perhitungan jarak *Euclidean Distance*.
6. Skenario perhitungan akurasi menggunakan metode gabungan GA, KNN dan *Naïve Bayes* dengan rumus perhitungan jarak *Manhattan Distance*.
7. Skenario perhitungan akurasi menggunakan metode gabungan GA, KNN dan *Naïve Bayes* dengan rumus perhitungan jarak *Chebyshev Distance*.

Representasi hasil 7 skenario perhitungan akurasi pada input dataset tanpa menggunakan normalisasi data dapat dilihat pada Tabel 5.1. Sedangkan representasi hasil 7 skenario perhitungan akurasi pada input dataset yang sama namun menggunakan normalisasi data dapat dilihat pada Tabel 5.2.

5.4 Analisis Hasil Uji Coba

Tanpa normalisasi			
	K	Fold	Akurasi
NB		3	95.45
GA+KNN (Euclidean)	44	4	88.39
GA+KNN (Manhttan)	127	4	91.96
GA+KNN (Chebyshev)	71	5	93.69
NB+GA+KNN (Euclidean)	44	4	88.39
NB+GA+KNN (Manhattan)	127	4	91.96
NB+GA+KNN (Chebyshev)	71	5	95.45

Tabel 5 1 Detail akurasi data tanpa normalisasi



Gambar 5 1 Grafik akurasi data tanpa normalisasi

Pada Gambar 5.1, ditampilkan grafik hasil uji coba. Di antara 7 skenario yang diuji tanpa menggunakan normalisasi data, didapatkan akurasi terbaik pada skenario 1 yang merupakan implementasi metode *Naïve Bayes* dan skenario 7 yang merupakan implementasi metode kombinasi dari GA, KNN, dan *Naïve Bayes* dengan perhitungan jarak *Chebyshev Distance*. Setelah dianalisa, ketiga algoritma tersebut memang memberikan kontribusi yang baik. Dimulai dari metode *Naïve Bayes* memiliki kestabilan dalam melakukan klasifikasi data, karena memang pada dasarnya metode *Naïve Bayes* merupakan metode dengan tingkat *independent* yang tinggi. Ia hanya membutuhkan sejumlah fitur model dalam melakukan

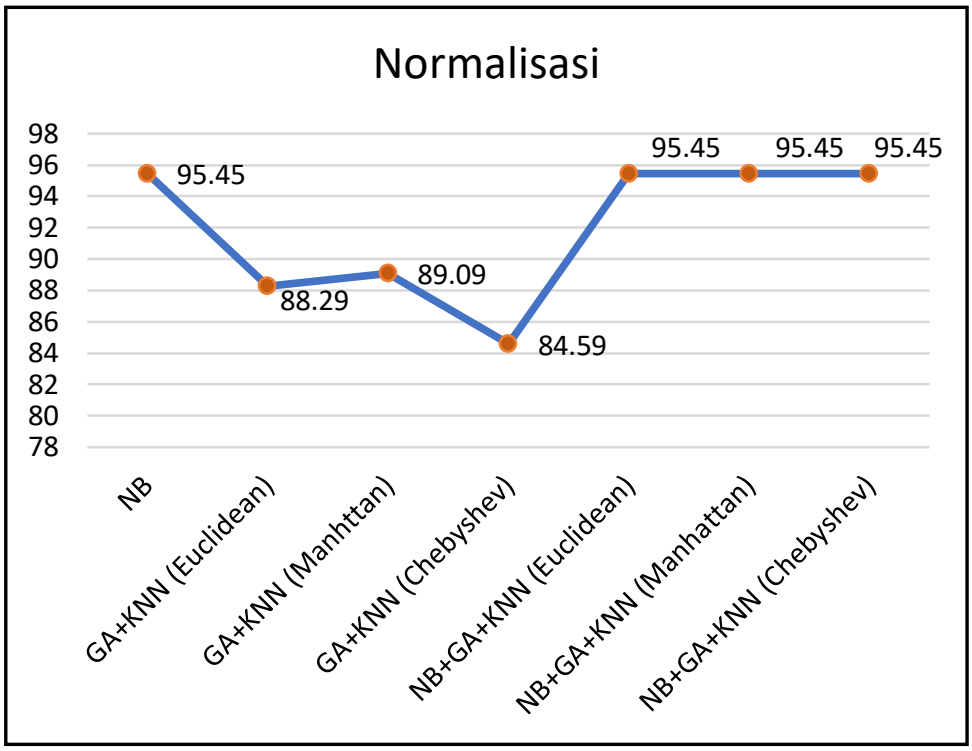
perhitungan dan tidak terkait dengan data *training sample* yang ada. Berbeda dengan metode *Naïve Bayes*, KNN merupakan metode yang dapat dikatakan kurang stabil karena hasil klasifikasi sangat bergantung pada nilai dari data-data lain yang digunakan. Selain itu, nilai *K* dan jenis perhitungan jarak yang digunakan juga memberikan pengaruh besar terhadap hasil output dari metode ini. Dalam mendapatkan nilai *K*, pada tugas akhir ini KNN dibantu oleh metode GA. Secara umum, jenis perhitungan jarak yang digunakan tidak dapat dikatakan mana yang lebih baik, karena pemilihan jenis metode perhitungan ini menyesuaikan dengan kebutuhan sistem yang dibangun. Namun untuk kasus data pada *BTS*, metode *Chebyshev Distance* memberikan hasil tertinggi hingga mencapai 95.45%.

Normalisasi			
	K	Fold	Akurasi
NB		3	95.45
GA+KNN (Euclidean)	44	4	88.29
GA+KNN (Manhttan)	127	4	89.09
GA+KNN (Chebyshev)	71	5	84.59
NB+GA+KNN (Euclidean)	44	4	95.45
NB+GA+KNN (Manhattan)	127	4	95.45
NB+GA+KNN (Chebyshev)	71	5	95.45

Tabel 5 2 Tabel detail akurasi tiap data dengan menggunakan normalisasi

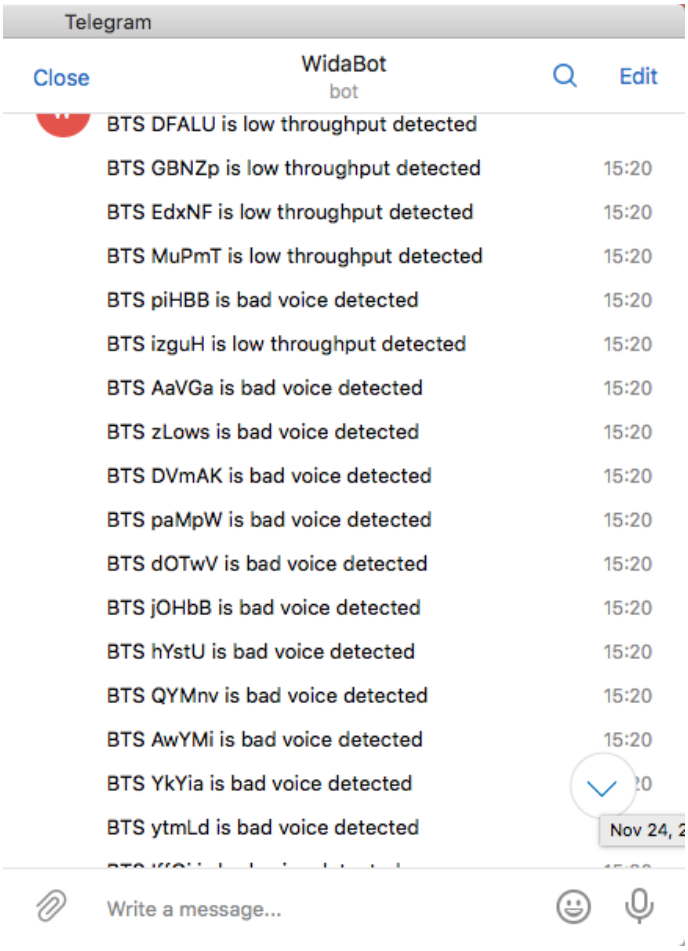
Pada Gambar 5.2, ditampilkan hasil perhitungan 7 skenario dengan menggunakan normalisasi data terlebih dahulu. Didapatkan bahwa dengan dilakukannya normalisasi data, skenario uji coba yang menggunakan metode KNN mengalami peningkatan nilai akurasi. Meskipun demikian

metode kombinasi antara GA, KNN, dan *Naïve Bayes* tetap unggul dengan menghasilkan tingkat akurasi sebesar 95.45%. Hal tersebut membuktikan bahwa KNN sangat bergantung pada data lainnya sehingga jika terdapat rentang yang jauh antar data maka dapat mengakibatkan tidak optimalnya perhitungan pada metode ini. Berbeda dengan skenario uji coba yang menggunakan metode *Naïve Bayes*, ia sama sekali tidak mengalami perubahan nilai akurasi. Hal tersebut membuktikan bahwa metode *Naïve Bayes* sangat bersifat *independent* dan stabil, dengan kata lain ia tidak bergantung pada data lainnya melainkan hanya membutuhkan fitur model dari data *training sample*. Berapapun rentang antar data tidak akan mempengaruhi nlai dari fitur model yang dihasilkan. Kombinasi ketiga algoritma tersebut akan menstabilkan tingkat akurasi yang didapat serta meningkatkan keakuratan hasil prediksi.



Gambar 5 2 Grafik akurasi data menggunakan normalisasi

Model data dan algoritma terbaik dari implementasi metode-metode tersebut diatas akan digunakan sebagai acuan dalam melakukan visualisasi pada bot telegram.



Gambar 5 3 Visualisasi bot telegram

Gambar 5.1 merupakan penampakan hasil visualisasi pada bot telegram dari 577 BTS bagian Jawa Timur. Setiap admin hanya akan menerima satu kali pemberitahuan kerusakan pada BTS yang menjadi tanggung jawab admin tersebut. Nama BTS disamarkan menggunakan *random string* sesuai implementasi

pada bab sebelumnya. Pemilahan admin yang mendapatkan pemberitahuan, didasari oleh inisialisasi yang ia lakukan ketika mendaftar pada bot telegram. Sama halnya dengan *random string*, tahap inisialisasi admin pada bot telegram telah dijelaskan pula pada bab sebelumnya.

BAB VI
KESIMPULAN DAN SARAN

Bab ini membahas mengenai kesimpulan yang dapat diambil dari hasil uji coba yang telah dilakukan sebagai jawaban dari rumusan masalah yang dikemukakan. Selain kesimpulan, juga terdapat saran yang ditujukan untuk pengembangan penelitian lebih lanjut.

6.1 Kesimpulan

Dari hasil uji coba yang telah dilakukan terhadap pembuatan model, dapat diambil kesimpulan sebagai berikut:

1. Adanya implementasi GA dalam mendukung algoritma KNN mengakibatkan tidak lagi membutuhkan penentuan nilai *K* secara manual. Sehingga nilai *K* yang didapat akan lebih beragam, namun bukan tanpa dasar seperti menggunakan variable random biasa.
2. Kombinasi GA, KNN, dan *Naïve Bayes* memberikan nilai akurasi yang tinggi dalam memprediksi kerusakan pada BTS. Ketiga algoritma tersebut saling memberikan kontribusi untuk mendapatkan prediksi yang akurat.
3. Algoritma *K-Fold Validation* yang digunakan dalam menentukan kelompok data uji menghasilkan model data yang memiliki performa terbaik.
4. Pada KNN, metode normalisasi memiliki peranan penting sehingga dapat meningkatkan performa dari hasil klasifikasi data. Hal ini disebabkan karena KNN merupakan metode yang sangat bergantung pada kelompok data lainnya, sehingga jika terdapat rentang yang jauh antar data, maka dapat

mengakibatkan tidak optimalnya perhitungan pada metode ini. Sedangkan pada algoritma *Naïve Bayes,* metode normalisasi tidak memiliki peran khusus sehingga tidak terjadi perubahan performa ketika diimplementasikan.

5. Pemilihan metode perhitungan jarak yang digunakan dalam algoritma KNN sebenarnya memiliki peranan yang penting. Namun tidak dapat ditentukan mana yang lebih baik, karena pemilihan jenis metode perhitungan ini menyesuaikan dengan kebutuhan sistem yang dibangun atau menyesuaikan pada dataset yang diberikan.
6. Pada hasil uji coba data tanpa normalisasi, didapatkan bahwa nilai akurasi tertinggi diperoleh ketika menggunakan metode kombinasi dari GA, KNN, dan *Naïve Bayes* dengan mencapai nilai 95.45% pada data BTS.
7. Pada hasil uji coba data menggunakan normalisasi, didapatkan bahwa nilai akurasi tertinggi diperoleh ketika menggunakan metode kombinasi dari GA, KNN, dan *Naïve Bayes* dengan mencapai nilai 95.45% pula pada data BTS.

6.2 Saran

Saran yang diberikan untuk pengembangan aplikasi ini adalah:

1. Untuk mencapai kestabilan akurasi, dapat dicoba menggunakan suatu rumus perhitungan tertentu untuk menentukan beberapa parameter pada GA, terutama jumlah iterasi agar dapat mencapai keseluruhan data.
2. Data testing yang digunakan dibuat time series agar sesuai dengan kebutuhan BTS.

DAFTAR PUSTAKA

[1] N. Ismail, Maharoni And L. Innel, "Analisis Perencanaan Pembangunan Bts (Base Transceiver Station) Berdasarkan Faktor Kelengkungan Bumi Dan Daerah Fresnel Di Regional Project Sumatera Bagian Selatan," Vol. Ix, No. 1979, P. 8911, 2015.

[2] S. W. Handoko, "Analisa Dan Optimasi Quality Of Service (Qos) Layanan Voice Dalam Jaringan Selular Cdma 2000 1x Telkom Flexi Regional Operation Semarang".

[3] G. D. Karunia, Y. S. Rohmah And A. Purwanto, "Penanganan Interferensi Pada Jaringan Seluler 3g Pt.Indosat Untuk Area Bandung," *Interference Problem Solving On 3g Celuler Network Pt.Indosat For Bandung Area,* 08 2015.

[4] L. Huawei Technologies Co., "Clarification For Higher Rtwp In Lampsite".

[5] A. C. Dewana, I. Santoso And A. A. Z.M., "Analisis Kualitas Panggilan Layanan Suara (Voice) Sistem Wcdma Saat Terjadi Drop Call Berdasarkan Data Statistik Dan Drive Test".

[6] W. Karner, "A Umts Dl Dch Bler Model Based On Measurements In Live Networks".

[7] N. W. Sianipar, T. Brotoharsono And F. A. Yulianto, "Analisis Performansi Web Server Dengan Algoritma Fastest Connection First," Telkom University, Bandung, 2008.

[8] M. R. Ridho, "Lte (Long Term Evolution) Dan Layer Fisikanya," 23 05 2014. [Online]. Available: Https://Emerer.Com/Lte-Long-Term-Evolution-Dan-Layer-Fisiknya/. [Accessed 11 07 2017].

[9] Digitrainee, "Wcdma Rno Rf Optimization," Huawei Technologies Co., Ltd..

[10] Zuriani Mustaffa And Yuhanis Yusof, "A Comparison Of Normalization Techiques In Predicting Dengue Outbreak," *International Conference Of Business And Economics Research ,* Vol. 1, 2010.

[11] L. T. A. H. L. P. Refaeilzadeh, "Cross-Validation," 11 November 2008. [Online]. Available: Http://Leitang.Net/Papers/Ency-Cross-Validation.Pdf.. [Accessed 1 June 2017].

[12] J. M.A., D. B. L And C. P, "Classification Of Heart Disease Using K-Nearest Neighbor And Genetic Algorithm," *Sciencedirect,* 2013.

[13] J. M. A, D. B. L And C. P, "Heart Disease Prediction System Associative Classification And Genetic Algorithm," *International Conference On Emerging Trends In Electrical, Electronomics And Communication Technologies-Icecit,* 2012.

[14] D. A. S. R. Kaushik, "Study Of Euclidean And Manhattan Distance Metrics Using Simple K-Means Clustering," *International Journal For Research In Applied Science And Engineering Technology (Ijraset),* Vol. 2, P. V, 2014.

[15] Kahkashan Kouser, Sunita And Assistant Professor, Dept. Of Cambridge Institute , "A Comparative Study Of K Means Algorithm By Different Distance Measures," *International Journal Of Innovative Research In Computer And Communication Engineering,* Vol. 1, No. 9, P. 2013.

[16] A. Saleh, "Implementasi Metode Klasifikasi NaïVe Bayes Dalam Memprediksi Besarnya Penggunaan Listrik Rumah Tangga," *Citec Journal,* Vol. 2, P. 3, May 2015.

[17] A. K. S. A. C. J. Christy, "Genetic Algorithm And Confusion Matrix For Document Clustering," *Ijcsi International Journal Of Computer Science,* Vol. 9, P. 1, 2012.

[18] (. S.-1.D, "Documentation Of Scikit-Learn 0.18," 2016. [Online]. Available: Http://Scikit-Learn.Org/Stable/Documentation.Html.. [Accessed 2017].

[19] Sumanjani .P And R. Gayathri Devi, "Improved Classification Techniques By Combining Knn And Random Forest With Naive Bayesian Classifier," *Ieee International Conference On Engineering And Technology (Icetech),* 2015.

[Halaman ini sengaja dikosongkan]

LAMPIRAN 1

Potongan dataset dari salah satu perusahaan telekomunikasi di Indonesia.

1	VSMeanRTWP	UL BLER	PRB Utilization	RSCP	ECNO	TROUBLE
2	-89.43	9	46.87675781	-95.85	-8.38	3
3	-89.61	15	39.90574449	-95.6	-8.36	3
4	-89.85	12	27.71606105	-95.35	-8.34	3
5	-90.77	12	13.8135514	-95.1	-8.32	3
6	-90.87	22	22.53777344	-94.85	-8.3	1
7	-91.00	12	22.04794922	-94.6	-8.28	1
8	-91.43	9	12.70438368	-94.35	-8.26	1
9	-91.48	12	18.02136628	-94.1	-8.24	1
10	-92.02	18	42.61894531	-93.85	-8.22	1
11	-92.15	18	24.44714844	-93.6	-8.2	1
12	-92.25	9	37.76289063	-93.35	-8.18	1
13	-92.71	9	42.09300373	-93.1	-8.16	1
14	-92.73	0	75.44365234	-92.85	-8.14	2
15	-92.75	18	30.07148438	-92.6	-8.12	1

530	-106.74	12	9.261279297	-65.72	-2.16	1
531	-106.80	6	2.383173077	-38.74	-2.18	0
532	-106.86	3	1.956009615	-32.92	-2.2	0
533	-106.90	12	16.63226744	-65.7	-2.22	1
534	-106.91	0	11.10240385	-38.72	-2.24	0
535	-106.99	9	51.99014423	-32.9	-2.26	0
536	-107.31	6	3.298828125	-65.68	-2.28	0
537	-107.60	9	7.24296875	-38.7	-2.3	0
538	-107.63	3	5.435390625	-32.88	-2.32	0
539	-77.23	12	22.78238636	-99.3	-12	3
540	-79.68	3	7.654927885	-99.35	-12.3	3

BIODATA PENULIS

Wida Dwitiayasa, lahir pada tanggal 20 Mei 1996 di Balikpapan. Penulis menempuh pendidikan mulai dari SDN 007 Samarinda (2001-2006), SDS Laboratorium Unesa Surabaya (2006-2007), SMPN 21 Surabaya (2007-2010), SMAN 15 Surabaya (2011-2014) dan S1 Teknik Informatika ITS (2014-2018).

Selama masa kuliah, penulis aktif dalam organisasi Himpunan Mahasiswa Teknik Computer (HMTC). Diantaranya adalah menjadi staff departemen kesejahteraan mahasiswa himpunan mahasiswa teknik computer ITS 2015-2016. Penulis juga aktif dalam kepanitiaan Schematics. Diantaranya penulis pernah menjadi staff Revolutionary Expo with Various Art Schematics ITS 2015-2016. Selain organisasi jurusan, penulis juga aktif dalam organisasi kerohanian dengan menjadi Sekertaris Departmen Komunikasi dan Informasi Tim Pembina Kerohanian Hindu ITS.

Selama kuliah di teknik informatika ITS, penulis mengambil bidang minat Dasar Terapan dan Komputasi (DTK). Penulis pernah menjadi asisten mata kuliah S1-Sistem Digital, S1-Komputasi Numerik, S1-Probabilitas Statistika, D1-Design Grafis, D1-Design Web Professional, D1-Teknologi Basis Data. Komunikasi dengan penulis dapat melalui email : **widadwitiayasa@gmail.com.**

[Halaman ini sengaja dikosongkan]