Institut
Teknologi
Sepuluh Nopember

TUGAS AKHIR - KI141502

Implementasi Algoritma K-Nearest Neighbors dengan Particle Swarm Optimization dalam Klasifikasi Trouble pada Base Transceiver Station (BTS)

Yusuf Dimas Hermawan
NRP 5114100159

Dosen Pembimbing I
Victor Hariadi, S.Si., M.Kom.

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

DEPARTEMEN INFORMATIKA
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember
Surabaya 2017

[Halaman ini sengaja dikosongkan]

TUGAS AKHIR - KI141502

Implementasi Algoritma K-Nearest Neighbors dengan Particle Swarm Optimization dalam Klasifikasi Trouble pada Base Transceiver Station (BTS)

Yusuf Dimas Hermawan
NRP 5114100159

Dosen Pembimbing I
Victor Hariadi, S.Si., M.Kom.

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

DEPARTEMEN INFORMATIKA
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember
Surabaya 2017

[Halaman ini sengaja dikosongkan]

UNDERGRADUATE THESES – KI141502

Implementation of K-Nearest Neighbors Algorithm with Particle Swarm Optimization in Trouble Classification on Base Transceiver Station (BTS)

Yusuf Dimas Hermawan
NRP 5112100159

Supervisor I
Victor Hariadi, S.Si., M.Kom.

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

DEPARTMENT OF INFORMATICS
FACULTY OF INFORMATION and COMMUNICATION TECHNOLOGY
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA 2017

[Halaman ini sengaja dikosongkan]

LEMBAR PENGESAHAN

Implementasi Algoritma K-Nearest Neighbors dengan Particle Swarm Optimization dalam Klasifikasi Trouble pada Base Transceiver Station (BTS)

TUGAS AKHIR

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

Oleh :
YUSUF DIMAS HERMAWAN
NRP : 5114 100 159

Disetujui oleh Dosen Pembimbing Tugas Akhir:

Victor Hariadi, S.Si., M.Kom. .....
NIP: 196912281994121001 (pembimbing 1)

Bilqis Amaliah, S.Kom., M.Kom. .....
NIP: 197509142001122002 (pembimbing 2)

SURABAYA
JANUARI 2017

[Halaman ini sengaja dikosongkan]

IMPLEMENTASI FITUR SIFT DAN FISHER VECTOR ENCODING UNTUK APLIKASI PENGENALAN WAJAH

Nama Mahasiswa	**: Yusuf Dimas Hermawan.**
NRP	**: 5114100159**
Jurusan	**: Teknik Informatika FTIF-ITS**
Dosen Pembimbing 1	**: Victor Hariadi, S.Si., M.Kom.**
Dosen Pembimbing 2	**: Bilqis Amaliah, S.Kom, M.Kom.**

ABSTRAK

Teknologi Informasi adalah suatu teknologi yang digunakan untuk mengolah data, termasuk memproses, mendapatkan, menyusun, menyimpan, atau memanipulasi data dalam berbagai cara untuk menghasilkan informasi yang berkualitas yaitu informasi yang relevan, akurat dan tepat waktu Maka demikian, trafik kebutuhan akses data, voice maupun SMS membuat banyak pengembang operator seluler mulai mengembangkan fitur-fitur canggih yang dapat menjawab berbagai kebutuhan pengguna. Salah satu fitur yang dimaksud adalah berupa pengembangan teknologi jaringan berupa layanan 1G, 2G, 3G hingga 4G. Akan tetapi teknologi-teknologi yang dianggap semakin canggih ini tetap memiliki keterbatasan sehingga tak jarang jika mengalami beberapa kerusakan (error) pada sistem.

Pada tugas akhir ini penulis berupaya melakukan klasifikasi error pada Base Transceiver Station (BTS) secara otomatis. Menggunakan K- Nearest Neighbors (KNN) yang dioptimasi dengan Particle Swarm Optimization (PSO).

Dataset yang digunakan dalam proses uji coba merupakan 8 dataset dari UCI machine learning dan 1 dataset dari salah satu perusahaan Telekomunikasi di Indonesia yang dibagi menjadi 4

dan 5 bagian sebagai data testing dengan menggunakan K-Fold Validation. Akurasi terbaik sebesar 100%.

Kata kunci: K-Nearest Neighbor, Particle Swarm Optimization, BTS, Klasifikasi.

SIFT FEATURE AND FISHER VECTOR ENCODING IMPLEMENTATION FOR FACE RECOGNITION APPLICATION

Student's Name	**: Yusuf Dimas Hermawan.**
Student's ID	**: 5114100159**
Department	**: Teknik Informatika FTIF-ITS**
First Advisor	**: Victor Hariadi, S.Si., M.Kom.**
Second Advisor	**: Bilqis Amaliah, S.Kom, M.Kom.**

ABSTRACT

Information Technology is a technology used to process data, including processing, obtaining, compiling, storing, or manipulating data in various ways to produce quality information that is relevant information, accurate and timely Thus, traffic needs data access, voice and SMS makes many developers mobile operators begin to develop advanced features that can answer the various needs of users. One of the features in question is the development of network technology in the form of services 1G, 2G, 3G up to 4G. However, these technologies are considered increasingly sophisticated still have limitations so that not infrequently if you experience some damage (error) on the system.

In this final project the author attempts to classify the error on Base Transceiver Station (BTS) automatically. Using K-Nearest Neighbors (KNN) optimized with Particle Swarm Optimization (PSO).

The dataset used in the process is 8 datasets from UCI machine learning and 1 dataset from one Telecommunication company in Indonesia which is divided into 4 and 5 sections as data testing using K-Fold Validation. Best accuracy of 100%.

Keyword: K-Nearest Neighbors, Particle Swarm Optimization, BTS, Classification.

[Halaman ini sengaja dikosongkan]

KATA PENGANTAR

بِسْمِ ٱللَّهِ ٱلرَّحْمَٰنِ ٱلرَّحِيمِ

Alhamdulillahirabbil'alamin, segala puji dan syukur bagi Allah SWT, yang telah melimpahkan rahmat, dan hidayah-Nya sehingga penulis dapat menyelesaikan Tugas Akhir yang berjudul

Implementasi Algoritma K-Nearest Neighbors dengan Particle Swarm Optimization dalam Klasifikasi Trouble pada Base Transceiver Station (BTS)

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. Allah SWT, karena atas izin-Nya lah penulis dapat menyelesaikan Tugas Akhir dengan baik.
2. Kedua orang tua penulis, terima kasih atas doa dan bantuan moral dan materil 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 Victor Hariadi, S.Si.,M.Kom. 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 pembimbing I Tugas Akhir yang telah memberikan banyak waktu untuk berdiskusi dan memberi semangat dan motivasi kepada penulis untuk menyelesaikan Tugas Akhir.
7. Ibu Bilqis Amaliah, S.Kom, M.Kom selaku pembimbing II Tugas Akhir yang telah memberikan banyak waktu untuk berdiskusi dan memberi semangat dan motivasi kepada penulis untuk menyelesaikan Tugas Akhir.
8. 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.
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. Andhika Rizky Pratiwi yang telah menemani dan mendukung pengerjaan Tugas Akhir ini.
12. Rekan-rekan Sunset Dhika, Wijdan, Dharu, Niar, Lubna, Tisa, Eva, Ari, Yudis, Yana. Yang telah yang telah membantu dan sangat mendukung pengerjaan Tugas Akhir ini.
13. Rekan-rekan di Ngagel Jaya 30 Lt 2 Room 2 Andre, Azzam, Adam, Yudis, Ardhan, Diva, Dito. Yang telah membantu dan sangat mendukung pengerjaan Tugas Akhir ini.
14. Rekan-rekan Soritia Rizal, Romi, Evan, Danis, Adiwinoto, Satria, Anggit, Rian, Adit, Andre, Fito, Angga, Rochman, OjanGG, Bagas, Iqbal, Digoz. Yang telah membantu dan sangat mendukung pengerjaan Tugas Akhir ini.
15. Rekan-rekan thrower EVAN BANGUN, Adiwintolien, Babang Dwiyan, Danis, Rizal. Yang telah memberi banyak warna merah di 1 page recent match.

16. Rekan-rekan perjuangan Wida, Rifat, Rian, Rizal, Rani. Yang telah membantu dan sangat mendukung pengerjaan Tugas Akhir ini. Terimakasih atas dukungan dan motivasi yang diberikan selama penulis menempuh kuliah di Teknik Informatika ITS.
17. Seluruh rekan-rekan TC 2014 yang saya banggakan.

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, Desember 2017

Yusuf Dimas Hermawan

[Halaman ini sengaja dikosongkan]

DAFTAR ISI

LEMBAR PENGESAHAN vii
ABSTRAK ix
ABSTRACT xi
KATA PENGANTAR xiii
DAFTAR ISI xvii
DAFTAR GAMBAR xix
DAFTAR TABEL xxi
DAFTAR KODE SUMBER xxiii
1 BAB I PENDAHUALUAN 1
1.1. Latar Belakang 1
1.2. Rumusan Permasalahan 3
1.3. Batasan Masalah 3
1.4. Tujuan Pembuatan Tugas Akhir 3
1.5. Manfaat Tugas Akhir 4
1.6. Metodologi 4
1.7. Sistematika Penulisan Laporan Tugas Akhir 6
2 BAB II DASAR TEORI 9
2.1. Base Transceiver Station 9
2.1.1. Bad Voice 9
2.1.2. Low Throughput 10
2.1.3. Low Coverage 11
2.2. K-Nearest Neighbor 11
2.3. Particle Swarm Optimization 14
2.4. Bot Telegram 20
2.5. K-Fold Validation 20
3 BAB III PERANCANGAN PERANGKAT LUNAK 23
3.1. Desain Metode Secara Umum 23
3.1.1. Processing 24
4 BAB IV IMPLEMENTASI 29
4.1. Lingkungan implementasi 29
4.2. Implementasi 29
4.2.1. Parameter yang digunakan 29
4.2.2. Implementasi Fungsi Telegram 30

4.2.3. Implementasi Fungsi Readfile 31
4.2.4. Implementasi Metode 4-Fold Validation 32
4.2.5. Implementasi Metode 5-Fold Validation 33
4.2.6. Implementasi Metode Particle Swarm Optimization 34
4.2.7. Implementasi Metode K-Nearest Neighbor 36
5 BAB V PENGUJIAN DAN EVALUASI 39
5.1. Lingkungan Pengujian 39
5.2. Data Uji Coba 39
5.3. Skenario dan Evaluasi Pengujian 39
5.4. Analisis Hasil Uji Coba 40
5.4.1. Hasil dengan menggunakan 4-Fold 41
5.4.2. Hasil dengan menggunakan 5-Fold 47
5.4.3. Hasil notifikasi BOT Telegram 54
6 BAB VI KESIMPULAN DAN SARAN 55
6.1. Kesimpulan 55
6.2. Saran 55
7 DAFTAR PUSTAKA 56
BIODATA PENULIS 57

DAFTAR GAMBAR

Gambar 2.1 Flowchart KNN.....14
Gambar 2.2 Flowchart PSO.....19
Gambar 2.3 Ilustrasi 4-Fold Validation.....20
Gambar 2.4 Ilustrasi 5-Fold Validation.....20
Gambar 3.1 Desain Metode Secara Umum.....23
Gambar 5.1 Grafik akurasi 4 fold.....46
Gambar 5.2 Grafik akurasi 5 fold.....53
Gambar 5.3 Grafik akurasi 4 fold dan 5 fold.....53
Gambar 5.4 Notifikasi bot Telegram.....54

[Halaman ini sengaja dikosongkan]

DAFTAR TABEL

Tabel 4.1 Parameter yang digunakan.....30
Tabel 5.1 Tabel akurasi 4 fold 41
Tabel 5.2 Tabel detail akurasi 4 fold 42
Tabel 5.3 Tabel akurasi 5 fold 47
Tabel 5.4 Tabel detail akurasi 5 fold 48

[Halaman ini sengaja dikosongkan]

DAFTAR KODE SUMBER

code 3.1 Pseudocode 4-Fold validation 25
code 3.2 Pseudocode 5-Fold validation 26
code 4.1 Notifikasi telegram 30
code 4.2 read file.....31
code 4.3 4-Fold 32
code 4.4 5-fold 33
code 4.5 Inisialisiasi partikel.....34
code 4.6 Evaluasi 35
code 4.7 Update kecepatan 35
code 4.8 Update posisi 36
code 4.9 euclidean distance.....36
code 4.10 manhattan distance 37
code 4.11 chebyshev distance 37
code 4.12 gettetangga.....38
code 4.13 Getrespon.....38

[Halaman ini sengaja dikosongkan]

BAB I
PENDAHUALUAN

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

Teknologi Informasi adalah suatu teknologi yang digunakan untuk mengolah data, termasuk memproses, mendapatkan, menyusun, menyimpan, atau memanipulasi data dalam berbagai cara untuk menghasilkan informasi yang berkualitas yaitu informasi yang relevan, akurat dan tepat waktu. Informasi ini digunakan dalam berbagai bentuk keperluan seperti keperluan pribadi, bisnis, dan pemerintahan. Terdapat tiga aspek penting dalam perkembangan teknologi informasi, yaitu perangkat hardware berupa komputer atau sejenisnya yang berperan dalam mengolah data, sistem jaringan yang berguna untuk menghubungkan satu komputer dengan komputer lainnya, dan teknologi komunikasi yang digunakan untuk mendistribusikan data sehingga dapat diakses secara global. Seiring dengan perkembangan teknologi informasi, teknologi komunikasi pun berkembang secara pesat. Maka munculah istilah teknologi komunikasi cyber atau internet. Contoh teknologi komunikasi yang menggunakan teknologi cyber adalah e-mail, chatting dan lain sebagainya. Teknologi komunikasi yang seperti ini lah yang dianggap berperan dominan sebagai salah satu aspek pembangun bangsa. Maka demikian, trafik kebutuhan akses data, voice maupun SMS membuat banyak pengembang operator seluler mulai mengembangkan fitur-fitur canggih yang dapat menjawab berbagai kebutuhan pengguna. Salah satu fitur yang dimaksud

adalah berupa pengembangan teknologi jaringan berupa layanan 1G, 2G, 3G hingga 4G. Akan tetapi teknologi-teknologi yang dianggap semakin canggih ini tetap memiliki keterbatasan sehingga tak jarang jika mengalami beberapa kerusakan (error) pada sistem, yang dalam perbaikannya dibutuhkan pengetahuan jenis kerusakan apa yang sedang dialami oleh sistem. Keterlambatan dalam mendapatkan jenis kerusakan yang dialami, tentu akan berdampak pada keterlambatan penanganan perbaikan sistem pula sehingga menyebabkan tidak tercapainya kebutuhan pengguna sebagaimana yang diharapkan.

Pada tugas akhir ini penulis berupaya melakukan klasifikasi error system secara otomatis. Salah satu metode klasifikasi yang sering digunakan adalah K-Nearest Neighbors (KNN). KNN adalah suatu metode umum yang termasuk dalam supervised learning yaitu dimana hasil dari class data yang baru diklasifikasikan adalah berdasarkan mayoritas dari kategori K tetangga terdekat. Namun algoritma ini memiliki beberapa kekurangan, salah satunya KNN tidak dapat mengetahui berapa nilai K optimal yang memiliki performa terbaik. Maka itu KNN akan dioptimasi menggunakan algoritma Particle Swarm Optimization dimana algoritma ini berperan dalam menghasilkan nilai K untuk selanjutnya diproses menggunakan KNN. PSO memiliki kelebihan yaitu konsep yang sederhana, mudah diimplementasikan, dan efisien dalam perhitungan jika dibandingkan dengan algoritma matematika dan teknik optimisasi heuristik lainnya.

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 cara implementasi algoritma K-Nearest Neighbors?
2. Bagaimana cara menemukan nilai K terbaik dengan menggunakan PSO?
3. Bagaimana cara mengkombinasikan antara KNN dan PSO?
4. Bagaimana performa KNN-PSO?

1.3. Batasan Masalah

Permasalahan yang dibahas dalam tugas akhir ini memiliki beberapa batasan antara lain :

1. Menggunakan bahasa pemrograman Python.
2. Dataset yang digunakan diambil dari salah satu perusahaan telekomunikasi di Indonesia dan 8 dataset dari UCI Machine Learning Repository.
3. Metode utama yang digunakan adalah K-Nearest Neighbors dan Particle Swarm Optimization. K-Nearest Neighbors digunakan sebagai metode pembanding.
4. Evaluasi dilakukan terhadap akurasi dataset menggunakan metode Accuracy pada Confusion Matrix.
5. Validasi dilakukan dengan K-fold validation.
6. Deployment aplikasi dilakukan dalam bentuk bot Telegram.

1.4. Tujuan Pembuatan Tugas Akhir

Tujuan dari pembuatan tugas akhir ini antara lain:

1. Untuk mengetahui cara implementasi algoritma *K-Nearest Neighbors.*
2. Untuk mengetahui cara optimasi algoritma *K-Nearest Neighbors* menggunakan *Particle Swarm Optimization.*

3. Untuk mengetahui cara mengukur performa dari algoritma *KNN* dan *PSO*.
4. Membuat sistem pembertitahun yang otomatis.

1.5. Manfaat Tugas Akhir

Manfaat dari hasil pembuatan tugas akhir ini antara lain :

1. Aplikasi dapat digunakan untuk melakukan tugas klasifikasi jenis trouble pada Base Transceiver Station (BTS).
2. Dapat mengetahui efektivitas optimasi oleh algoritma PSO pada algoritma KNN dalam melakukan klasifikasi jenis trouble Base Transceiver Station (BTS).

1.6. Metodologi

Tahapan-tahapan yang 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 Base Transceiver Station (BTS) menggunakan metode Particle Swarm Optimization dan K-Nearest Neighbor.

2. Studi literatur

   Pada tahap ini dilakukan pencarian, pengumpulan, penyaringan, pemahaman, dan pembelajaran literatur yang berhubungan dengan fitur *Particle Swarm Optimization*, *K-Nearest Neighbor*, dan *K-fold validation*. Literatur yang digunakan meliputi: buku referensi, 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 suatu 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 terhadap perangkat lunak yang telah dibuat untuk mengetahui kemampuan algoritma yang dipakai, mengamati kinerja sistem, serta mengidentifikasi kendala yang mungkin timbul pada aplikasi yang dibuat.

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 pengerjaan tugas akhir ini. Selain itu, diharapkan dapat berguna untuk pembaca yang tertarik untuk melakukan pengembangan lebih lanjut. Secara garis besar, buku tugas akhir terdiri atas beberapa bagian seperti berikut ini.

1. 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.
2. 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.
3. Bab III Perancangan Perangkat Lunak
   Bab ini berisi penjelasan mengenai desain dan perancangan sistem yang direpresentasikan dalam bentuk *pseudocode* dan *flowchart*.
4. Bab IV. Implementasi
   Bab ini merupakan pembangunan aplikasi dengan *python* sesuai permasalahan dan batasan yang telah dijabarkan pada Bab I.
5. Bab V. Hasil Uji Coba dan Evaluasi
   Bab ini berisi penjelasan mengenai data hasil percobaan, pengukuran, dan pembahasan mengenai hasil percobaan yang telah dilakukan.

6. 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*

Base Transceiver Station (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:

a. 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.

b. 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. 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 *base transceiver station*. Dalam tugas akhir ini, acuan nilai PRB adalah **<=70%** 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. Dalam tugas akhir ini, acuan nilai Ec/No adalah 0 hingga (- 8)dB 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 sesuai dengan standard yang dimiliki oleh salah satu perusahaan telekomunikasi di Indonesia.

2.2. *K-Nearest Neighbor*

Algoritma k-nearest neighbor (k-NN atau KNN) adalah sebuah metode untuk melakukan klasifikasi terhadap objek berdasarkan data pembelajaran yang jaraknya paling dekat dengan objek tersebut.

K-Nearest Neighbor berdasarkan konsep ‘learning by analogy’. Data learning dideskripsikan dengan atribut numerik n-dimensi. Tiap data learning merepresentasikan sebuah titik,

yang ditandai dengan c, dalam ruang n-dimensi. Jika sebuah data query yang labelnya tidak diketahui diinputkan, maka K-Nearest Neighbor akan mencari k buah data learning yang jaraknya paling dekat dengan data query dalam ruang n-dimensi. Jarak antara data query dengan data learning dihitung dengan cara mengukur jarak antara titik yang merepresentasikan data query dengan semua titik yang merepresentasikan data learning dengan rumus Euclidean Distance.

Pencarian jarak minimum ini menggunakan rumus Euclidean Distance seperti yang dituliskan seperti berikut:

a. Euclideance Distance

$$D = \sqrt{\sum_{k=1}^{m} (x_{ik} - x_{jk})^2} \qquad (2.1)$$

b. Manhattan Distance

$$D = \sum_{k=1}^{m} |x_{ik} - x_{jk}| \qquad (2.2)$$

c. Chebyshev Distance

$$D = max_k \mid x_{ik} - x_{jk} \mid \qquad (2.3)$$

Pada fase training, algoritma ini hanya melakukan penyimpanan vektor- vektor fitur dan klasifikasi data training

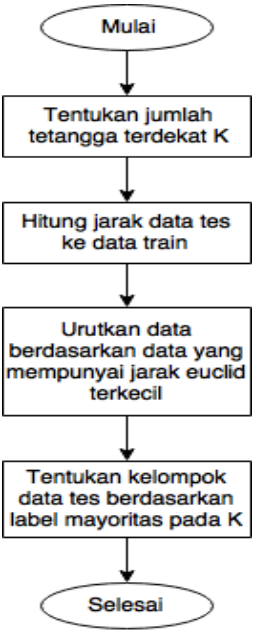
sample. Pada fase klasifikasi, fitur – fitur yang sama dihitung untuk testing data (klasifikasinya belum diketahui). Jarak dari vektor yang baru ini terhadap seluruh vektor training sample dihitung, dan sejumlah k buah yang paling dekat diambil. Titik yang baru klasifikasinya diprediksikan termasuk pada klasifikasi terbanyak dari titik – titik tersebut.

Nilai k yang terbaik untuk algoritma ini tergantung pada data; secara umumnya, nilai k yang tinggi akan mengurangi efek noise pada klasifikasi, tetapi membuat batasan antara setiap klasifikasi menjadi lebih kabur. Nilai k yang bagus dapat dipilih dengan optimasi parameter, misalnya dengan menggunakan cross-validation. Kasus khusus di mana klasifikasi diprediksikan berdasarkan data pembelajaran yang paling dekat (dengan kata lain, k = 1) disebut algoritma nearest neighbor.

Ketepatan algoritma k-NN ini sangat dipengaruhi oleh ada atau tidaknya fitur-fitur yang tidak relevan, atau jika bobot fitur tersebut tidak setara dengan relevansinya terhadap klasifikasi. Riset terhadap algoritma ini sebagian besar membahas bagaimana memilih dan memberi bobot terhadap fitur, agar performa klasifikasi menjadi lebih baik.

K buah data learning terdekat akan melakukan voting untuk menentukan label mayoritas. Label data query akan ditentukan berdasarkan label mayoritas dan jika ada lebih dari satu label mayoritas maka label data query dapat dipilih secara acak di antara label-label mayoritas yang ada.

Adapun algoritma dari KNN ditunjukan pada flowchart berikut:



Gambar 2.1 Flowchart KNN

2.3. Particle Swarm Optimization

Particle Swarm Optimization (PSO) didasarkan pada perilaku sekawanan burung atau ikan. Algoritma PSO meniru perilaku sosial organisme ini. Perilaku sosial terdiri dari tindakan individu dan pengaruh dari individu-individu lain dalam suatu kelompok. Kata partikel menunjukkan, misalnya, seekor burung dalam kawanan burung. Setiap individu atau partikel berperilaku dengan cara menggunakan kecerdasannya (intelligence) sendiri dan juga dipengaruhi perilaku kelompok kolektifnya. Dengan demikian, jika satu partikel atau seekor burung menemukan jalan yang tepat atau pendek menuju ke sumber makanan, sisa kelompok yang lain juga akan dapat

segera mengikuti jalan tersebut meskipun lokasi mereka jauh di kelompok tersebut.

Dalam Particle Swarm Optimization (PSO), kawanan diasumsikan mempunyai ukuran tertentu dengan setiap partikel posisi awalnya terletak di suatu lokasi yang acak dalam ruang multidimensi. Setiap partikel diasumsikan memiliki dua karakteristik: posisi dan kecepatan. Setiap partikel bergerak dalam ruang/space tertentu dan mengingat posisi terbaik yang pernah dilalui atau ditemukan terhadap sumber makanan atau nilai fungsi objektif. Setiap partikel menyampaikan informasi atau posisi terbaiknya kepada partikel yang lain dan menyesuaikan posisi dan kecepatan masing-masing berdasarkan informasi yang diterima mengenai posisi tersebut. Sebagai contoh, misalnya perilaku burung-burung dalam dalam kawanan burung. Meskipun setiap burung mempunyai keterbatasan dalam hal kecerdasan, biasanya ia akan mengikuti kebiasaan (rule) seperti berikut :

1. Seekor burung tidak berada terlalu dekat dengan burung yang lain
2. Burung tersebut akan mengarahkan terbangnya ke arah rata-rata keseluruhan burung
3. Akan memposisikan diri dengan rata-rata posisi burung yang lain dengan menjaga sehingga jarak antar burung dalam kawanan itu tidak terlalu jauh

Dengan demikian perilaku kawanan burung akan didasarkan pada kombinasi dari 3 faktor simpel berikut:

1. Kohesi - terbang bersama
2. Separasi - jangan terlalu dekat
3. Penyesuaian (alignment) - mengikuti arah bersama

Jadi PSO dikembangkan dengan berdasarkan pada model berikut:

1. Ketika seekor burung mendekati target atau makanan (atau bisa minimum atau maximum suatu fungsi tujuan) secara cepat mengirim informasi kepada burung-burung yang lain dalam kawanan tertentu
2. Burung yang lain akan mengikuti arah menuju ke makanan tetapi tidak secara langsung
3. Ada komponen yang tergantung pada pikiran setiap burung, yaitu memorinya tentang apa yang sudah dilewati pada waktu sebelumnya.

Pada algoritma PSO ini, pencarian solusi dilakukan oleh suatu populasi yang terdiri dari beberapa partikel. Populasi dibangkitkan secara acak dengan batasan nilai terkecil dan terbesar. Setiap partikel merepresentasikan posisi atau solusi dari permasalahan yang dihadapi. Setiap partikel melakukan pencarian solusi yang optimal dengan melintasi ruang pencarian (search space). Hal ini dilakukan dengan cara setiap partikel melakukan penyesuaian terhadap posisi terbaik dari partikel tersebut (local best) dan penyesuaian terhadap posisi partikel terbaik dari seluruh kawanan (global best) selama melintasi ruang pencarian. Jadi, penyebaran pengalaman atau informasi terjadi di dalam partikel itu sendiri dan antara suatu partikel dengan partikel terbaik dari seluruh kawanan selama proses pencarian solusi. Setelah itu, dilakukan proses pencarian untuk mencari posisi terbaik setiap partikel dalam sejumlah iterasi tertentu sampai didapatkan posisi yang relatif stabil atau mencapai batas iterasi yang telah ditetapkan. Pada setiap iterasi, setiap solusi yang direpresentasikan oleh posisi partikel, dievaluasi performansinya dengan cara memasukkan solusi tersebut kedalam fitness function.

Setiap partikel diperlakukan seperti sebuah titik pada suatu dimensi ruang tertentu. Kemudian terdapat dua faktor yang memberikan karakter terhadap status partikel pada ruang pencarian yaitu posisi partikel dan kecepatan partikel.

Berikut ini merupakan formulasi matematika yang menggambarkan posisi dan kecepatan partikel pada suatu dimensi ruang tertentu :

$$X_i\,(t) = x_{i1}(t), x_{i2}(t), \dots, x_{iN}(t) \quad (2.3)$$
$$V_i\,(t) = v_{i1}(t), v_{i2}(t), \dots, v_{iN}(t) \quad (2.4)$$

Dimana
X = posisi partikel
V = kecepatan partikel
i = indeks partikel
t = iterasi ke-t
N = ukuran dimensi ruang

Berikut ini merupakan model matematika yang menggambarkan mekanisme updating status partikel Kennedy and Eberhart [1995]:

$$V_i(t) = V_i(t\text{-}1) + c_1 r_1\left(X_i^L - X_i(\text{t-}1)\right) + C_2 R_2\left(X^G - X_I(\text{t-}1)\right) \quad (2.5)$$
$$X(t) = V_i(t) + X_i(t\text{-}1) \quad (2.6)$$

dimana
$X_i^L = X_{i1}^L, X_{i2}^L, \dots, X_{iN}^L$ merepresentasikan local best dari partikel ke-i. Sedangkan $X^G = X_{i1}^G, X_{i2}^G, \dots, X_{iN}^G$ merepresentasikan global best dari seluruh kawanan. Sedangkan c_1 dan c_2 adalah suatu konstanta yang bernilai positif yang biasanya disebut sebagai learning factor. Kemudian r_1 dan r_2 adalah suatu bilangan

random yang bernilai antara 0 sampai 1. Persamaan (2.5) digunakan untuk menghitung kecepatan partikel yang baru berdasarkan kecepatan sebelumnya, jarak antara posisi saat ini dengan posisi terbaik partikel (local best), dan jarak antara posisi saat ini dengan posisi terbaik kawanan (global best). Kemudian partikel terbang menuju posisi yang baru berdasarkan persamaan (2.6). Setelah algoritma PSO ini dijalankan dengan sejumlah iterasi tertentu hingga mencapai kriteria pemberhentian, maka akan didapatkan solusi yang terletak pada global best.

Model ini akan disimulasikan dalam ruang dengan dimensi tertentu dengan sejumlah iterasi sehingga di setiap iterasi, posisi partikel akan semakin mengarah ke target yang dituju (minimasi atau maksimasi fungsi). Ini dilakukan hingga maksimum iterasi dicapai atau bisa juga digunakan kriteria penghentian yang lain.

Algoritma PSO meliputi langkah berikut

- Bangkitkan posisi awal sejumlah partikel sekaligus kecepatan awalnya secara random.
- Evaluasi fitness dari masing-masing partikel berdasarkan posisinya.
- Tentukan partikel dengan fitness terbaik, dan tetapkan sebagai Gbest.

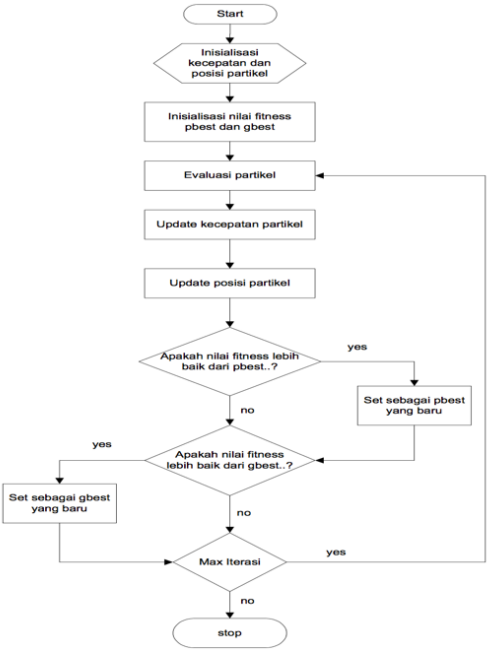
Untuk setiap partikel, Pbest awal akan sama dengan posisi awal.
Ulangi langkah berikut sampai stopping criteria dipenuhi

1. Menggunakan Pbest dan Gbest yang ada, perbarui kecepatan setiap partikel menggunakan persamaan 2.5. Lalu dengan kecepatan baru yang didapat, perbarui posisi setiap partikel menggunakan persamaan 2.6.
2. Evaluasi fitness dari setiap partikel.
3. Tentukan partikel dengan fitness terbaik, dan tetapkan sebagai Gbest. Untuk setiap partikel, tentukan Pbest

dengan membandingkan posisi sekarang dengan Pbest dari iterasi sebelumnya.

4. Cek stopping criteria. Jika dipenuhi, berhenti. Jika tidak, kembali ke 1

Adapun algoritma dari PSO ditunjukan pada flowchart berikut:



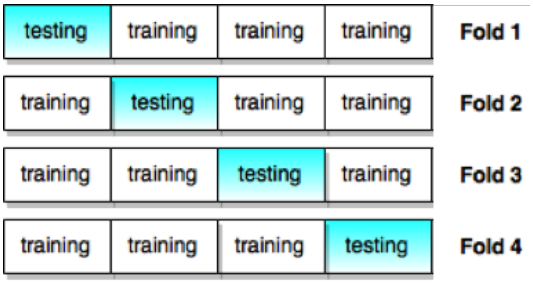
Gambar 2.2 Flowchart PSO

Pada tugas akhir ini, jumlah iterasi yang digunakan sebanyak 15 kali iterasi dan jumlah populasi awal adalah sebanyak 5 partikel.

2.4. Bot Telegram

Telegram adalah sebuah aplikasi pesan instan multiplatform berbasis cloud yang berfokus pada kecepatan dan kemanan proses yang berlangsung. Telegram 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. Pada tugas akhir ini *Bot* Telegram digunakan untuk memberikan informasi secara otomatis ketika terjadi kerusakan pada *BTS*.

2.5. K-Fold Validation



Gambar 2.3 Ilustrasi 4-Fold Validation

Gambar 2.4 Ilustrasi 5-Fold Validation

K-Fold Validation 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 4 dan 5-fold.

[Halaman ini sengaja dikosongkan]

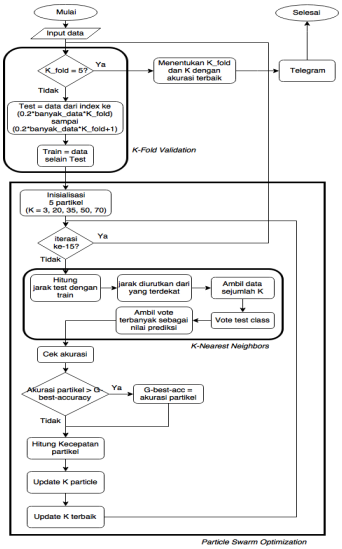
.

BAB III
PERANCANGAN PERANGKAT LUNAK

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

1. Pengimplementasian *Particle Swarm Optimization* untuk mendukung algoritma *K-Nearest Neighbor*. Perhitungan akurasi menggunakan Confusion Matrix yang diikuti dengan pengukuran performa dataset menggunakan algoritma *K-Fold Validation*.
2. 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



Gambar 3.1 Desain Metode Secara Umum

Dimulai dari inputan dataset yang 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 *K-Nearest Neighbor* dan *Particle Swarm Optimization* dalam melakukan klasifikasi sesuai kelas atau prediksi kelas pada setiap data testing. Sebelumnya Particle Swarm Optimization 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. Processing

Pada bagian ini dijelaskan pseudocode dari algoritma yang digunakan yaitu *K-fold Validation*, *Particle Swarm Optimization* (PSO) dan *K-Nearest Neighbor* (KNN). Penerapan keseluruhan algoritma ini menggunakan bahasa pemrograman Python 2.7.

3.1.1.1. 4-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 4 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.

Masukkan	Keseluruhan data, Iterasi ke-n, banyak data
Keluaran	Data *Training* dan *Testing*

```
1.  min = int(0.25 * banyak_data * iterasi ke)
2.  max = int(0.25 * banyak_data * (iterasi ke + 1))
3.  if((banyak_data % 4) != 0):
4.      banyak = int(0.25 * banyak_data) + 1
5.  else:
6.      banyak = int(0.25 * banyak_data)
7.  for i to banyak:
8.       if iterasi ke != 0:
9.            test.append(test_all[i+min])
10.      else:
11.           test.append(test_all[i])
12. for i to banyak_data:
13.      if data[i] not in test:
14.           train.append(data[i])
```

code 3.1 Pseudocode 4-Fold validation

3.1.1.2. 5-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.

Masukkan	Keseluruhan data, Iterasi ke-n, banyak data
Keluaran	Data *Training* dan *Testing*

```
1.  min = int(0.2 * banyak_data * iterasi ke)
2.  max = int(0.2 * banyak_data * (iterasi ke + 1))
```

```
3.  if((banyak_data % 5) != 0):
4.      banyak = int(0.2 * banyak_data) + 1
5.  else:
6.      banyak = int(0.2 * banyak_data)
7.  for i to banyak:
8.       if iterasi ke != 0:
9.            test.append(test_all[i+min])
10.      else:
11.           test.append(test_all[i])
12. for i to banyak_data:
13.      if data[i] not in test:
14.           train.append(data[i])
```

code 3.2 Pseudocode 5-Fold validation

3.1.1.3. Particle Swarm Optimization

Metode PSO terdiri dari 5 tahapan yaitu tahap *generate swarm*, update nilai G-best, update kecepatan, update posisi, evaluasi g-best menggunakan algoritma *K-Nearest Neighbor* (KNN).

Tahap awal adalah generate swarm dimana sejumlah n-swarm diberikan posisi awal, kecepatan, dan akurasi awal. Kecepatan akan di random dengan nilai float dari batas -10 sampai dengan 10 dan posisi awal di tetapkan dengan nilai masing-masing swarm yaitu 3, 20, 35, 50, 70 nilai ini bebas karena hanya sebagai nilai awal dan akan diubah berdasarkan nilai kecepatan dari tiap partikel. Tahap selanjutnya adalah evaluasi menggunakan algoritma *K-Nearest Neighbor* (KNN) yang hasilnya disimpan di nilai akurasi disetiap partikel yang kemudian akan masuk ke tahap update nilai G-best. G-best adalah posisi partikel dengan nilai akurasi tertinggi. Kemudian masuk ke tahap update kecepatan dengan perhitungan seperti persamaan pada (2.5). Selanjutnya masuk ke tahap update posisi dengan perhitungan seperti persamaan pada (2.6). Disinilah nilai K yang digunakan pada algoritma *K-Nearest Neighbor* (KNN) didapatkan, posisi setiap partikel akan menjadi nilai K pada algoritma *K-Nearest Neighbor* (KNN). Proses ini diulang sebanyak 10 kali iterasi dan akan mendapatkan nilai K dengan akurasi terbaik.

3.1.1.4. K-Nearest Neighbor

Fungsi K-Nearest Neighbor 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 *Particle Swarm Optimization* (PSO). 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.

[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 *python 2.7* yang diinstal pada sistem operasi CentOS 7.0.

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 nilai *G-Best* berdasarkan nilai akurasi dari setiap *fold*. Parameter yang digunakan dapat dilihat pada tabel berikut.

Tabel 4.1 Parameter yang digunakan

Parameter PSO	Nilai	Parameter K-NN	Nilai	Parameter K-Fold	Nilai
Posisi per-partikel	3, 20, 35, 50, 70	K (tetangga terdekat)	PSO	Itersi fold ke	Itersi ke-N
Kecepatan per-partikel	Random float dari range -10 sampai 10				
Jumlah Iterasi	15				
Jumlah Partikel	5				

4.2.2. Implementasi Fungsi Telegram

Fungsi telegramchat adalah fungsi untuk mengirimkan pesan notifikasi ke telegram. Fungsi ini memiliki *input* berupa hasil prediksi dari knn.

```
1.  def telegramchat(tipeerror):
2.      if tipeerror == 1:
3.          errornya = "Bad Voice"
4.      elif tipeerror == 2:
5.          errornya = "Low Throughput"
6.      else :
7.          errornya = "Low Coverage"
8.      bot = telegram.Bot('
        502247305:
        AAFO2tGwNBYZuuwhpbOLKxGIGuoImkyxqw')
9.      bot.sendMessage(chat_id = 262825724,
10.             text = errornya)
```

code 4.1 Notifikasi telegram

Fungsi ini akan mengirimkan pesan "*Bad Voice*" apabila hasil prediksi dari knn bernilai 1, "*Low Throughput*" apabila hasil prediksi dari knn bernilai 2, dan "*Low Coverage*" apabila hasil prediksi dari knn bernilai 3.

4.2.3. Implementasi Fungsi Readfile

Fungsi readfile() adalah fungsi yang dibuat untuk membaca suatu file dataset berformat txt. *Input* dari fungsi ini berupa namafile yang akan dibaca dan *output* dari fungsi ini akan menyimpan semua data ke dalam variabel train_all dan test_all.

```
1.  def readfile(namafile):
2.      global train_all, test_all, banyak_data
3.      train_all = []
4.      test_all = []
5.      with codecs.open(namafile, 'rb',
6.        encoding = "utf-8-sig") as data:
7.          semua = data.readlines()
8.          banyak_data = len(semua)
9.          for x in semua:
10.             train_all.append
11.             (x.strip('\n').split(','))
12.             test_all.append
13.             (x.strip('\n').split(','))
```

code 4.2 read file

fungsi ini berfungsi untuk membaca isi file yang kemudian memisahkan data dengan tanda koma " , " dan disimpan ke dalam variable train_all dan test_all yang berbentuk array.

4.2.4. Implementasi Metode 4-Fold Validation

Fungsi vold() adalah fungsi yang dibuat untuk membagi dataset menjadi data train dan data test menjadi 4 bagian yang dibagi menjadi seperti pada ilustrasi Gambar 2.3 Ilustrasi 4-Fold Validation. *Input* dari fungsi ini merupakan iterasi vold dan *output* dari fungsi ini merupakan data train dan test yang sudah siap diolah.

```
1.  def vold(ke):
2.     min = int(0.25 * banyak_data * ke)
3.     max = int(0.25 * banyak_data * (ke + 1))
4.
5.     if ((banyak_data % 4) != 0):
6.
    banyak = int(0.25 * banyak_data) + 1
7.     else :
8.        banyak = int(0.25 * banyak_data)
9.
10.    for i in range(banyak):
11.       if ke != 0:
12.
    test.append(test_all[i + min])
13.       else :
14.          test.append(test_all[i])
15.    for i in range(banyak_data):
16.       if [float(j) for j in train_all[i]]
17.        not in [map(float, j) for j in test]:
18.          train.append(train_all[i])
```

code 4.3 4-Fold

dalam fungsi ini jumlah data akan dibagi 4 dan disusun menjadi bagian *training* dan *testing* seperti pada ilustrasi Gambar 2.3 Ilustrasi 4-Fold Validation yang kemudian disimpan di array *train* dan *test*.

4.2.5. Implementasi Metode 5-Fold Validation

Fungsi vold() adalah fungsi yang dibuat untuk membagi dataset menjadi data train dan data test menjadi 5 bagian yang dibagi menjadi seperti pada ilustrasi Gambar 2.4 Ilustrasi 5-Fold Validation. *Input* dari fungsi ini merupakan iterasi vold dan *output* dari fungsi ini merupakan data train dan test yang sudah siap diolah.

```
1. def vold(ke):
2.     min = int(0.2 * banyak_data * ke)
3.     max = int(0.2 * banyak_data * (ke + 1))
4.
5.     if ((banyak_data % 5) != 0):
6.
   banyak = int(0.2 * banyak_data) + 1
7.     else :
8.        banyak = int(0.2 * banyak_data)
9.
10.          for i in range(banyak):
11.             if ke != 0:
12.                test.append
                   (test_all[i + min])
13.             else :
14.                test.append(test_all[i])
15.
16.          for i in range(banyak_data):
17.
   if [float(j) for j in train_all[i]]
      not in [map(float, j) for j in test]:
18.           train.append(train_all[i])
```

code 4.4 5-fold

dalam fungsi ini jumlah data akan dibagi 5 dan disusun menjadi bagian *training* dan *testing* seperti pada ilustrasi

Gambar 2.4 Ilustrasi 5-Fold Validation yang kemudian disimpan di array *train* dan *test.*

4.2.6. Implementasi Metode Particle Swarm Optimization

4.2.6.1. Inisialisasi partikel

Dalam fungsi ini partikel-partikel diinisialisasi, nilai posisi partikel diisi dengan posisi awal yang sudah ditentukan penulis yaitu 3, 20, 35, 50, 70 dan kecepatan awal partikel di *random* dengan tipe *float* dari nilai negatif 10 sampai dengan 10.

```
1. class Particle:
2. def __init__(self, x0):
3.    self.position_i = []
4.    self.velocity_i = []
5.    self.pos_best_i = []
6.    self.acc_best_i = 1
7.    self.acc_i = -1
8.    for i in range(0, num_dimensions):
9.
  self.velocity_i.append(random.uniform(-
  10, 10))
10.           self.position_i.append(x0[i])
```

code 4.5 Inisialisiasi partikel

4.2.6.2. Evaluasi akurasi terbaik

Dalam fungsi ini nilai akurasi akan dicek apakah nilai akurasi yang didapat dari *KNN* lebih baik dari nilai akurasi sekarang, jika iya nilai akan digantikan dengan nilai akurasi baru dan posisi juga digantikan dengan posisi baru.

```
1. def evaluate(self, costFunc, j):
2.    self.acc_i = costFunc
      (int(self.position_i[j]))
3.    if self.acc_i > self.acc_best_i
      or self.acc_best_i == -1:
4.       self.pos_best_i = self.position_i
5.     self.acc_best_i = self.acc_i
```

code 4.6 Evaluasi

4.2.6.3. Update kecepatan

Dalam fungsi ini kecepatan setiap partikel akan diganti dengan nilai dari hasil perhitungan persamaan matematika pada rumus 2.5 dan 2.6.

```
1. def update_velocity(self, pos_best_g):
2.    w = 0.729
3.    c1 = 1
4.    c2 = 2
5.    for i in range(0, num_dimensions):
6.       r1 = random.random()
7.       r2 = random.random()
8.       vel_cognitive = c1 * r1 *

   (self.pos_best_i[i] - self.position_i[i])
9.       vel_social = c2 * r2 *
          (pos_best_g[i] - self.position_i[i])
10.
   self.velocity_i[i] = w * self.velocity_i[i]
   + vel_cognitive + vel_social
```

code 4.7 Update kecepatan

4.2.6.4. Update posisi

Dalam fungsi ini posisi partikel akan bergerak sesuai dengan nilai kecepatan partikel yang dimiliki namun tidak dapat melebihi batas maksimal yaitu 100 dan batas minimal yaitu 0.

```
1. def update_position(self, bounds):
2. for i in range(0, num_dimensions):
3. self.position_i[i] = self.position_i[i] + se
   lf.velocity_i[i]
4. if self.position_i[i] > bounds[i][1]:
5. self.position_i[i] = bounds[i][1]
6. if self.position_i[i] < bounds[i][0]:
7. self.position_i[i] = bounds[i][0]
```

code 4.8 Update posisi

4.2.7. Implementasi Metode K-Nearest Neighbor

Pada fungsi ini kita akan mendapatkan jarak dari nilai data test dengan nilai data train setiap kolomnya. Disini penulis menggunakan 3 penghitungan jarak yang berbeda jarak *euclidean, manhattan,* dan *chebyshev*. *Input* dari fungsi ini ada 3 yaitu data tes, data train dan panjang kolom. *Output* dari fungsi ini adalah jarak dari nilai data *test* dengan nilai data *train.*

4.2.7.1. Euclidean distance

```
1. def euclideanDistance(x1, x2, length):
2. distance = 0
3. for x in range(length):
4. x1[x] = float(x1[x])
5. x2[x] = float(x2[x])
6. distance += pow((x1[x] - x2[x]),2)
```

code 4.9 euclidean distance

4.2.7.2. Manhattan distance

```
1. def euclideanDistance(x1, x2, length):
2. distance = 0
3. for x in range(length):
4. x1[x] = float(x1[x])
5. x2[x] = float(x2[x])
6. beda = x1[x] - x2[x]
7. if (beda < 0):
8. beda = beda * -1
9. distance += beda
```

code 4.10 manhattan distance

4.2.7.3. Chebychev distance

```
1. def euclideanDistance(x1, x2, length):
2. distance = 0
3. for x in range(length):
4. x1[x] = float(x1[x])
5. x2[x] = float(x2[x])
6. beda = x1[x] - x2[x]
7. if (beda < 0):
8. beda = beda * -1
9. if beda > distance:
10. distance = beda
```

code 4.11 chebyshev distance

4.2.7.4. Nilai tetangga

Pada fungsi ini kita akan mendapatkan tetangga-tetangga terdekat yang jaraknya sudah diurutkan sebanyak nilai K. *Input* fungsi ini merupakan data *test* dan data *train* yang nantinya akan

masuk ke fungsi euclidean. *Output* dari fungsi ini adalah K tetangga terdekat.

```
1. def gettetangga(trainset, test, k):
2. jarak = []
3. length = len(test) - 1
4. for x in range(len(trainset)):
5. dist = euclideanDistance(test, trainset[x],
    length)  jarak.append((trainset[x], dist))
6. jarak.sort(key = operator.itemgetter(1))
7. tetangga = []
8. for x in range(k):
9. tetangga.append(jarak[x][0])
10. return tetangga
```

code 4.12 gettetangga

4.2.7.5. Getrespon

Pada fungsi ini tetangga-tetangga yang sudah didapatkan akan diambil kelasnya dan dilakukan pemilihan kelas mana yang akan diambil sebagai hasil prediksi akhir. *Input* dari fungsi ini adalah sejumlah k-tetangga. *Output* dari fungsi ini adalah prediksi nilai.

```
1. def getrespon(neighbors):
2. classVotes = {}
3. for x in range(len(neighbors)):
4. response = neighbors[x][-1]
5. if response in classVotes:
6. classVotes[response] += 1
7. else :
8. classVotes[response] = 1
9. sortedVotes = sorted(classVotes.iteritems(), key = ope
    rator.itemgetter(1), reverse = True)
10. return sortedVotes[0][0]
```

code 4.13 Getrespon

BAB V
PENGUJIAN 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 yang akan digunakan adalah,

1. Perangkat Keras
   Prosesor: Intel® Core™ i7-4770 @ 3.40 GHz
   Memori: 32 GB.
   Sistem Operasi: 64-bit .
2. Perangkat Lunak
   Sistem Operasi: CentOS 7.0
   Perangkat Pengembang: Python2.7.

5.2. Data Uji Coba

Tahap uji coba pada tugas akhir ini menggunakan 1 dataset dari salah satu perusahaan Telekomunikasi di Indonesia dan 8 dataset dari *UCI Machine Learning*. Data yang terpilih kemudian akan dibagi menjadi 2 yaitu data training dan data testing. Pembagian data dalam tahap ini menggunakan metode 4-*fold validation* dan 5-*fold validation*, dimana data keseluruhan dibagi menjadi 4 dan 5 bagian.

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 4-*foldvalidation* dan 5-*foldvalidation*. Setiap kelompok terdiri dari 3 skenario pengujian seperti berikut:

1. Skenario perhitungan akurasi menggunakan metode gabungan *Particle Swarm Optimization* dan *K-Nearest Neighbor* dengan rumus perhitungan jarak *Euclidean Distance*.
2. Skenario perhitungan akurasi menggunakan metode gabungan *Particle Swarm Optimization* dan *K-Nearest Neighbor* dengan rumus perhitungan jarak *Manhattan Distance*.
3. Skenario perhitungan akurasi menggunakan metode gabungan *Particle Swarm Optimization* dan *K-Nearest Neighbor* dengan rumus perhitungan jarak *Chebyshev Distance*.

Skenario perhitungan akurasi menggunakan metode gabungan *Particle Swarm Optimization* dan *K-Nearest Neighbor* dengan rumus perhitungan jarak *Chebyshev Distance*.

5.4. Analisis Hasil Uji Coba

Berikut merupakan tabel hasil kenario perhitungan akurasi menggunakan metode gabungan *Particle Swarm Optimization* dan *K-Nearest Neighbor* dengan K-fold 4 dan K-fold 5.

5.4.1. Hasil dengan menggunakan 4-Fold

Tabel 5.1 Tabel akurasi 4 fold

Dataset	PSO-KNN (Euclidean)	PSO-KNN (Manhattan)	PSO-KNN (Chebyshev)
WHOLESALE	95.45	95.45	95.45
K	24	70	70
Fold-Ke	2	2	2
TELKOMSEL	95.68	93.53	94.24
K	1	42	1
Fold-Ke	4	4	4
BLOOD	89.33	89.33	89.33
K	3	3	3
Fold-Ke	3	3	3
PERFUME	60.00	60.00	60.00
K	1	1	1
Fold-Ke	4	4	4
WINE	82.22	91.11	77.78
K	1	2	2
Fold-Ke	1	1	1
IRIS	100.00	100.00	100.00
K	3	3	3
Fold-Ke	1	1	1
USER	88.12	92.08	79.21
K	3	3	5
Fold-Ke	4	4	2
DIABETIC	87.17	87.17	87.17
K	35	35	39
Fold-Ke	4	4	4
SPECTF	92.54	91.04	95.52
K	28	61	67
Fold-Ke	3	3	3

Tabel 5.2 Tabel detail akurasi 4 fold

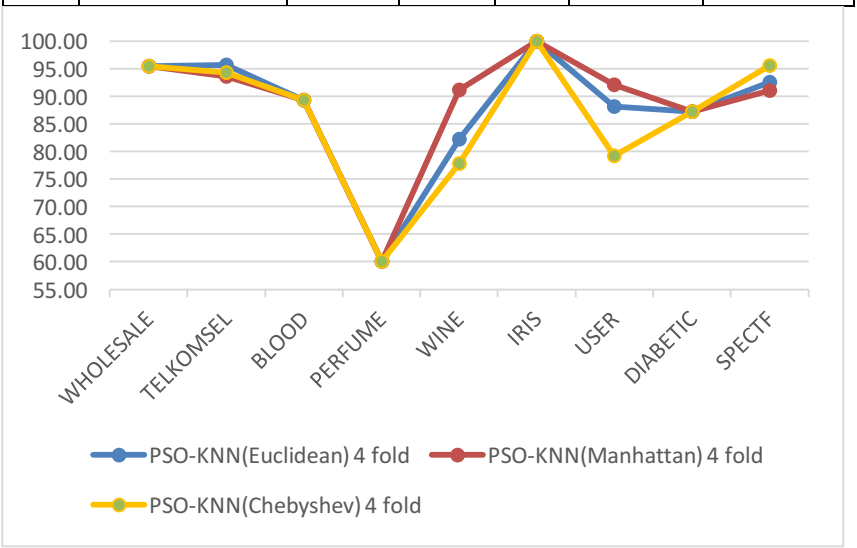
No		Train	Test	K	Fold-Ke	Akurasi
1	WHOLESALE					
	PSO-KNN(Euclidean)	330	110	3	1	87.27
				24	2	95.45
				3	3	90.91
				3	4	89.09
	PSO-KNN(Manhattan)			20	1	87.27
				70	2	95.45
				20	3	93.64
				11	4	90.00
	PSO-KNN(Chebyshev)			4	1	87.27
				70	2	95.45
				16	3	90.91
				1	4	89.09
2	TELKOMSEL					
	PSO-KNN(Euclidean)	417	139	81	1	65.47
				8	2	89.21
				3	3	86.33
				1	4	95.68
	PSO-KNN(Manhattan)			3	1	59.71
				42	2	93.53
				32	3	91.37
				1	4	91.37
	PSO-KNN(Chebyshev)			54	1	69.06
				7	2	86.33
				1	3	86.33
				1	4	94.24

No		Train	Test	K	Fold-ke	Akurasi
3	BLOOD					
	PSO-KNN(Euclidean)	561	187	3	1	60.67
				50	2	78.67
				50	3	89.33
				5	4	69.33
	PSO-KNN(Manhattan)			3	1	61.33
				35	2	78.67
				50	3	89.33
				14	4	68.67
	PSO-KNN(Chebyshev)			3	1	60.00
				35	2	78.67
				50	3	89.33
				3	4	67.33
4	PERFUME					
	PSO-KNN(Euclidean)	30	10	1	1	60.00
				1	2	50.00
				1	3	60.00
				1	4	50.00
	PSO-KNN(Manhattan)			1	1	60.00
				1	2	50.00
				1	3	60.00
				1	4	50.00
	PSO-KNN(Chebyshev)			1	1	50.00
				1	2	50.00
				1	3	60.00
				1	4	50.00

No		Train	Test	K	Fold-ke	Akurasi
5	WINE					
	PSO-KNN(Euclidean)	133	45	3	1	71.11
				1	2	71.11
				1	3	82.22
				3	4	2.22
	PSO-KNN(Manhattan)			16	1	75.56
				3	2	75.56
				2	3	91.11
				3	4	2.22
	PSO-KNN(Chebyshev)			3	1	71.11
				2	2	77.78
				1	3	77.78
				1	4	4.44
6	IRIS					
	PSO-KNN(Euclidean)	112	38	3	1	100.00
				1	2	94.74
				1	3	94.74
				2	4	73.68
	PSO-KNN(Manhattan)			3	1	100.00
				3	2	94.74
				1	3	94.74
				3	4	63.16
	PSO-KNN(Chebyshev)			3	1	100.00
				3	2	94.74
				1	3	97.37
				3	4	71.05

No		Train	Test	K	Fold-ke	Akurasi
7	USER					
	PSO-KNN(Euclidean)	302	101	3	1	81.19
				3	2	86.14
				1	3	84.16
				3	4	88.12
	PSO-KNN(Manhattan)			20	1	86.14
				8	2	89.11
				13	3	91.09
				3	4	92.08
	PSO-KNN(Chebyshev)			1	1	75.25
				5	2	78.22
				6	3	74.26
				3	4	77.23
8	DIABETIC					
	PSO-KNN(Euclidean)	561	187	21	1	62.57
				20	2	81.82
				20	3	80.21
				35	4	87.17
	PSO-KNN(Manhattan)			3	1	63.64
				35	2	81.82
				21	3	79.68
				35	4	87.17
	PSO-KNN(Chebyshev)			16	1	61.50
				50	2	81.82
				6	3	81.82
				39	4	87.17

No		Train	Test	K	Fold-ke	Akurasi
9	SPECTF					
	PSO-KNN(Euclidean)	200	67	3	1	68.66
				1	2	82.09
				28	3	92.54
				28	4	80.60
	PSO-KNN(Manhattan)			3	1	68.66
				20	2	80.60
				61	3	91.04
				34	4	80.60
	PSO-KNN(Chebyshev)			9	1	62.69
				13	2	83.58
				67	3	95.52
				35	4	80.60



Gambar 5.1 Grafik akurasi 4 fold

5.4.2. Hasil dengan menggunakan 5-Fold

Tabel 5.3 Tabel akurasi 5 fold

Dataset	PSO-KNN (Euclidean)	PSO-KNN (Manhattan)	PSO-KNN (Chebyshev)
WHOLESALE	97.73	97.73	96.59
K	34	35	35
Fold-Ke	2	2	2
TELKOMSEL	95.54	96.43	96.43
K	20	20	1
Fold-Ke	4	4	5
BLOOD	90.00	90.00	90.00
K	35	35	50
Fold-Ke	5	5	5
PERFUME	25.00	25.00	25.00
K	1	1	1
Fold-Ke	4	4	4
WINE	83.33	86.11	86.11
K	3	3	3
Fold-Ke	1	1	1
IRIS	100	100	100
K	3	3	3
Fold-Ke	1	1	1
USER	92.59	96.30	82.72
K	8	8	4
Fold-Ke	3	3	1
DIABETIC	90	90	90.67
K	35	35	50
Fold-Ke	5	5	5
SPECTF	92.59	100.00	94.44
K	4	91	77
Fold-Ke	3	4	3

Tabel 5.4 Tabel detail akurasi 5 fold

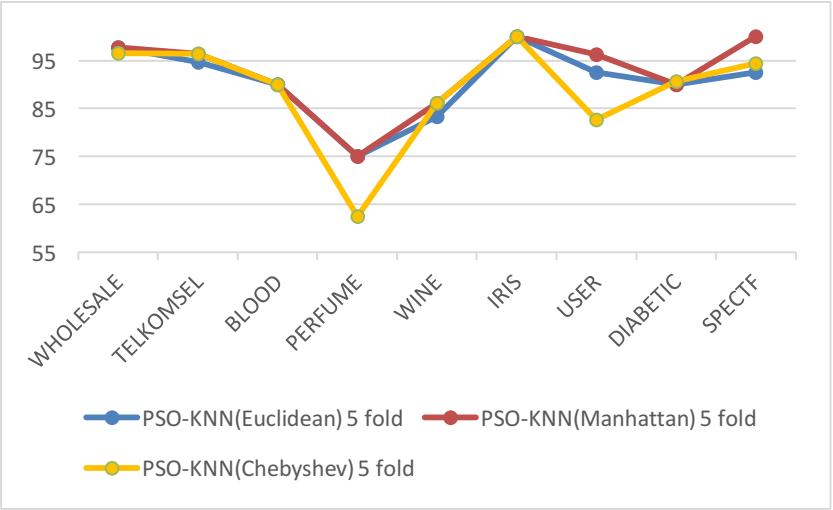
No		Train	Test	K	Fold	Akurasi
1	WHOLESALE					
	PSO-KNN(Euclidean)	352	88	67	1	83.93
				34	2	97.73
				11	3	84.82
				22	4	95.54
				3	5	83.93
	PSO-KNN(Manhattan)			20	1	86.36
				35	2	97.73
				20	3	89.77
				13	4	96.59
				20	5	88.64
	PSO-KNN(Chebyshev)			2	1	86.36
				35	2	96.59
				70	3	88.64
				3	4	94.32
				20	5	87.50
2	TELKOMSEL					
	PSO-KNN(Euclidean)	444	112	64	1	83.04
				62	2	88.39
				18	3	82.14
				22	4	95.54
				1	5	92.86
	PSO-KNN(Manhattan)			3	1	66.07
				70	2	88.39
				53	3	90.18
				20	4	96.43
				1	5	88.39
	PSO-KNN(Chebyshev)			50	1	84.82
				50	2	82.14
				5	3	82.14

No		Train	Test	K	Fold	Akurasi
				1	4	93.75
				1	5	96.43
3	BLOOD					
	PSO-KNN(Euclidean)	598	150	3	1	60.67
				50	2	78.67
				50	3	89.33
				5	4	69.33
				35	5	90.00
	PSO-KNN(Manhattan)			3	1	61.33
				35	2	78.67
				50	3	89.33
				14	4	68.67
				35	5	90.00
	PSO-KNN(Chebyshev)			3	1	60.00
				35	2	78.67
				50	3	89.33
				3	4	67.33
				50	5	90.00
4	PERFUME					
	PSO-KNN(Euclidean)	32	8	1	1	62.5
				1	2	75
				1	3	37.5
				1	4	62.5
				1	5	37.5
	PSO-KNN(Manhattan)			1	1	62.5
				1	2	75
				1	3	37.5
				1	4	62.5
				1	5	37.5
	PSO-KNN(Chebyshev)			1	1	50
				1	2	62.5

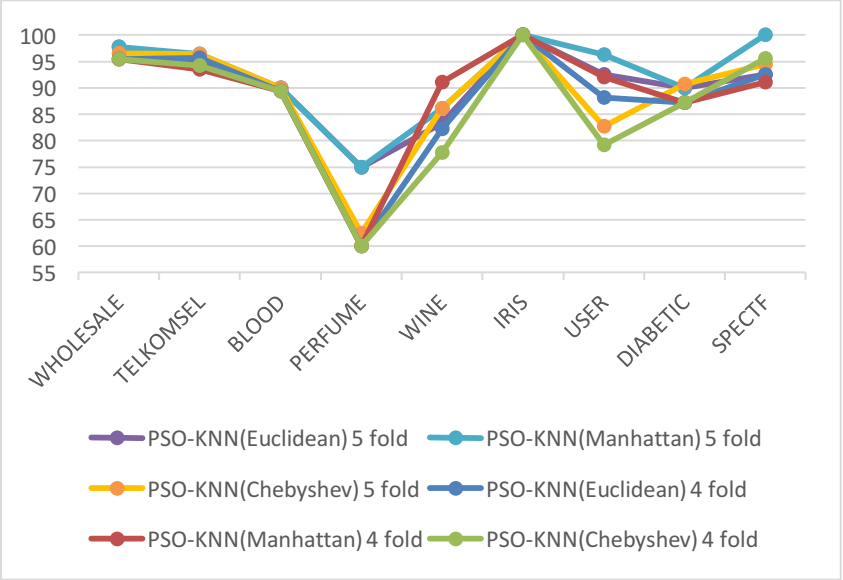
No		Train	Test	K	Fold	Akurasi
4	PSO-KNN(Chebyshev)	32	8	1	3	50
				1	4	50
				1	5	50
5	WINE					
	PSO-KNN(Euclidean)	142	36	3	1	83.33
				1	2	80.56
				3	3	58.33
				1	4	83.33
				1	5	19.44
	PSO-KNN(Manhattan)			3	1	86.11
				1	2	86.11
				2	3	77.78
				17	4	86.11
				11	5	8.33
	PSO-KNN(Chebyshev)			3	1	86.11
				1	2	80.56
				3	3	58.33
				24	4	83.33
				3	5	5.56
6	IRIS					
	PSO-KNN(Euclidean)	120	30	3	1	100.0
				3	2	100.0
				13	3	86.67
				3	4	93.33
				1	5	80.00
	PSO-KNN(Manhattan)			3	1	100.0
				3	2	100.0
				3	3	86.67
				14	4	96.67
				9	5	80.00

No		Train	Test	K	Fold	Akurasi
6	PSO-KNN(Chebyshev)	120	30	3	1	100.0
				3	2	100.0
				1	3	90.00
				14	4	96.67
				2	5	80.00
7	USER					
	PSO-KNN(Euclidean)	322	81	12	1	83.95
				12	2	88.89
				8	3	92.59
				1	4	83.95
				3	5	88.89
	PSO-KNN(Manhattan)			20	1	85.19
				9	2	92.59
				8	3	96.30
				10	4	85.19
				3	5	93.83
	PSO-KNN(Chebyshev)			4	1	82.72
				3	2	81.48
				11	3	79.01
				2	4	80.25
				6	5	80.25
8	DIABETIC					
	PSO-KNN(Euclidean)	598	150	12	1	59.33
				23	2	79.33
				20	3	89.33
				20	4	73.33
				35	5	90.00
	PSO-KNN(Manhattan)			3	1	60.00
				35	2	79.33
				20	3	89.33
				20	4	73.33

No		Train	Test	K	Fold	Akurasi
8				35	5	90.00
	PSO-KNN(Chebyshev)			8	1	57.33
				20	2	79.33
				20	3	89.33
				24	4	73.33
				50	5	90.67
9	SPECTF					
	PSO-KNN(Euclidean)	213	54	35	1	77.78
				1	2	64.81
				4	3	92.59
				85	4	92.59
				13	5	77.78
	PSO-KNN(Manhattan)			3	1	79.63
				1	2	61.11
				62	3	90.74
				91	4	100.0
				3	5	77.78
	PSO-KNN(Chebyshev)			25	1	81.48
				1	2	61.11
				77	3	94.44
				57	4	94.44
				20	5	72.22



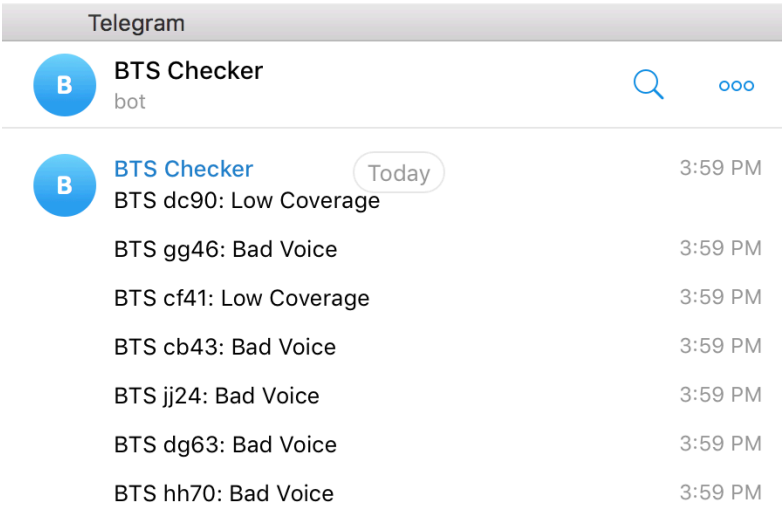
Gambar 5.2 Grafik akurasi 5 fold



Gambar 5.3 Grafik akurasi 4 fold dan 5 fold

5.4.3. Hasil notifikasi BOT Telegram

Seperti yang dapat kita lihat pada Gambar 5.4 Notifikasi bot Telegram akan muncul pesan secara otomatis yang akan memberi informasi jenis masalah pada *BTS*.



Gambar 5.4 Notifikasi bot Telegram

BAB VI
KESIMPULAN DAN SARAN

Pada bab ini akan dijelaskan mengenai kesimpulan dari proses dan uji coba dari program dan saran untuk pengembangan dari program itu sendiri.

6.1. Kesimpulan

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

1. Adanya implementasi PSO dalam mendukung algoritma *K-Nearest Neighbor* 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. Dari 3 cara perhitungan jarak yang digunakan dalam algoritma *KNN* yang terbaik adalah *Manhattan Distance* karena dari 9 dataset, 8 diantaranya memiliki nilai akurasi terbesar atau sama besar.

6.2. Saran

Saran yang diberikan untuk pengembangan aplikasi ini adalah:

1. Memperbaiki algoritma saat pemilihan data, karena apabila data yang dimasukkan berjumlah besar aplikasi ini akan berjalan sangat lama untuk melakukan pemilihan data *training* dan *testing.*
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".

BIODATA PENULIS

Yusuf Dimas Hermawan, lahir pada tanggal 12 Agustus 1996 di Surabaya. Penulis merupakan seorang mahasiswa yang sedang menempuh studi di Jurusan Teknik Informatika Institut Teknologi Sepuluh Nopember (ITS). Selama menempuh pendidikan di kampus, penulis juga aktif dalam organisasi kemahasiswaan, antara lain Staff Kewirausahaan Himpunan Mahasiswa Teknik Computer-Informatika (HMTC), Staff National Programming Contest (NPC) Schematics ITS.