W

;/--n

**TUGAS AKHIR – KI1502**

**Implementasi Model *Deterministic Finite Automaton* untuk Interpretasi *Regular Expression* pada Studi Kasus Permasalahan SPOJ Klasik 10354**

**MUHAMMAD YUNUS BAHARI**

**NRP 5111100079**

**Dosen Pembimbing I**

**Arya Yudhi Wijaya, S.Kom., M.Kom.**

**Dosen Pembimbing II**

**Rully Soelaiman, S.Kom., M.Kom.**

**JURUSAN TEKNIK INFORMATIKA**

**Fakultas Teknologi Informasi**

**Institut Teknologi Sepuluh Nopember**

**Surabaya 2015**

***[Halaman ini sengaja dikosongkan]***

**TUGAS AKHIR – KI1502**

**Implementasi Model *Deterministic Finite Automaton* untuk Interpretasi *Regular Expression* pada Studi Kasus Permasalahan SPOJ Klasik 10354**

**MUHAMMAD YUNUS BAHARI**

**NRP 5111100079**

**Dosen Pembimbing I**

**Arya Yudhi Wijaya, S.Kom., M.Kom.**

**Dosen Pembimbing II**

**Rully Soelaiman, S.Kom., M.Kom.**

**JURUSAN TEKNIK INFORMATIKA**

**Fakultas Teknologi Informasi**

**Institut Teknologi Sepuluh Nopember**

**Surabaya 2015**

****

***[Halaman ini sengaja dikosongkan]***

**UNDERGRADUATE THESES – KI1502**

**IMPLEMENTATION OF DETERMINISTIC FINITE AUTOMATON MODEL FOR REGULAR EXPRESSION INTERPRETATION A CASE STUDY AT SPOJ CLASSIC PROBLEM 10354**

**MUHAMMAD YUNUS BAHARI**

**NRP 5111100079**

**Supervisor I**

**Arya Yudhi Wijaya, S.Kom., M.Kom.**

**Supervisor II**

**Rully Soelaiman, S.Kom., M.Kom.**

**DEPARTMENT OF INFORMATICS**

**FACULTY OF INFORMATION TECHNOLOGY**

**INSTITUT TEKNOLOGI SEPULUH NOPEMBER**

**SURABAYA 2015**

# LEMBAR PENGESAHAN

**IMPLEMENTASI MODEL *DETERMINISTIC FINITE AUTOMATON* UNTUK INTERPRETASI *REGULAR EXPRESSION* PADA STUDI KASUS PERMASALAHAN SPOJ KLASIK 10354**

**TUGAS AKHIR**

Diajukan Untuk Memenuhi Salah Satu Syarat

Memperoleh Gelar Sarjana Komputer

pada

Bidang Studi Desain dan Terapan Komputasi

Program Studi S-1 Jurusan Teknik Informatika

Fakultas Teknologi Informasi

Institut Teknologi Sepuluh Nopember

Oleh

**MUHAMMAD YUNUS BAHARI**

**NRP : 5111 100 079**

Disetujui oleh Dosen Pembimbing Tugas Akhir:

1. Arya Yudhi Wijaya, S.Kom., M.Kom. .....................

NIP: 198409042010121002 (Pembimbing 1)

1. Rully Soelaiman, S.Kom., M.Kom. .....................

NIP: 197002131994021001 (Pembimbing 2)

**SURABAYA**

**MEI, 2015**

***[Halaman ini sengaja dikosongkan]***

**Implementasi Model *Deterministic Finite Automaton* untuk Interpretasi *Regular Expression* pada Studi Kasus Permasalahan SPOJ Klasik 10354**

|  |  |  |
| --- | --- | --- |
| **Nama Mahasiswa** | **:** | **MUHAMMAD YUNUS BAHARI** |
| **NRP** | **:** | **5111100079** |
| **Jurusan** | **:** | **Teknik Informatika FTIF-ITS** |
| **Dosen Pembimbing 1** | **:** | **Arya Yudhi Wijaya, S.Kom., M.Kom.** |
| **Dosen Pembimbing 2** | **:** | **Rully Soelaiman, S.Kom., M.Kom.** |

# *Abstrak*

*Dewasa ini, …*

***Kata kunci: …***

**IMPLEMENTATION OF DETERMINISTIC FINITE AUTOMATON MODEL FOR REGULAR EXPRESSION INTERPRETATION A CASE STUDY AT SPOJ CLASSIC PROBLEM 10354**

|  |  |  |
| --- | --- | --- |
| **Student’s Name** | **:** | **MUHAMMAD YUNUS BAHARI** |
| **Student’s ID** | **:** | **5111100079** |
| **Department** | **:** | **Teknik Informatika FTIF-ITS** |
| **First Advisor** | **:** | **Arya Yudhi Wijaya, S.Kom., M.Kom.** |
| **Second Advisor** | **:** | **Rully Soelaiman, S.Kom., M.Kom.** |

# *Abstract*

*Nowadays , …*

***Keywords: …***

# KATA PENGANTAR



Alhamdulillahirabbil’alamin, segala puji bagi Allah SWT, yang telah melimpahkan rahmat dan hidayah-Nya sehingga penulis dapat menyelesaikan Tugas Akhir yang berjudul ***“*IMPLEMENTASI METODE *GENETIC ALGORITHM, K-MEANS, DAN K-NEAREST NEIGHBOR* UNTUK KLASIFIKASI PENYAKIT KANKER *COLON”*.**

Pengerjaan Tugas Akhir ini merupakan suatu kesempatan yang sangat baik bagi penulis. Dengan pengerjaan Tugas Akhir ini, penulis bisa belajar lebih banyak untuk memperdalam dan meningkatkan apa yang telah didapatkan penulis selama menempuh perkuliahan di Teknik Informatika ITS. Dengan Tugas Akhir ini penulis juga dapat menghasilkan suatu implementasi dari apa yang telah penulis pelajari.

Selesainya Tugas Akhir ini tidak lepas dari bantuan dan dukungan beberapa pihak. Sehingga pada kesempatan ini penulis mengucapkan syukur dan terima kasih kepada:

1. Allah SWT dan Nabi Muhammad SAW.
2. Papa, Mama, Dek Nisa, Dek Azra, dan Yu’sup yang telah memberikan dukungan moral dan material serta do’a yang tak terhingga untuk penulis. Serta selalu memberikan semangat dan motivasi pada penulis dalam mengerjakan Tugas Akhir ini.
3. Ibu Isye Arieshanti, S.Kom, M.Phil. selaku pembimbing I yang telah membantu, membimbing, dan memotivasi penulis dalam menyelesaikan Tugas Akhir ini dengan sabar.
4. Ibu Dr.Eng. Chastine Fatichah, S.Kom., M.Kom. selaku pembimbing II yang juga telah membantu, membimbing, dan memotivasi kepada penulis dalam mengerjakan Tugas Akhir ini.
5. Ibu Dr. Eng. Nanik Suciati, S.Kom., M.Kom. selaku Kepala Jurusan Teknik Informatika ITS, Bapak Radityo Anggoro, S.Kom.,M.Sc. selaku koordinator TA, dan segenap dosen Teknik Informatika yang telah memberikan ilmunya.
6. Ananta Pratama Chrisna Putra, S.Kom. yang telah memberikan motivasi, bantuan, doa serta hiburan kepada penulis dalam mengerjakan tugas akhir.
7. Teman-teman guys lalalayeyeye: Aida, Billa, Deasy, Nisa, dan Vivi yang telah mengisi hari-hari penulis penuh canda tawa.
8. Ustad/zah SD-SMA Al-Hikmah Surabaya yang senantiasa mengirimkan doa dan semangat kepada penulis.
9. Teman-teman Pengurus Harian HMTC 2013-2014 Kabinet Bersahabat dan teman-teman angkatan 2011 yang yang telah membantu, berbagi ilmu, menjaga kebersamaan, dan memberi motivasi kepada penulis, kakak-kakak angkatan 2009 dan 2010 serta adik-adik angkatan 2012 dan 2013 yang membuat penulis untuk selalu belajar.
10. Serta semua pihak yang yang telah turut membantu penulis dalam menyelesaikan Tugas Akhir ini.

Penulis menyadari bahwa Tugas Akhir ini masih memiliki banyak kekurangan. Sehingga dengan kerendahan hati, penulis mengharapkan kritik dan saran dari pembaca untuk perbaikan ke depannya.

Surabaya, Januari 2015

# DAFTAR ISI

[LEMBAR PENGESAHAN v](#_Toc416699956)

[*Abstrak* vii](#_Toc416699957)

[*Abstract* viii](#_Toc416699958)

[KATA PENGANTAR ix](#_Toc416699959)

[DAFTAR ISI xi](#_Toc416699960)

[DAFTAR GAMBAR xv](#_Toc416699961)

[DAFTAR TABEL xvii](#_Toc416699962)

[DAFTAR KODE SUMBER xix](#_Toc416699963)

[BAB I PENDAHULUAN 1](#_Toc416699964)

[1.1 Latar Belakang 1](#_Toc416699965)

[1.2 Rumusan Masalah 2](#_Toc416699966)

[1.3 Batasan Masalah 2](#_Toc416699967)

[1.4 Tujuan 3](#_Toc416699968)

[1.5 Manfaat 3](#_Toc416699969)

[1.6 Metodologi 3](#_Toc416699970)

[1.7 Sistematika Penulisan Laporan Tugas Akhir 5](#_Toc416699971)

[BAB II TINJAUAN PUSTAKA 7](#_Toc416699972)

[2.1 Kanker *Colon* 7](#_Toc416699974)

[2.2 Dataset Kanker *Colon* 9](#_Toc416699975)

[2.3 *Genetic Algorithm* 10](#_Toc416699976)

[2.3.1 *Encoding* 10](#_Toc416699977)

[2.3.2 Seleksi 11](#_Toc416699978)

[2.3.3 *Crossover* 11](#_Toc416699979)

[2.3.4 Mutasi 12](#_Toc416699980)

[2.3.5 Evaluasi 12](#_Toc416699981)

[2.3.6 *Stopping Criteria* 12](#_Toc416699982)

[2.4 *K-Means* 14](#_Toc416699983)

[2.5 *K-Nearest Neighbor* 15](#_Toc416699984)

[2.6 *Principal Component Analysis* 16](#_Toc416699985)

[2.7 Normalisasi 17](#_Toc416699986)

[2.7.1 Normalisasi Skala 17](#_Toc416699987)

[2.7.2 Normalisasi *Gaussian* 18](#_Toc416699988)

[BAB III DESAIN PERANGKAT LUNAK 19](#_Toc416699989)

[3.1 Desain Metode Secara Umum 19](#_Toc416699991)

[3.2 *Preprocessing* 21](#_Toc416699992)

[3.2.1 Normalisasi 21](#_Toc416699993)

[3.2.2 Reduksi Dimensi Atribut 21](#_Toc416699994)

[3.3 *Processing* 22](#_Toc416699995)

[3.3.1 Fungsi *Generate* Kromosom 22](#_Toc416699996)

[3.3.2 Fungsi Menghitung *Fitness Value* 24](#_Toc416699997)

[3.3.3 Fungsi Seleksi 26](#_Toc416699998)

[3.3.4 Fungsi *Crossover* 26](#_Toc416699999)

[3.3.5 Fungsi Mutasi 27](#_Toc416700000)

[3.3.6 Program Utama (*main*) 28](#_Toc416700001)

[BAB 4 BAB IVZZZZZ IMPLEMENTASI 29](#_Toc416700002)

[4.1 Lingkungan Implementasi 29](#_Toc416700003)

[4.2 Implementasi 29](#_Toc416700004)

[4.2.1 Parameter yang Digunakan 29](#_Toc416700005)

[4.2.2 Implementasi Fungsi readFile() 30](#_Toc416700006)

[4.2.3 Implementasi Fungsi generateKromosom() 31](#_Toc416700007)

[4.2.4 Implementasi Fungsi processCrossValidation() 32](#_Toc416700008)

[4.2.5 Implementasi Fungsi calculateFitFunc() 35](#_Toc416700009)

[4.2.6 Implementasi Fungsi doSelection() 40](#_Toc416700010)

[4.2.7 Implementasi Fungsi doCrossOver() 40](#_Toc416700011)

[4.2.8 Implementasi Fungsi doMutation() 42](#_Toc416700012)

[4.2.9 Implementasi Fungsi doEvaluation() 43](#_Toc416700013)

[4.2.10 Implementasi Program Utama 43](#_Toc416700014)

[BAB V UJI COBA DAN EVALUASI 45](#_Toc416700015)

[5.1 Lingkungan Uji Coba 45](#_Toc416700017)

[5.2 Data *Training* dan Data *Testing* 45](#_Toc416700018)

[5.3 Skenario dan Evaluasi Pengujian 47](#_Toc416700019)

[5.3.1 Skenario Uji Coba 1 47](#_Toc416700020)

[5.3.2 Skenario Uji Coba 2 49](#_Toc416700021)

[5.3.3 Skenario Uji Coba 3 51](#_Toc416700022)

[5.4 Analisis Hasil Uji Coba 52](#_Toc416700023)

[BAB 6 BAB VIZZZZZ KESIMPULAN DAN SARAN 55](#_Toc416700024)

[6.1 Kesimpulan 55](#_Toc416700025)

[6.2 Saran 56](#_Toc416700026)

[DAFTAR PUSTAKA 57](#_Toc416700027)

[LAMPIRAN 61](#_Toc416700028)

[BIODATA PENULIS 75](#_Toc416700029)

***[Halaman ini sengaja dikosongkan]***

# DAFTAR GAMBAR

[Gambar 2.1. Gambaran Stadium pada Kanker Colon [8] 8](file:///D:\KULIAH\SEMESTER%207\TA\BUKU%20TA\5111100013_Dini%20Putri%20Mandasari_Buku%20TA.docx#_Toc409550656)

[Gambar 2.2. Ilustrasi Bit-String Encoding 11](#_Toc409550657)

[Gambar 2.3. Ilustrasi Two-Point Crossover 11](file:///D:\KULIAH\SEMESTER%207\TA\BUKU%20TA\5111100013_Dini%20Putri%20Mandasari_Buku%20TA.docx#_Toc409550658)

[Gambar 2.4. Ilustrasi Bit Inversion 12](file:///D:\KULIAH\SEMESTER%207\TA\BUKU%20TA\5111100013_Dini%20Putri%20Mandasari_Buku%20TA.docx#_Toc409550659)

[Gambar 2.5. Flowchart Genetic Algorithm 13](file:///D:\KULIAH\SEMESTER%207\TA\BUKU%20TA\5111100013_Dini%20Putri%20Mandasari_Buku%20TA.docx#_Toc409550660)

[Gambar 2.6. Pseudocode Algoritma K-means 14](file:///D:\KULIAH\SEMESTER%207\TA\BUKU%20TA\5111100013_Dini%20Putri%20Mandasari_Buku%20TA.docx#_Toc409550661)

[Gambar 2.7. Ilustrasi Algoritma K-NN. (a)1-Nearest Neighbor (b)2-Nearest Neighbor (c) 3-Nearest Neighbor 15](file:///D:\KULIAH\SEMESTER%207\TA\BUKU%20TA\5111100013_Dini%20Putri%20Mandasari_Buku%20TA.docx#_Toc409550662)

[Gambar 2.8. Pseudocode Algoritma K-NN 15](file:///D:\KULIAH\SEMESTER%207\TA\BUKU%20TA\5111100013_Dini%20Putri%20Mandasari_Buku%20TA.docx#_Toc409550663)

[Gambar 2.9. Pseudocode Algoritma PCA (1) 16](file:///D:\KULIAH\SEMESTER%207\TA\BUKU%20TA\5111100013_Dini%20Putri%20Mandasari_Buku%20TA.docx#_Toc409550664)

[Gambar 2.10. Pseudocode Algoritma PCA (2) 17](file:///D:\KULIAH\SEMESTER%207\TA\BUKU%20TA\5111100013_Dini%20Putri%20Mandasari_Buku%20TA.docx#_Toc409550665)

[Gambar 3.1. Alur proses program 20](file:///D:\KULIAH\SEMESTER%207\TA\BUKU%20TA\5111100013_Dini%20Putri%20Mandasari_Buku%20TA.docx#_Toc409550666)

[Gambar 3.2. Pseudocode Principal Component Analysis 22](#_Toc409550667)

[Gambar 3.4. Pseudocode Fungsi Generate Kromosom 23](#_Toc409550668)

[Gambar 3.3. Representasi Kromosom 23](file:///D:\KULIAH\SEMESTER%207\TA\BUKU%20TA\5111100013_Dini%20Putri%20Mandasari_Buku%20TA.docx#_Toc409550669)

[Gambar 3.5. Ilustrasi Implementasi Algoritma K-means dan K-nearest Neighbor pada Fungsi Menghitung Fitness Value 24](file:///D:\KULIAH\SEMESTER%207\TA\BUKU%20TA\5111100013_Dini%20Putri%20Mandasari_Buku%20TA.docx#_Toc409550670)

[Gambar 3.6. Pseudocode Fungsi Menghitung Fitness Value 25](file:///D:\KULIAH\SEMESTER%207\TA\BUKU%20TA\5111100013_Dini%20Putri%20Mandasari_Buku%20TA.docx#_Toc409550671)

[Gambar 3.7. Pseudocode Fungsi Seleksi 26](file:///D:\KULIAH\SEMESTER%207\TA\BUKU%20TA\5111100013_Dini%20Putri%20Mandasari_Buku%20TA.docx#_Toc409550672)

[Gambar 3.8. Pseudocode Fungsi Crossover 27](#_Toc409550673)

[Gambar 3.9. Pseudocode Fungsi Mutasi (1) 27](#_Toc409550674)

[Gambar 3.10. Pseudocode Fungsi Mutasi (2) 28](#_Toc409550675)

[Gambar 3.11 Pseudocode Fungsi Utama 28](#_Toc409550676)

[Gambar 5.1. Visualisasi 5-Crossvalidation 46](file:///D:\KULIAH\SEMESTER%207\TA\BUKU%20TA\5111100013_Dini%20Putri%20Mandasari_Buku%20TA.docx#_Toc409550677)

[Gambar 5.2. Grafik Hasil Akurasi Skenario Uji Coba 1 48](file:///D:\KULIAH\SEMESTER%207\TA\BUKU%20TA\5111100013_Dini%20Putri%20Mandasari_Buku%20TA.docx#_Toc409550678)

[Gambar 5.3. Grafik Hasil Running Time Skenario Uji Coba 1 48](file:///D:\KULIAH\SEMESTER%207\TA\BUKU%20TA\5111100013_Dini%20Putri%20Mandasari_Buku%20TA.docx#_Toc409550679)

[Gambar 5.4. Grafik Hasil Akurasi Skenario Uji Coba 2 50](file:///D:\KULIAH\SEMESTER%207\TA\BUKU%20TA\5111100013_Dini%20Putri%20Mandasari_Buku%20TA.docx#_Toc409550680)

[Gambar 5.5. Grafik Hasil Running Time Skenario Uji Coba 2 50](file:///D:\KULIAH\SEMESTER%207\TA\BUKU%20TA\5111100013_Dini%20Putri%20Mandasari_Buku%20TA.docx#_Toc409550681)

[Gambar 5.6. Grafik Hasil Akurasi Skenario Uji Coba 3 51](file:///D:\KULIAH\SEMESTER%207\TA\BUKU%20TA\5111100013_Dini%20Putri%20Mandasari_Buku%20TA.docx#_Toc409550682)

[Gambar 5.7. Grafik Hasil Running Time Skenario Uji Coba 3 52](file:///D:\KULIAH\SEMESTER%207\TA\BUKU%20TA\5111100013_Dini%20Putri%20Mandasari_Buku%20TA.docx#_Toc409550683)

***[Halaman ini sengaja dikosongkan]***

# DAFTAR TABEL

[Tabel 4.1. Tabel Parameter yang Digunakan 30](#_Toc409550684)

[Tabel 5.1. Tabel Perbandingan Hasil Akurasi dan Running Time 53](#_Toc409550685)

[Tabel A.1. Rekap Hasil Keluaran Program Skenario Uji Coba 1 pada Percobaan 1 61](#_Toc409550686)

[Tabel A.2. Rekap Hasil Keluaran Program Skenario Uji Coba 1 pada Percobaan 2 62](#_Toc409550687)

[Tabel A.3. Rekap Hasil Keluaran Program Skenario Uji Coba 1 pada Percobaan 3 63](#_Toc409550688)

[Tabel A.4. Rekap Hasil Keluaran Program Skenario Uji Coba 1 pada Percobaan 4 64](#_Toc409550689)

[Tabel A.5. Rekap Hasil Keluaran Program Skenario Uji Coba 1 pada Percobaan 5 65](#_Toc409550690)

[Tabel A.6. Rekap Hasil Keluaran Program Skenario Uji Coba 1 pada Percobaan 6 66](#_Toc409550691)

[Tabel A.7. Rekap Hasil Keluaran Program Skenario Uji Coba 1 pada Percobaan 7 67](#_Toc409550692)

[Tabel A.8. Rekap Hasil Keluaran Program Skenario Uji Coba 1 pada Percobaan 8 68](#_Toc409550693)

[Tabel A.9. Rekap Hasil Keluaran Program Skenario Uji Coba 1 pada Percobaan 9 69](#_Toc409550694)

[Tabel A.10. Rekap Hasil Keluaran Program Skenario Uji Coba 1 pada Percobaan 10 70](#_Toc409550695)

[Tabel A.11. Rekap Hasil Keluaran Program 10 Kali Percobaan pada Skenario Uji Coba 1 71](#_Toc409550696)

[Tabel A.12. Rekap Hasil Keluaran Program 10 Kali Percobaan pada Skenario Uji Coba 2 72](#_Toc409550697)

[Tabel A.13. Rekap Hasil Keluaran Program 10 Kali Percobaan pada Skenario Uji Coba 3 73](#_Toc409550698)

[Tabel A.14. Model Terbaik yang Dihasilkan 74](#_Toc409550699)

***[Halaman ini sengaja dikosongkan]***

# DAFTAR KODE SUMBER

[Kode Sumber 4.1. Kode Sumber Fungsi readFile (1) 30](#_Toc409550700)

[Kode Sumber 4.2. Kode Sumber Fungsi readFile (2) 31](#_Toc409550701)

[Kode Sumber 4.3. Kode Sumber Fungsi generateKromosom() 32](#_Toc409550702)

[Kode Sumber 4.4. Kode Sumber Fungsi processCrossValidation (1) 32](#_Toc409550703)

[Kode Sumber 4.5. Kode Sumber Fungsi processCrossValidation (2) 33](#_Toc409550704)

[Kode Sumber 4.6. Kode Sumber Fungsi processCrossValidation (3) 34](#_Toc409550705)

[Kode Sumber 4.7. Kode Sumber Fungsi processCrossValidation (4) 35](#_Toc409550706)

[Kode Sumber 4.8. Kode Sumber Pemilihan Data sebagai Centroid 35](#_Toc409550707)

[Kode Sumber 4.9. Kode Sumber Perhitungan Nilai Disimilarity 36](#_Toc409550708)

[Kode Sumber 4.10. Kode Sumber Pengecekkan Anggota Cluster 36](#_Toc409550709)

[Kode Sumber 4.11. Kode Sumber Perbarui Nilai Centroid 37](#_Toc409550710)

[Kode Sumber 4.12. Kode Sumber Perhitungan Akurasi 38](#_Toc409550711)

[Kode Sumber 4.13. Kode Sumber Fungsi isCompleted() dan isDuplicated(2) 39](#_Toc409550712)

[Kode Sumber 4.14. Kode Sumber Fungsi Check() 39](#_Toc409550713)

[Kode Sumber 4.15. Kode Sumber Fungsi doSelection() 40](#_Toc409550714)

[Kode Sumber 4.16. Kode Sumber Individu yang Tidak Dikenai Crossover 40](#_Toc409550715)

[Kode Sumber 4.17. Kode Sumber Individu yang Dikenai Crossover 41](#_Toc409550716)

[Kode Sumber 4.18. Kode Sumber Fungsi doMutation() 42](#_Toc409550717)

[Kode Sumber 4.19. Kode Sumber Fungsi doEvaluation() 43](#_Toc409550718)

[Kode Sumber 4.20. Kode Sumber Program Utama 44](#_Toc409550719)

***[Halaman ini sengaja dikosongkan]***

# BAB I PENDAHULUAN

## Latar Belakang

Dewasa ini, perkembangan penyakit semakin memprihatinkan dikarenakan gaya hidup manusia yang tidak sehat. Banyak sekali penyakit-penyakit ganas yang bermunculan dan salah satunya adalah kanker *colon*. Kanker *colon* menempati peringkat kedua penyebab kematian di dunia sehingga kita tidak bisa memandang remeh penyakit ini. Melihat tingkat keganasan penyakit kanker *colon*, dibutuhkan metode yang handal untuk mendiagnosisnya.

Banyak metode yang sudah dilakukan untuk melakukan klasifikasi/pengelompokkan data *microarray* DNA. Data *microarray* DNA memiliki dimensi yang besar sebagai representasi detail dari data tersebut. Sayangnya, dimensi yang banyak itu tidak menjamin bahwa semuanya mempunyai relevansi yang tinggi untuk suatu data, sehingga akan berpengaruh buruk terhadap komputasi data tersebut, begitu pula dengan dataset kanker *colon. K-Means* adalah salah satu metode yang sering digunakan untuk pengelompokkan data medis. Pengimplementasian yang sederhana dan ketepatan prediksi yang cukup tinggi menjadi alasannya. Tetapi, *K-Means* mempunyai kelemahan, yaitu penentuan *initial seeds* yang random bisa saja tidak mewakili grup yang ada dalam populasi. Disamping permasalahan penentuan *initial seeds,* permasalahan lain yang sering dihadapi pada metode *K-Means* adalah penentuan jumlah klaster atau nilai *k*. Perlu adanya optimasi pemilihan model terbaikuntuk hasil klastering yang lebih baik.

Penentuan klasifikasi penyakit kanker *colon* akan dilakukan menggunakan metode *Principal Component Analysis* dan *Genetic Algorithm* untuk pemilihan atribut (mereduksi dimensi) dan optimasi nilai *k* untuk *K-Means*. Setelah mendapatkan model terbaik, dilakukan klasifikasi data menggunakan metode *K-Nearest Neighbor.*  Adapun hasil yang diharapkan adalah tingginya tingkat akurasi pengklasifikasian menggunakan metode gabungan *Genetic Algorithm, K-Means,* dan *K-Nearest Neighbor* dibandingkan dengan algoritma *K-Nearest Neighbor* saja, sehingga metode gabungan ini bisa dijadikan metode alternatif untuk kasus pengklasifikasian data.

## Rumusan Masalah

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

* + - 1. Bagaimana mengimplementasikan metode *Genetic Algorithm, K-Means,* dan *K-Nearest Neighbor,* serta *Principal Component Analysis* untuk mereduksi dimensi fitur?
      2. Bagaimana menerapkan metode *Genetic Algorithm, K-Means,* dan *K-Nearest Neighbor,* serta *Principal Component Analysis* untuk mendapatkan model terbaik untuk permasalahan klasifikasi data kanker *colon*?
      3. Bagaimana tingkat akurasi yang didapatkan?

## Batasan Masalah

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

1. Dataset yang digunakan adalah data kanker *colon* dengan 2000 atribut dan 62 data (40 data positif kanker dan 22 data negatif kanker).
2. Prosentase pembagian data *training* dan data *testing* adalah *5-crossvalidation.*
3. Pengklasifikasian data berdasarkan positif kanker atau negatif.

## Tujuan

Tugas Akhir ini mempunyai beberapa tujuan, yaitu sebagai berikut:

1. Mengembangkan sistem aplikasi pengklasifikasian penyakit kanker *colon.*
2. Mengimplementasikan metode *Principal Component Analysis* untuk mereduksi dimensi fitur, *Genetic Algorithm* untuk optimasi nilai *k* untuk *K-Means,* serta *K-Nearest Neighbor* untuk menyelesaikan permasalahan klasifikasi penyakit kanker *colon*.

## Manfaat

Dengan dibuatnya Tugas Akhir ini maka klasifikasi data kanker *colon* akan lebih akurat dan efisien. Selain itu, dapat memberikan alternatif klasifikasi menggunakan metode klastering.

## Metodologi

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

1. Penyusunan proposal Tugas Akhir.

Tahap awal untuk memulai pengerjaan Tugas Akhir adalah penyusunan proposal Tugas Akhir. Proposal Tugas Akhir yang diajukan memiliki gagasan yang sama dengan Tugas Akhir ini, yaitu implementasi metode *Genetic Algorithm, K-Means,* dan *K-Nearest Neighbor* untuk melakukan klasifikasi data.

1. Studi literatur

Pada tahap ini dilakukan pencarian, pengumpulan, pembelajaran dan pemahaman informasi dan literatur yang diperlukan untuk pembuatan implementasi metode *Genetic Algorithm* sebagai metode optimasi *K-Means* dan *K-Nearest Neighbor*. Informasi dan literatur didapatkan dari literatur buku dan sumber-sumber informasi lain yang berhubungan.

1. Analisis dan desain 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. Serta dilakukan desain suatu sistem dan desain proses-proses yang ada.

1. Implementasi perangkat lunak

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

1. Pengujian dan evaluasi

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

1. Penyusunan buku Tugas Akhir.

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

## 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:

**Bab I Pendahuluan**

Bab yang berisi mengenai latar belakang, tujuan, dan manfaat dari pembuatan Tugas Akhir. Selain itu permasalahan, batasan masalah, metodologi yang digunakan, 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 tentang desain sistem yang disajikan dalam bentuk *pseudocode*.

**Bab IV Implementasi**

Bab ini membahas implementasi dari desain yang telah dibuat pada bab sebelumnya. Penjelasan berupa *code* yang digunakan untuk proses implementasi.

**Bab V Uji Coba Dan Evaluasi**

Bab ini menjelaskan kemampuan perangkat lunak dengan melakukan pengujian kebenaran dan pengujian kinerja dari sistem yang telah dibuat.

**Bab VI Kesimpulan Dan Saran**

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

# BAB II TINJAUAN PUSTAKA

Bab ini berisi penjelasan teori-teori yang berkaitan dengan algoritma yang diajukan pada pengimplementasian program. Penjelasan ini bertujuan untuk memberikan gambaran secara umum terhadap program yang dibuat dan berguna sebagai penunjang dalam pengembangan perangkat lunak.



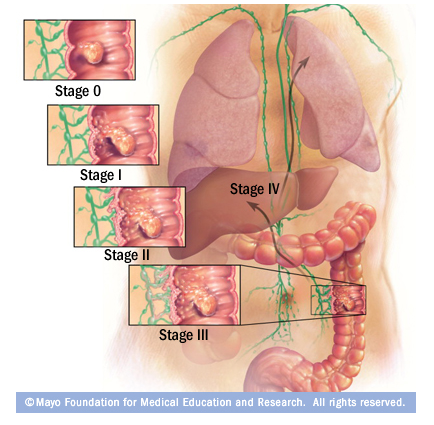
## Kanker *Colon*

Kanker *colon* adalah penyakit yang terjadi dikarena kemunculan sel yang tidak normal di dalam usus besar. Kanker *colon* merupakan jenis kanker yang sering terjadi dan menempati peringkat kedua penyebab kematian di dunia. Kanker *colon* menyebabkan 655.000 kematian di seluruh dunia setiap tahunnya [2]. Beberapa penyebab terjadinya kanker *colon*:

1. Pola makan yang salah dengan terlalu banyak mengonsumsi lemak dan protein, tetapi rendah serat.
2. Obesitas (kegemukan).
3. Sering mengonsumsi makanan yang mengandung bahan pengawet maupun bahan pewarna sintetis.
4. Merokok.
5. Jarang melakukan kegiatan fisik.
6. Pernah terkena kanker *colon*/polip di usus.

Gambar 2.1 menunjukkan gambaran stadium pada kanker *colon*. Mencegah lebih baik daripada mengobati. Melihat dari penyebab terjadinya kanker *colon* di atas, pencegahan dapat dilakukan dengan cara [3]:

1. Hindari makanan yang mengandung tinggi lemak, protein, kalori, serta daging merah. Disarankan pula mengonsumsi makanan yang mengandung kalsium dan asam folat.
2. Banyak mengonsumsi vitamin E dan vitamin D yang dapat membantu memperkuat kerja sistem imun.
3. Makan buah dan sayuran setiap hari.
4. Pertahankan IMT (Indeks Massa Tubuh).
5. Lakukan aktivitas fisik.
6. Hindari kebiasaan merokok.



*Gambar 2.1. Gambaran Stadium pada Kanker Colon [8]*

Pengobatan yang dapat dilakukan apabila terdiagnosis terkena kanker *colon* adalah [3]:

1. Pembedahan /operasi
2. Kemoterapi
3. Radiasi /radioterapi

Selain dari faktor eksternal, kanker *colon* juga dapat terjadi dari faktor internal atau turunan [8]. Mutasi gen bawaan yang meningkatkan resiko kanker usus dapat diturunkan dalam silsilah keluarga, tetapi ini hanya merupakan sebagian kecil penyebab kanker *colon*. Beberapa sindrom kanker usus antara lain:

1. *Familial adenomatous polyposis (FAP)*

FAP adalah penyakit langka penyebab ribuan polip di saluran usus besar dan *rectum*. Penanganan yang telat dan perawatan yang tidak baik pada orang yang memiliki FAP mampu memperbesar resiko kanker *colon* sebelum usia 40 tahun.

1. *Hereditary nonpolyposis colorectal cancer (HNPCC)*

HNPCC yang juga disebut *Lynch syndrome* meningkatkan resiko kanker *colon* dan kanker lain. Orang dengan HNPCC cenderung terkena kanker *colon* sebelum usia 50 tahun.

## Dataset Kanker *Colon*

Dataset kanker *colon,* diambil dari situs *BioInformatics Research Group*[13],terdiri atas 2000 atribut dengan keseluruhan atribut bertipe data numerik yang merepresentasikan ekspresi gen dari jaringan *tissue* pasien. Dataset ini mempunyai 62 *record* dengan 40 pasien positif terkena kanker *colon* dan 22 pasien negatif terkena kanker *colon.* Adapun *summary statistics* dari dataset tersebut dapat dilihat pada Tabel 2.1.

*Tabel 2.1. Summary Statistics*

|  |  |
| --- | --- |
| Jumlah Kolom | 2000 |
| Jumlah Baris | 62 |
| Mean | 403.79 |
| Varians | 495480.39 |
| Standar Deviasi | 703.90 |
| Minimal | 5.8163 |
| Maksimal | 20903.178 |

## *Genetic Algorithm*

*Genetic Algorithm (GA)* terinspirasi dari teori Darwin dan ditemukan pertama kali oleh John Holland [1]. GA adalah salah satu jenis metode pencarian stokastik global dan metode optimasi yang mengadaptasi proses natural evolusi biologi. Berikut istilah-istilah dan operasi-operasi yang digunakan pada GA:

1. Gen

Gen adalah bagian dari tiap individu/kromosom yang berisi parameter sesuai permasalahan yang akan dicari.

1. Individu/kromosom

Kromosom adalah serangkaian gen yang merepresentasikan solusi sesuai dengan permasalahan yang akan dicari.

1. Populasi

Populasi adalah kumpulan dari beberapa kromosom yang akan diproses pada setiap generasi.

1. Generasi

Generasi adalah kumpulan populasi yang diperoleh dari operasi GA.

GA beroperasi dimulai dari pemilihan populasi solusi yang potensial. Perhitungan nilai *fitness* adalah tahapan yang harus dilakukan pada metode ini. GA mempunyai beberapa operasi penting, yaitu *encoding,* seleksi, *crossover*, mutasi, dan evaluasi. Selain itu, GA juga mempunyai kriteria untuk menghentikan jalannya proses GA ini (*stopping criteria*). Masing-masing operasi dan *stopping criteria* tersebut akan dijelaskan sebagai berikut.

### *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 0/1 [14]. Ilustrasi *bit-string encoding* ditunjukkan pada Gambar 2.2.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | .. | .. | .. | 0 |

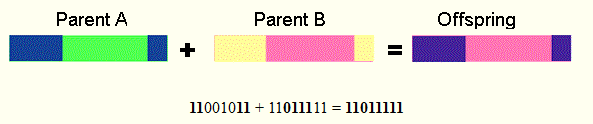
*Gambar 2.2. Ilustrasi Bit-String Encoding*

### Seleksi

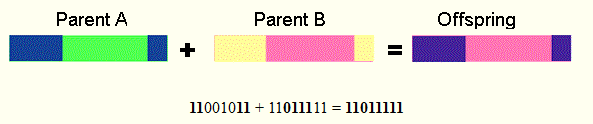
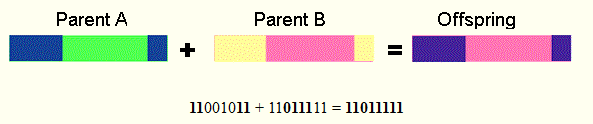
Seleksiadalah operasi memilih invidu baru sebagai *parent*. Pemilihan yang dilakukan berdasarkan hasil *fitness value* yang dimiliki setiap individu. Ada berbagai jenis metode seleksi pada GA, salah satunya adalah *elitism. Elitism* adalah proses menyeleksi individu dengan mengambil beberapa individu terbaik. *Elitism* mampu dengan cepat meningkatkan performa GA karena teknik seleksi ini mampu mencegah hilangnya individu terbaik.

### *Crossover*

*Crossover* adalah kawin saling, yaitu saling bertukarnya beberapa bagian pada dua individu *parent*, berdasarkan dari titik tertentu pada sebuah individu. Operasi ini dilakukan untuk mengaktifkan proses evolusi menuju ruang hasil yang lebih baik. Hal ini dikarenakan keturunan-keturunan yang dihasilkan mewarisi gen-gen yang baik dari kedua *parent* yang merupakan individu terbaik pula. Terdapat banyak sekali teknik *crossover,* salah satunya *two-point crossover,* dimana akan dipilih secara random titik awal dan titik akhir bagian dari individu yang dikenakan kawin silang [14]. Ilustrasi teknik *two-point crossover* ditunjukkan pada Gambar 2.3.



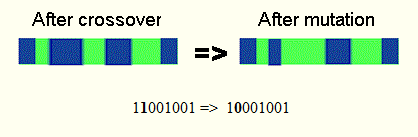
*Gambar 2.3. Ilustrasi Two-Point Crossover*



**11001011 + 11011111 = 11011100 dan 11001011**

### Mutasi

Mutasi adalah proses mengganti beberapa nilai gen pada kromosom untuk menghidari *local convergence* pada GA. Mutasi dilakukan bertujuan untuk menjaga variasi kromosom dari satu generasi ke generasi selanjutnya. Banyak sekali teknik mutasi, salah satunya adalah *bit inversion,* dimana gen yang dikenakan mutasi apabila bernilai 0 akan diubah menjadi 1, sedangkan yang bernilai 1 akan diubah menjadi 0 [14]. Ilustrasi teknik *bit inversion* ditunjukkan pada Gambar 2.4.



**11001001 => 10001001**

**Setelah Mutasi**

**Setelah *Crossover***

*Gambar 2.4. Ilustrasi Bit Inversion*

### Evaluasi

Evaluasi pada GA adalah menghitung *fitness value* dari suatu individu/kromosom. *Fitness value* pada GA bermacam-macam, tergantung dengan permasalahan yang akan diselesaikan[14]. Pada tugas akhir kali ini, perhitungan *fitness value* menggunakan nilai akurasi metode *K-Nearest Neighbor* dari individu yang terbentuk.

### *Stopping Criteria*

*Stopping Criteria* adalah kriteria untuk menghentikan proses jalannya GA. Beberapa kriteria yang sering digunakan antara lain [21]:

1. Telah tercapainya generasi yang diinginkan.
2. Dalam beberapa generasi berturut-turut didapatkan nilai *fitness* tertinggi yang tidak berubah-ubah.
3. Dalam *n* generasi berikutnya tidak didapatkan nilai *fitness* yang lebih tinggi dari sebelumnya.

Pada pengerjaan tugas akhir kali ini, kriteria yang digunakan adalah telah tercapainya jumlah generasi yang diinginkan.

Secara keseluruhan, tahapan GA dirangkum dalam sebuah f*lowchart* yang ditunjukkan pada Gambar 2.5.

*Generate* populasi awal

Evaluasi nilai *fitness* untuk masing-masing populasi

Selesai

Tidak

Ya

Mutasi

Jumlah maksimal generasi tercapai

Populasi baru terbentuk dan evaluasi nilai *fitness*

*Crossover*

Seleksi

*Gambar 2.5. Flowchart Genetic Algorithm*

## *K-Means*

*K-Means* adalah salah satu algoritma klastering (pengelompokkan) yang paling populer. Algoritma ini mengelompokkan objek ke beberapa klaster, dimana jumlah klaster telah ditentukan sebelumnya. *K-Means* dimulai dengan pemilihan *centroid* (titik tengah) secara random, satu untuk masing-masing klaster. Setelah itu, masing-masing data yang ingin dikelompokkan dihitung jaraknya (*disimilarity*) dengan masing-masing *centroid.* Perhitungan *disimilarity* dilakukan menggunakan *Euclidian Distance* denganrumus [3]:

|  |  |
| --- | --- |
|  | (2.1) |

Setelah dilakukan perhitungan *disimilarity*, data tersebut akan dikelompokkan ke dalam klaster yang mempunya nilai jarakpaling kecil. Iterasi terus dilakukan sampai *stopping criteria* terpenuhi. Dalam hal ini, *stopping criteria* yang digunakan adalah data dalam sebuah klaster tidak berubah-ubah lagi. *Pseudocode* algoritma *K-Means* ditunjukkan pada Gambar 2.6.

**Masukan** : K : Jumlah klaster

D : Dataset yang terdiri dari n objek

**Keluaran** : Beberapa klaster yang masing-masing berisi

beberapa data

**Metode** :

* 1. Pilih secara acak k objek sebagai *centroid.*
  2. *do*
  3. Masukkan masing-masing objek ke klaster yang mempunyai nilai disimilaritas paling rendah (jarak paling kecil) berdasarkan rata-rata klaster.
  4. Perbarui terus nilai rata-rata klaster
  5. *until* data dalam klaster tidak berubah

*Gambar 2.6. Pseudocode Algoritma K-means*

## *K-Nearest Neighbor*

*K-Nearest Neighbour* (K-NN) adalah algoritma klasifikasi sederhana yang menyimpan semua data (kasus) dan mengklasifikasikan data (kasus) baru berdasarkan nilai similaritas. Algoritma K-NN sudah digunakan sejak 1970 di berbagai kasus klasifikasi. K-NN digunakan ketika seluruh atribut bernilai kontinu [5]. Dimulai dengan penentuan jumlah tetangga atau nilai *k* sebagai inisialisasi. Selanjutnya dihitung jarak (*disimilarity*) antara data *testing* dengan data *training* sesuai dengan persamaan 2.1.

**x**

**x**

**x**

(a) (b) (c)

*Gambar 2.7. Ilustrasi Algoritma K-NN. (a)1-Nearest Neighbor (b)2-Nearest Neighbor (c) 3-Nearest Neighbor*

*Pseudocode* algoritma K-NN dapat dilihat pada Gambar 2.8.

**Step 1)** Masukkan nilai *k*

**Step 2)** Cari *k* data terdekat dengan data yang

belum diketahui kelasnya menggunakan

persamaan 2.1.

*Gambar 2.8. Pseudocode Algoritma K-NN*

## *Principal Component Analysis*

Metode *Principal Component Analysis* (PCA) dibuat pertama kali oleh para ahli statistik dan ditemukan oleh Karl Pearson pada tahun l90l yang memakainya pada bidang biologi. Perkembangannya baru mulai pesat pada akhir tahun l930 dan awal 1940. Setelah itu, perkembangan PCA agak berkurang hingga ditemukannya komputer. Dunia komputer mampu mengaplikasikan teknik ini pada masalah-masalah yang masuk akal [20].

PCA adalah teknik yang digunakan untuk menyederhanakan suatu data. Data akan ditransformasi secara linier ke dalam sistem koordinat yang baru. Pereduksian dimensi oleh PCA mampu mengurangi besar dimensi tanpa mengurangi karakteristik data tersebut secara signifikan. Metode ini mengubah sebagian besar variabel asli yang saling berkorelasi menjadi satu himpunan variabel baru yang lebih kecil dan tidak berkorelasi lagi [12].

Diberikan dataset matriks *X* berukuran (*n*x*D*) yang terdiri dari *n* observasi dengan *D* dimensi. Tahapan algoritma PCA dapat dilihat pada Gambar 2.9 dan Gambar 2.10.

*Gambar 2.9. Pseudocode Algoritma PCA (1)*

1. Hitung matriks varians-kovarian. Diagonal mengandung nilai varians dan sisi lainnya mengandung nilai kovarians.

|  |  |
| --- | --- |
|  | (2.2) |
|  | (2.3) |

Dimana dan adalah data dan dan adalah rata – rata dari masing – masing atribut data.

1. Hitung nilai eigen *λ* dan vektor eigen ***v***

|  |  |
| --- | --- |
|  | (2.4) |

Dimana Ʃ adalah kovarians matriks, λ adalah eigen value dan *v* adalah eigen vektor.

*Gambar 2.10. Pseudocode Algoritma PCA (2)*

1. Urutkan nilai eigen secara *descending* untuk menentukan pengambilan jumlah principal komponen yang dibutuhkan. Berapa persen varians yang terjelaskan merepresentasikan berapa persen informasi yang diambil dan terdapat kesepakatan menurut skala kaiser mayer olkin sebagai berikut :

*Tabel 2.2. Skala Kaiser Mayer Olkin (KMO)*

|  |  |
| --- | --- |
| **SKALA KAISER MAYER OLKIN** | |
| 0,9 | Sangat Baik |
| 0,8 | Baik |
| 0,7 | Cukup |
| 0,6 | Kurang |
| 0,5 | Jelek |
| <0,5 | Tidak dapat diterima |

1. Selanjutnya dicari nilai loading *l* untuk masing – masing variabel.

|  |  |
| --- | --- |
|  | (2.5) |

1. Pada persamaan diatas skor komponen adalah *Y*, dimana merupakan perkalian antara vektor eigen dengan matriks *X* yang telah dinormalisasi menggunakan persamaan 2.7.

## Normalisasi

Normalisasi adalah teknik *preprocessing* untuk melakukan penskalaan dan penstandaran data. Dua contoh normalisasi yang dikenal adalah normalisasi skala dan normalisasi *Gaussian* (standardisasi).

### Normalisasi Skala

Normalisasi skala yaitu melakukan penskalaan data pada rentang tertentu, dimana rentang yang umum digunakan adalah rentang 0-1. Normalisasi ini bertujuan untuk menghindari rentang data yang terlalu jauh. Rumus untuk menghitung normalisasi skala adalah [22]:

|  |  |
| --- | --- |
|  | (2.6) |

### Normalisasi *Gaussian*

Standardisasi adalah proses pembuatan data berdistribusi normal (0,1). Proses ini sangat berguna untuk mengecilkan varians secara total, sehingga tidak ada dominasi varians pada beberapa data/atribut. Proses ini sangat penting untuk dilakukan, khususnya pada tahap PCA. Rumus untuk menghitung standardisasi adalah sebagai berikut [22]:

|  |  |
| --- | --- |
|  | (2.7) |

# BAB III DESAIN PERANGKAT LUNAK

Pada bab ini akan dijelaskan perancangan program yang dibuat. Perancangan akan dibagi menjadi dua proses utama, yaitu:

1. *Preprocessing* untuk menormalisasi dataset serta mereduksi dimensi menggunakan PCA.
2. Pengimplementasian *Genetic Algorithm* dan *K-Means* untuk membangun model terbaik yang diikuti dengan perhitungan keakuratan model menggunakan algoritma *K-Nearest Neighbor.*

Pada bab ini akan dijelaskan gambaran umum setiap program utama dalam *flowchart* selanjutnya untuk penjelasan lebih detail akan disajikan dalam *pseudocode*.



## Desain Metode Secara Umum

Metode ini hanya memiliki satu tahap inti, yaitu tahap membangun model terbaik menggunakan *Genetic Algorithm* dan *K-Means.* Dimulai dari inputan berupa dataset hasil tahap *preprocessing.* Kemudian selanjutnya adalah tahap inti yaitu *Genetic Algorithm* dan *K-Means* untuk membangun model terbaik. Terakhir adalah hasil klasifikasi dan perhitungan akurasi dari data *testing* terhadap model terbaik yang telah terbentuk pada tahap sebelumnya menggunakan *K-Nearest Neighbor*. Diagram alur dari proses ditunjukkan oleh Gambar 3.1.

K-NN *Classification*

K-NN *Classification*

Normalisasi

Reduksi dimensi fitur menggunakan PCA

*Preprocessing*

*Processing*

Data *Training*

*Genetic Algorithm*

Data Training 1 + parameter *K-Means*

Data Training 2 + parameter *K-Means*

Data Training n + parameter *K-Means*

*K-Means clustering*

*K-Means clustering*

*K-Means clustering*

K-NN *Classification*

Perhitungan Performa

Data *Testing*

Model Terbaik

*Gambar 3.1. Alur proses program*

## *Preprocessing*

Dalam mengawali proses pengerjaan program, maka harus disiapkan terlebih dahulu data yang akan digunakan dalam proses pengerjaan. *Preprocessing* adalah tahapan yang penting untuk dilakukan. Pada tugas akhir kali ini, tahap *preprocessing* yang dilakukan adalah normalisasi dan reduksi dimensi fitur.

### Normalisasi

Normalisasi yang dilakukan adalah normalisasi *gaussian* atau standardisasi. Hal ini membuat data berdistribusi normal (0,1). Perhitungan nilai normalisasi menggunakan persamaan 2.7.

### Reduksi Dimensi Atribut

Dataset kanker *colon* mempunyai dimensi fitur yang besar, yaitu 2000 atribut. Besarnya dimensi fitur tentu akan sangat mempengaruhi kecepatan (*running time*) dalam proses pengolahan data. Oleh karena itu, perlu dilakukan reduksi dimensi fitur.

*Principal Component Analysis* (PCA) yaitu metode yang dapat mentransformasikan fitur ke dimensi lain dengan menghitung tingkat kepentingan informasi yang akan diambil. Sebelum melakukan PCA, data harus distandardisasi terlebih dahulu karena akan mempengaruhi nilai eigen yang dihasilkan. Jumlah nilai eigen yang terbentuk dari proses PCA harus sama dengan jumlah atribut. Standardisasi dilakukan untuk menghilangkan efek dominasi varians suatu variabel terhadap varaibel lain.

Pengerjaan normalisasi dan reduksi dimensi atribut menggunakan *library* matlab, dimana masukan dalam proses ini adalah matriks data kanker *colon. Pseudocode* PCA dapat dilihat pada Gambar 3.2.

|  |  |
| --- | --- |
| Masukan | Matriks data kanker *colon* |
| Keluaran | Data yang telah distandardisasi, nilai loading, nilai eigen, skor komponen, varians terjelaskan |
| * 1. data = data kanker *colon*   2. dataNormal = standardisasi(data)   3. [COEFF,SCORE,LATENT] = princomp(dataNormal)   4. varTerjelaskan = cumsum(LATENT) / sum(LATENT) | |

*Gambar 3.2. Pseudocode Principal Component Analysis*

COEFF menunjukkan nilai loading, SCORE menunjukkan nilai skor komponen, dan LATENT menunjukkan nilai eigen yang dihasilkan.

Dari nilai eigen yang dihasilkan, dapat dihitung nilai varians yang terjelaskan dari sejumlah fitur baru yang diambil tersebut. Pada tugas akhir kali ini, diambil fitur sejumlah total varians terjelaskan >= 99.98% yang menurut skala Kaiser-mayor olkin data tersebut berkriteria sangat baik. Oleh karena itu, atribut yang awalnya berjumlah 2000 artribut disusutkan menjadi 61 atribut. Hanya 61 atribut ini sajalah yang akan terus digunakan untuk tahapan selanjutnya.

## *Processing*

Pada bagian ini dijelaskan *pseudocode* dari metode *Genetic Algorithm* (GA). Metode GA mempunyai 6 fungsi utama, yaitu fungsi*generate* kromosom, fungsi menghitung *fitness value,* fungsi seleksi, fungsi *crossover,* dan fungsi mutasi. Penerapan algoritma GA ini menggunakan bahasa pemrogramanJava.

### Fungsi *Generate* Kromosom

Fungsi *generate* kromosom adalah fungsi pertama dalam algoritma genetik. Kromosom direpresentasikan dalam bilangan biner seukuran banyaknya fitur, yaitu 61 atribut. Atribut yang bernilai 1 akan digunakan untuk perhitungan selanjutnya, sedangkan atribut yang bernilai 0 tidak akan digunakan.

Selain itu, kromosom juga direpresentasikan dengan sebuah nilai sebagai parameter k pada *K-Means. Generate* bit dan nilai kromosom menggunakan fungsi yang sudah disediakan Java, yaitu Math.Random()*.* Representasi kromosom yang digunakan pada tugas akhir kali ini ditunjukkan pada Gambar 3.3.

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | .. | .. | 1 | 3 |

Bilangan biner berukuran banyaknya atribut

Parameter nilai *k*

*Gambar 3.3. Representasi Kromosom*

Input dari fungsi ini merupakan informasi banyaknya atribut hasil reduksi dimensi fitur oleh PCA, batas minimal dan maksimal parameter nilaiK untuk *K-Means*. Untuk inisialisasi awal, akan di-*generate* sebanyak 100 individu. *Pseudocode* fungsi *generate* kromosom dapat dilihat pada Gambar 3.4.

|  |  |
| --- | --- |
| Masukan | Informasi banyaknya atribut, batas minimal, dan maksimal parameter nilai *k* |
| Keluaran | Representasi kromosom sesuai dengan *Gambar 3.3* |
| * 1. individu = new byte[dimensi fitur]   2. for(int i=0; i<individu.length; i++)   3. individu[i] = *generate* bit   4. end for   5. int k = batas maksimal parameter nilai k   6. nilaiK = *generate* k | |

*Gambar 3.4. Pseudocode Fungsi Generate Kromosom*

### Fungsi Menghitung *Fitness Value*

Fungsi menghitung *fitness value* adalah fungsi untuk menghitung nilai *fitness* sebuah individu. Di dalam fungsi inilah letak implementasi algoritma *K-Means* dan *K-Nearest Neighbor.* Nilai *fitness* yang digunakan adalah nilai akurasi dari algoritma *K-Nearest Neighbor.* Data *training* akan dipisah antara label positif dan negatif kanker, yang kemudian masing-masing kelompok akan dikenakan algoritma *K-Means.* Data akan dikelompokkan sebanyak nilaiK kelompok. Pemisahan ini bertujuan untuk pemberian label data *testing* untuk evaluasi keakuratan prediksi kelas nantinya. Setelah terbentuk kelompok-kelompok, data *testing* akan dihitung jarak (atau nilai *disimilarity*) terhadap kelompok data *training*. Data *testing* dengan nilai jarak paling kecil atau kemiripan paling besar akan dilabeli dengan label kelompok tersebut. Jarak data *testing* dengan data *training* dihitung menggunakan *Euclidian Distance* seperti pada persamaan 2.1.

**Data yang belum diketahui labelnya**

**Dihitung kemiripannya dengan antar klaster**

**Klaster**

**1**

**Klaster**

**2**

**Klaster**

**3**

**Klaster**

**n**

**Kumpulan data *training* berlabel negatif**

**Klaster**

**1**

**Klaster**

**2**

**Klaster**

**3**

**Klaster**

**n**

**Kumpulan data *training* berlabel positif**

*Gambar 3.5. Ilustrasi Implementasi Algoritma K-means dan K-nearest Neighbor pada Fungsi Menghitung Fitness Value*

**Data yang sudah diketahui labelnya**

Gambar 3.5. adalah ilustrasi implementasi algoritma *K-Means* dan *K-Nearest Neighbor* pada fungsi menghitung *fitness value*. Data yang telah dilabeli akan dihitung keakuratan prediksi kelasnya. Akurasi dihitung menggunakan rumus:

(3.1)

Di mana *True Positive* adalah jumlah data yang diklasifikasikan dengan tepat.

Input atau masukan pada fungsi ini berupa data *training* yang disimpan dalam *Arraylist* bertipe *Double[],* kromosom seperti Gambar 3.3, dan data *testing* yang juga disimpan dalam *Arraylist* bertipe *Double[]*. *Pseudocode* fungsi menghitung *fitness value* dapat dilihat pada Gambar 3.6.

|  |  |
| --- | --- |
| Masukan | Data *training*, kromosom, dan data *testing* |
| Keluaran | Akurasi |
| 1. for(int i=0; i<nilaiK; i++) 2. centroid[i] = *generate centroid* 3. Menyimpan data c*entroid* 4. end for 5. do 6. Hitung *Euclidian distance* antara data *training* dengan *centroid* 7. Kelompokkan data ke-n dalam *cluster* k sesuai dengan hasil *Euclidian distance* terkecil 8. Update rata-rata *centroid* masing-masing *cluster* 9. until data dalam *cluster* k tetap 10. for(int i=0; i<jumlah data *testing*;i++) 11. Hitung similarity 12. Labeli data *testing* dengan label kelompok yang memilki nilai similarity terbesar 13. end for 14. return akurasi | |
| ***Gambar 3.6. Pseudocode Fungsi Menghitung Fitness Value*** | |

### Fungsi Seleksi

Fungsi seleksi yang digunakan adalah teknik *Elitism,* di mana akan dilakukan pengurutan semua individu dari yang mempunyai nilai akurasi terbesar hingga terkecil. Populasi yang mempunyai nilai akurasi 50 terbesar akan terpilih untuk diikutsertakan pada proses generasi selanjutnya. Proses ini menjamin tidak akan ada solusi yang lebih buruk. Solusi yang didapatkan pada tiap proses nantinya adalah solusi yang sama atau lebih baik dari solusi terbaik sebelumnya. Hal ini tentu saja akan menjamin tersedianya kromosom berkualitas baik untuk generasi selanjutnya. *Pseudocode* fungsi seleksidapat dilihat pada Gambar 3.7.

|  |  |
| --- | --- |
| Masukan | Seluruh individu |
| Keluaran | 50 individu terbaik |
| 1. Hitung nilai akurasi setiap individu 2. Urutkan dari terbesar hingga terkecil 3. Pilih 50 individu terbaik untuk diikutsertakan pada proses selanjutnya | |

***Gambar 3.7.*** ***Pseudocode Fungsi Seleksi***

### Fungsi *Crossover*

Fungsi *crossover*adalah fungsi yang digunakan untuk menghasilkan keturunan baru melalui kawin silang (*crossover*). Dalam fungsi ini, akan dipilih dua individu random sebagai *parent.* Dua *parent* ini memiliki dua kemungkinan, yaitu dikenakan kawin silang dan tidak dikenakan kawin silang. Keputusan untuk dikenakan atau tidak dikenakan berdasarkan pada nilai random [0,1] dan batas *threshold* kawin silang (*crossover rate*). Digunakan *crossover rate* bernilai 0.7. Tipe kawin silang yang digunakan adalah *two-point crossover,* dimana akan dipilih secara random titik awal dan titik akhir bagian dari individu yang dikenakan kawin silang. Ilustrasi *two-point crossover* telah ditunjukkan pada Gambar 2.3. *Pseudocode* fungsi c*rossover*dapat dilihat pada Gambar 3.8.

|  |  |
| --- | --- |
| Masukan | Dua individu sebagai *parent, a*ngka random [0,1] |
| Keluaran | Dua individu baru hasil kawin silang |
| 1. for(int i=0; i<maxIndividu; i++) 2. *Generate* nilai [0,1] 3. if(nilai<*crossover rate*) 4. Tidak terjadi kawin silang 5. Dua individu baru = *parent* 6. else 7. Random bit keberapa yang akan dikenakan kawin silang 8. Dua individu baru = dua individu hasil kawin silang 9. end for | |

*Gambar 3.8. Pseudocode Fungsi Crossover*

### Fungsi Mutasi

Fungsi mutasiadalah fungsi yang digunakan untuk menghasilkan keturunan baru melalui mutasi individu. Akan dilakukan pengacakan nilai secara random [0,1] untuk memutuskan apakah sebuah individu dikenakan mutasi atau tidak. Apabila nilai yang keluar lebih kecil dari threshold mutasi (*mutation rate*), maka individu tidak akan dimutasi, begitu pula sebaliknya. Nilai *mutation rate* yang digunakan adala 0.1 dengan tipe mutasi yang dilakukan adalah *bit inversion*. Ilustrasi *bit inversion* telah ditunjukkan pada Gambar 2.4. *Pseudocode* fungsi mutasidapat dilihat pada Gambar 3.9. dan Gambar 3.9.

|  |  |
| --- | --- |
| Masukan | Individu hasil kawin silang*, a*ngka random [0,1] |
| Keluaran | Dua individu baru hasil mutasi |
| 1. for(int i=0; i<maxPopulation; i++) 2. *Generate* nilai [0,1] 3. if(nilai<*mutation rate*) 4. Tidak terjadi mutasi 5. Individu baru = *parent* | |

*Gambar 3.9. Pseudocode Fungsi Mutasi (1)*

|  |
| --- |
| 1. else 2. Random bit keberapa yang akan dikenakan mutasi 3. Individu baru = individu hasil mutasi 4. end for |

*Gambar 3.10. Pseudocode Fungsi Mutasi (2)*

### Program Utama (*main*)

Program utama atau sering disebut dengan fungsi *main* berisi tentang proses memanggil kembali lima fungsi utama yang telah dijelaskan pada sub-bab sebelumnya, yaitu fungsi*generate* kromosom, fungsi menghitung *fitness value*, fungsi seleksi, fungsi *crossover* danfungsi mutasi**,** ditambah dengan tiga fungsi tambahan yaitu fungsi membaca dataset dalam format excel, fungsi melakukan *crossvalidation,*dan fungsi evaluasi.

*Input* dari program utama adalah dataset kanker *colon.* Berhubung jumlah data/*record* yang sedikit pada dataset ini, maka pengujian menggunakan 5-crossvalidation, dimana lima bagian data awal menjadi data *testing* dan sisanya menjadi data *training*. Pun begitu untuk lima bagian kedua dan seterusnya. *Pseudocode* fungsi *main* dapat dilihat pada Gambar 3.11.

|  |  |
| --- | --- |
| Masukan | data |
| Keluaran | Model dengan akurasi terbaik |
| 1. Memanggil fungsi readFile 2. for(int i=0; i<maxGeneration; i++) 3. Memanggil fungsi generateKromosom 4. Memanggil fungsi processCrossValidation untuk pembagian data *testing* dan data *training* 5. Memanggil fungsi doSelection 6. Memanggil fungsi doCrossover 7. Memanggil fungsi doMutation 8. Memanggil fungsi doEvaluation 9. end for | |

*Gambar 3.11 Pseudocode Fungsi Utama*

# BAB IVZZZZZ IMPLEMENTASI

Pada bab ini akan dibahas mengenai implementasi yang dilakukan berdasarkan rancangan yang telah dijabarkan pada bab sebelumnya. Sebelum penjelasan implementasi akan ditunjukkan terlebih dahulu lingkungan untuk melakukan implementasi.

## Lingkungan Implementasi

Lingkungan implementasi yang akan digunakan untuk melakukan implementasi adalah *Eclipse* yang diinstal pada sistem operasi *Windows* 7.

## Implementasi

Pada subbab ini akan dijelaskan parameter yang digunakan dan implementasi setiap subbab 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.

### Parameter yang Digunakan

Pada tugas akhir ini, penulis menggunakan strategi seleksi *elitism,* di mana akan dipilih 50 individu terbaik berdasarkan nilai *fitness*. Selain itu, penulis juga menggunakan tipe kawin siliang *two-point crossover* dan tipe mutasi *bit-inversion.* Parameter yang digunakan dapat dilihat pada Tabel 4.1*.*

*Tabel 4.1. Tabel Parameter yang Digunakan*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Parameter GA [19]** | **Nilai** | **Parameter *K-Means*** | **Nilai** | **Parameter K-NN** | **Nilai** |
| Jumlah populasi | 100 | Minimal *Cluster* | 2 | Jumlah tetangga terdekat | 1 |
| Jumlah generasi | 10 |
| *Crossover rate* | 0.7 |
| *Mutation rate* | 0.1 | Maksimal *Cluster* |  |
| Individu yang dipilih (elitism) | 50 |
| Jumlah *running* percobaan | 10 |

### Implementasi Fungsi readFile()

Fungsi **readFile()** adalah fungsi untuk membaca data dari Microsoft Excel yang kemudian akan disimpan dalam variabel dataset bertipe ArrayList<Double []>. Implementasi fungsi ini dapat dilihat pada Kode Sumber 4.1 dan Kode Sumber 4.2.

|  |  |
| --- | --- |
|  | public static int readFile() throws FileNotFoundException |
|  | { |
|  | FileInputStream fstream =  new FileInputStream( |
|  | + "D:\\""KULIAH\\SEMESTER 7\\TA\\" |
|  | + "cobaHubJavaMatlab\\cobaHubJavaMatlab" |
|  | + "\\dataResult.csv"); |

*Kode Sumber 4.1. Kode Sumber Fungsi readFile (1)*

|  |  |
| --- | --- |
|  | DataInputStream in =  new DataInputStream(fstream); |
|  | BufferedReader br =  new BufferedReader  (new InputStreamReader(in)); |
|  | String strLine = null; |
|  | String[] hasilSplit; |
|  |  |
|  | try { |
|  | while ((strLine = br.readLine()) != null){ |
|  | hasilSplit = strLine.split(","); |
|  | jmlAtribut = hasilSplit.length; |
|  | Double[] hasilSplit2 =  new Double[jmlAtribut]; |
|  | for(int i=0; i<jmlAtribut; i++) |
|  | { |
|  | hasilSplit2[i]=Double.parseDouble  (hasilSplit[i]); |
|  | } |
|  | dataset.add(hasilSplit2); |
|  | } |
|  | } catch (IOException ex) { |
|  | Logger.getLogger(main.class.getName()).  log(Level.SEVERE, null, ex); |
|  | } |

*Kode Sumber 4.2. Kode Sumber Fungsi readFile (2)*

Fungsi utama untuk membaca file dari Microsoft Excel terdapat pada baris 4 sampai 22. Baris 30 menjelaskan tentang pembagian jumlah data per bagiannya apabila dilakukan *5-crossvalidation*.

### Implementasi Fungsi generateKromosom()

Fungsi **generateKromosom()** digunakan untuk me-*generate* bilangan biner random sebanyak 61 bit (sebanyak jumlah atribut). Bilangan biner ini pada akhirnya akan merepresentasikan sebuah individu. Atribut yang dikenai nilai 1 akan digunakan dalam perhitungan, begitu pula sebaliknya. Implementasi fungsi **generateKromosom()** dapat dilihat pada Kode Sumber 4.3.

|  |  |
| --- | --- |
| 1 | public static void generateKromosom(){ |
| 2 | individu = new byte[geneLength]; |
| 3 | for(int i=0; i<individu.length; i++){ |
| 4 | byte gen = (byte)Math.round(Math.random()); |
| 5 | individu[i]=gen; |
| 6 | } |
| 7 | } |

*Kode Sumber 4.3. Kode Sumber Fungsi generateKromosom()*

### Implementasi Fungsi processCrossValidation()

Fungsi **processCrossValidation()** digunakan sebagai metode pengelompokkan data *training* dan data *testing*. Setelah mendapatkan banyak data per bagiannya pada Kode Sumber 4.2, maka 5 bagian pertama akan digunakan sebagai data *testing*, sedangkan sisanya sebagai data *training*. Setelah itu, secara bergantian 5 bagian kedua digunakan sebagai data *testing*, sisanya sebagai data *training*. Begitu seterusnya hingga 5 bagian terakhir digunakan sebagai data *testing* dan sisanya sebagai data *training*. Implementasi fungsi **processCrossValidation()** dapat dilihat pada Kode Sumber 4.4, Kode Sumber 4.5, Kode Sumber 4.6, dan Kode Sumber 4.7.

|  |  |
| --- | --- |
|  | public static void processCrossValidation(){ |
|  | tempAcc = new Integer[5]; |
|  | for(int i=0; i<crossval; i++){ |
|  | dataTesting.add(dataset.get(i)); |
|  | } |
|  |  |
|  | for(int i=crossval; i<dataset.size(); i++){ |
|  | if(dataset.get(i)[jmlAtribut-1]==1) |
|  | trainingPositive.add(dataset.get(i)); |
|  | else |
|  | trainingNegative.add(dataset.get(i)); |

*Kode Sumber 4.4. Kode Sumber Fungsi processCrossValidation (1)*

|  |  |
| --- | --- |
|  | } |
|  | calculateFitFunc(nilaiK); |
|  | tempAcc[0] = acc; |
|  |  |
|  | for(int i=crossval; i<2\*crossval; i++){ |
|  | dataTesting.add(dataset.get(i)); |
|  | } |
|  |  |
|  | for(int i=0; i<crossval; i++){ |
|  | if(dataset.get(i)[jmlAtribut-1]==1) |
|  | trainingPositive.add(dataset.get(i)); |
|  | else |
|  | trainingNegative.add(dataset.get(i)); |
|  | } |
|  |  |
|  | for(int i=2\*crossval; i<dataset.size(); i++){ |
|  | if(dataset.get(i)[jmlAtribut-1]==1) |
|  | trainingPositive.add(dataset.get(i)); |
|  | else |
|  | trainingNegative.add(dataset.get(i)); |
|  | } |
|  | calculateFitFunc(nilaiK); |
|  | tempAcc[1] = acc; |
|  |  |
|  | for(int i=2\*crossval; i<3\*crossval; i++){ |
|  | dataTesting.add(dataset.get(i)); |
|  | } |
|  |  |
|  | for(int i=0; i<2\*crossval; i++){ |
|  | if(dataset.get(i)[jmlAtribut-1]==1) |
|  | trainingPositive.add(dataset.get(i)); |
|  | else |
|  | trainingNegative.add(dataset.get(i)); |
|  | } |
|  | for(int i=0; i<2\*crossval; i++){ |
|  |  |
|  | for(int i=3\*crossval; i<dataset.size(); i++){ |
|  | if(dataset.get(i)[jmlAtribut-1]==1) |
|  | trainingPositive.add(dataset.get(i)); |

*Kode Sumber 4.5. Kode Sumber Fungsi processCrossValidation (2)*

|  |  |
| --- | --- |
|  | else |
|  | trainingNegative.add(dataset.get(i)); |
|  | } |
|  | calculateFitFunc(nilaiK); |
|  | tempAcc[2] = acc; |
|  |  |
|  | for(int i=3\*crossval; i<4\*crossval; i++){ |
|  | dataTesting.add(dataset.get(i)); |
|  | } |
|  |  |
|  | for(int i=0; i<3\*crossval; i++){ |
|  | if(dataset.get(i)[jmlAtribut-1]==1) |
|  | trainingPositive.add(dataset.get(i)); |
|  | else |
|  | trainingNegative.add(dataset.get(i)); |
|  | } |
|  |  |
|  | for(int i=4\*crossval; i<dataset.size(); i++){ |
|  | if(dataset.get(i)[jmlAtribut-1]==1) |
|  | trainingPositive.add(dataset.get(i)); |
|  | else |
|  | trainingNegative.add(dataset.get(i)); |
|  | } |
|  | calculateFitFunc(nilaiK); |
|  | tempAcc[3] = acc; |
|  |  |
|  | for(int i=4\*crossval; i<dataset.size(); i++){ |
|  | dataTesting.add(dataset.get(i)); |
|  | } |
|  |  |
|  | for(int i=0; i<4\*crossval; i++){ |
|  | if(dataset.get(i)[jmlAtribut-1]==1) |
|  | trainingPositive.add(dataset.get(i)); |
|  | else |
|  | trainingNegative.add(dataset.get(i)); |
|  | } |
|  |  |
|  | calculateFitFunc(nilaiK); |
|  | tempAcc[4] = acc; |

*Kode Sumber 4.6. Kode Sumber Fungsi processCrossValidation (3)*

|  |  |
| --- | --- |
|  | acc = 0; |
|  | for(int i=0; i<tempAcc.length; i++){ |
|  | acc += tempAcc[i]; |
|  | } |
|  | accuracy = ((double) acc/ (double)dataset.size())\*100; |
|  | acc = 0; |
|  | } |

*Kode Sumber 4.7. Kode Sumber Fungsi processCrossValidation (4)*

### Implementasi Fungsi calculateFitFunc()

Fungsi **calculateFitFunc()** digunakan untuk menghitung akurasi tiap individu yang dihasilkan. Pada fungsi ini, akan diimplementasikan algoritma *K-Means* dan *K-Nearest Neighbor* dalam perhitungan akurasinya. Pada algoritma *K-Means,* langkah pertama yang harus dilakukan adalah memilih data sebagai centroid. Proses tersebut diimplementasikan pada Kode Sumber 4.8.

|  |  |
| --- | --- |
|  | Random rands = new Random(); |
|  | int total\_elements\_cnt = 0; |
|  | boolean loop\_status = true; |
|  |  |
|  | while(loop\_status){ |
|  | int next\_num = rand.nextInt(dataP-1); |
|  | if(!isCompleted()){ |
|  | if(!isDuplicated(next\_num)){ |
|  | centroidP[total\_elements\_cnt] = next\_num; |
|  | total\_elements\_cnt++; |
|  | } |
|  | else |
|  | continue; |
|  | } |
|  | else |
|  | loop\_status = false; |
|  | } |

*Kode Sumber 4.8. Kode Sumber Pemilihan Data sebagai Centroid*

Langkah selanjutnya adalah menghitung kedekatan data dengan seluruh *centroid*. Pengimplementasian rumus *Euclidian Distance* untuk menghitung jarak ditunjukkan pada Kode Sumber 4.9.

|  |  |
| --- | --- |
|  | for(int i=0; i<trainingPositive.size(); i++){ |
|  | for(int j=0; j<centroidP.length; j++){ |
|  | for(int l=0; l<jmlAtribut-1; l++){ |
|  | if(individu[l]==1){ |
|  | tampung += Math.pow((trainingPositive.get(i)[l] - meanPerClusterP.get(j)[l]),2); |
|  | } |
|  | } |
|  | distance = Math.sqrt(tampung); |
|  | distPositiveClass[i][j] = distance; |
|  | tampung=0; distance=0; |
|  | } |
|  | } |

*Kode Sumber 4.9. Kode Sumber Perhitungan Nilai Disimilarity*

Perlu dilakukan pengecekkan anggota setiap klaster sebagai acuan berhentinya proses *K-Means.* Apabila anggota dalam klaster saat ini sama dengan iterasi sebelumnya, maka proses berhenti. Implementasi pengecekkan anggota klaster ditunjukkan pada Kode Sumber 4.10.

|  |  |
| --- | --- |
|  | for(int i=0; i<trainingPositive.size(); i++){ |
|  | double temp = distPositiveClass[i][0]; |
|  | for(int j=0; j<centroidP.length; j++){ |
|  | if(distPositiveClass[i][j]<=temp){ |
|  | temp= distPositiveClass[i][j]; |
|  | data = i; |
|  | inCluster = j; |
|  | } |
|  | } |
|  | labelDataP[data] = inCluster; |
|  | } |

*Kode Sumber 4.10. Kode Sumber Pengecekkan Anggota Cluster*

Sesuai dengan Kode Sumber 4.10, apabila anggota klaster saat ini tidak sama dengan iterasi sebelumnya, maka proses akan berjalan terus. Ketika proses berlanjut ke iterasi selanjutnya, perlu dilakukan peng-*update*-an nilai centroid. Implementasi peng-*update*-an nilai centroid ditunjukkan pada Kode Sumber 4.11.

|  |  |
| --- | --- |
|  | int count=0; |
|  | double tempMeanP=0; |
|  |  |
|  | for(int i=0; i<numb; i++){ |
|  | for(int j=0; j<jmlAtribut-1; j++){ |
|  | (int l=0; l<labelDataP.length; l++){ |
|  | if(labelDataP[l]==i){ |
|  | if(individu[j]==1){ |
|  | tempMeanP += trainingPositive.get(l)[j]; |
|  | } |
|  | count++; |
|  | } |
|  | else{ |
|  | tempMeanP += 0; |
|  | } |
|  | } |
|  | tempMeanPerAtribut[j] = (tempMeanP/count); |
|  | tempMeanP = 0; |
|  | count = 0; |
|  | } |
|  | tempMeanPerAtribut[jmlAtribut-1] = 1.0; |
|  | meanPerClusterP2.add(tempMeanPerAtribut); |
|  | meanPerClusterP.add(meanPerClusterP2.get(i).clone()); |
|  | } |

*Kode Sumber 4.11. Kode Sumber Perbarui Nilai Centroid*

Kode Sumber 4.8, Kode Sumber 4.9, Kode Sumber 4.10, dan Kode Sumber 4.11. adalah implementasi algoritma *K-Means* sebagai proses awal perhitungan nilai *fitness.* Keempat kode sumber tersebut hanya contoh pengimplementasian untuk kelompok data *training* yang berlabel positif saja. Perlu dilakukan pengimplementasi untuk kelompok data *training* yang berlabel negatif.

Setelah itu, akan dibandingkan antara *actual class* dengan *predicted class*-nya sesuai dengan perhitungan jarak (implementasi perhitungan jarak pada Kode Sumber 4.9). Pengimplementasian perhitungan akurasi pada *K-Nearest Neighbor* ditunjukkan pada Kode Sumber 4.12

|  |  |
| --- | --- |
|  | double a=0; int b=0; |
|  | for(int i=0; i<dataTesting.size();i++){ |
|  | a = dist[i][0]; |
|  | for(int j=0; j<clusterMix; j++){ |
|  | if(dist[i][j]<=a){ |
|  | a = dist[i][j]; |
|  | b=j; |
|  | } |
|  | } |
|  | if(b<meanPerClusterP.size()){ |
|  | predictClass[i] = 1.0; |
|  | } |
|  | else |
|  | predictClass[i] = 0.0; |
|  | } |
|  | acc=0; |
|  | for(int i=0; i<actualClass.length; i++){ |
|  | if(actualClass[i] == predictClass[i]){ |
|  | acc++; |
|  | } |
|  | } |

*Kode Sumber 4.12. Kode Sumber Perhitungan Akurasi*

Kembali ke algoritma *K-Means, s*aat pemilihan *centroid* awal, diharapkan tidak ada duplikasi data yang terpilih. Artinya apabila terdapat dua klaster dan data ke-1 terpilih sebagai *centroid* klaster pertama, maka dia tidak akan boleh terpilih sebagai *centroid* klaster kedua. Proses tersebut dicek menggunakan fungsi **isCompleted()** dan **isDuplicated().** Implementasi fungsi **isCompleted()** dan **isDuplicated()** dapat dilihat pada **Error! Reference source not found.**.

|  |  |
| --- | --- |
|  | public static boolean isCompleted(){ |
|  | boolean status = true; |
|  | for (int i = 0; i < centroidP.length; i++){ |
|  | if(centroidP[i]==0){ |
|  | status = false; |
| 1. 6 | break;}} |
| 1. 9 | return status; |
|  | } |
|  |  |
|  | public static boolean isDuplicated(int num){ |
|  | boolean status = false; |
|  | for (int i = 0; i < centroidP.length; i++){ |
|  | if(centroidP[i]== num){ |
|  | status = true; |
|  | break;}} |
|  | return status; |
|  | } |

*Kode Sumber 4.13. Kode Sumber Fungsi isCompleted() dan isDuplicated(2)*

Pada saat dilakukannya pengklasteran menggunakan *K-Means,* iterasi akan berhenti apabila komponen penyusun suatu klaster tidak akan berubah lagi. Untuk melakukan proses pengecekkan tersebut, maka dibuatlah sebuah fungsi **check().** Implementasi fungsi **check()** dapat dilihat pada Kode Sumber 4.14.

|  |  |
| --- | --- |
| 1 | static int check() |
| 2 | { |
| 3 | for(int i=0;i<labelDataP.length; i++) |
| 4 | { |
| 5 | if(tempLabelDataP[i]!=labelDataP[i]) |
| 6 | return 0; |
| 7 | } |
| 8 | return 1; |
| 9 | } |

*Kode Sumber 4.14. Kode Sumber Fungsi Check()*

### Implementasi Fungsi doSelection()

Implementasi fungsi **doSeelection()** digunakan untuk memilih 50 individu terbaik untuk diikutsertakan dalam proses kawin silang (crossover). Implementasi fungsi **doSelection()** dapat dilihat pada Kode Sumber 4.15.

|  |  |
| --- | --- |
| 1 | public static void doSelection(){ |
| 2 | for(int i=0; i<50; i++){ |
| 3 | selectedPop.add(populationSort.get(i)); |
| 4 | } |
| 5 | } |

*Kode Sumber 4.15. Kode Sumber Fungsi doSelection()*

### Implementasi Fungsi doCrossOver()

Implementasi fungsi **doCrossOver()** digunakan untuk mengawinkan dua individu sebagai *parent*. *Parent* dapat melakukan kawin silang ataupun tidak melakukan kawin silang tergantung dari hasil random apakah di atas/di bawah nilai *croossoverRate.* Kondisi dua individu yang tidak melakukan kawin silang terdapat pada Kode Sumber 4.16.

|  |  |
| --- | --- |
|  | Double rcross = Math.random(); |
|  | if(rcross<crossoverRate){ |
|  | offspring.add(new Individu(selectedPop.get(temp[i]).getIndex(),selectedPop.get(temp[i]).getKromosom(),selectedPop.get(temp[i]).getKromosomK(),selectedPop.get(temp[i]).getAccuracy())); |
|  | offspring.add(new Individu(selectedPop.get(temp[i+1]).getIndex(),selectedPop.get(temp[i+1]).getKromosom(),selectedPop.get(temp[i+1]).getKromosomK(),selectedPop.get(temp[i+1]).getAccuracy())); |
|  | } |

*Kode Sumber 4.16. Kode Sumber Individu yang Tidak Dikenai Crossover*

Sedangkan dua individu yang dapat melakukan kawin silang ditunjukkan pada Kode Sumber 4.17.

|  |  |
| --- | --- |
|  | else{ |
|  | Random rand = new Random(); |
|  | int poin[] = new int[2]; |
|  | poin[0] = rand.nextInt(individu.length)+1; |
|  | if(poin[0]==61){ |
|  | poin[1]=61; |
|  | } |
|  | else |
|  | poin[1]=poin[0] + rand.nextInt(individu.length-poin[0]); |
|  |  |
|  | byte[] tempInd = new byte[geneLength]; |
|  | int tempK=0, a=0, b=0; |
|  |  |
|  | a = temp[i]; |
|  | b = temp[i+1]; |
|  | tempInd=selectedPop.get(a).getKromosom().clone(); |
|  | tempK = selectedPop.get(b).getKromosomK(); |
|  |  |
|  | for(int j=(poin[0])-1; j<poin[1]; j++){ |
|  | tempInd[j]=selectedPop.get(b).getKromosom()[j]; |
|  | } |
|  | offspring.add(new Individu(selectedPop.get(a).getIndex(),tempInd,tempK,0)); |
|  | tempInd=selectedPop.get(b).getKromosom().clone(); |
|  | tempK = selectedPop.get(a).getKromosomK(); |
|  |  |
|  | for(int j=(poin[0])-1; j<poin[1]; j++){ |
|  | tempInd[j]=selectedPop.get(a).getKromosom()[j]; |
|  | } |
|  | offspring.add(new Individu(selectedPop.get(b).getIndex(),tempInd,tempK,0));}} |
|  | } |
|  | } |

*Kode Sumber 4.17. Kode Sumber Individu yang Dikenai Crossover*

### Implementasi Fungsi doMutation()

Implementasi fungsi **doMutation()** digunakan untuk menentukan apakah sebuah individu melakukan mutasi atau tidak. Implementasi fungsi **doMutation()** dapat dilihat pada Kode Sumber 4.18. Kondisi individu yang tidak dikenai mutasi terdapat pada baris 5 sampai 7. Sedangkan individu yang dikenai mutasi terdapat pada baris 8 sampai 35.

|  |  |
| --- | --- |
| 1 | public static void doMutation() |
| 2 | { |
| 3 | Double rmut = Math.random(); |
| 4 | if(rmut<mutationRate){ |
| 5 | //do nothing |
| 6 | } |
| 7 | else{ |
| 8 | Random rand = new Random(); |
|  | int m = rand.nextInt(geneLength)+1; |
|  | byte[] tempInd = new byte[geneLength]; |
|  | tempInd = offspring.get(i+2).getKromosom().clone(); |
|  | if(tempInd[m-1]==0){ |
|  | tempInd[m-1]=1; |
|  | } |
|  | else |
|  | tempInd[m-1]=0; |
|  |  |
|  | offspring.get(i+2).setKromosom(tempInd); |
|  | rand = new Random(); |
|  | int maxK = (int) Math.round(Math.sqrt((dataset.size())/2)); |
|  | int range = maxK - 2 + 1; |
|  | m = rand.nextInt(range) + 2; |
|  | while(Math.abs(m-offspring.get(i+2).getKromosomK())>2){ |
|  | rand = new Random(); |
|  | m = rand.nextInt(range)+2;} |
|  | offspring.get(i+2).setKromosomK(m); |
|  | offspring.get(i+2).setAccuracy(0.0);} |
|  | } |

*Kode Sumber 4.18. Kode Sumber Fungsi doMutation()*

### Implementasi Fungsi doEvaluation()

Implementasi fungsi **doEvaluation()** digunakan untuk mengevaluasi populasi yang terbentuk dan memilih lagi 50 populasi terbaik untuk diikutsertakan di generasi selanjutnya. Implementasi fungsi **doEvaluation()** dapat dilihat pada Kode Sumber 4.19.

|  |  |
| --- | --- |
| 1 | public static void doEvaluation(){ |
| 3 | nilaiK = 0; |
| 4 | for(int g=0; g<(offspring.size()-2); g++){ |
| 5 | int k = offspring.get(g+2).getKromosomK(); |
| 6 |  |
| 7 | processCrossValidation(); |
| 8 |  |
|  | Collections.sort(offspring, new Individu.CompAcc()); |
|  |  |
|  | population.clear(); |
|  |  |
|  | if(e==maxGeneration-1){ |
|  | population.add(offspring.get(0)); |
|  | } |
|  | else{ |
|  | for(int j=0; j<50; j++){ |
|  | population.add(offspring.get(j)); |
|  | } |
|  | } |
|  |  |
|  | printPopulation(); |
|  | offspring.clear(); |
|  | selectedPop.clear(); |
|  | } |

*Kode Sumber 4.19. Kode Sumber Fungsi doEvaluation()*

### Implementasi Program Utama

Program utama yang dibuat adalah program *callback* dimana dalam program utama ini hanya terdapat pemanggilan fungsi-fungsi yang telah dibuat sebelumnya. Implementasi fungsi utama dapat dilihat pada Kode Sumber 4.20.

|  |  |
| --- | --- |
| 1 | public static void main(String[] args) throws FileNotFoundException { |
| 2 |  |
| 3 | long startTime = System.currentTimeMillis(); |
| 4 | readFile(); |
| 6 | //Genetic Algorithm |
| 7 | for(e=0; e<maxGeneration; e++){ |
|  | if(e==0){ |
|  | for(int pop=0; pop<maxPopulation; pop++) |
|  | { |
|  | generateKromosom(); |
|  | Random rand = new Random(); |
|  | Int maxK = (int) Math.round(Math.sqrt((dataset.size())/2)); |
|  | int range = maxK - 2 + 1; |
|  | nilaiK = rand.nextInt(range) + 2; |
|  | processCrossValidation(); |
|  | population.add(new Individu(pop,individu,nilaiK,accuracy)); |
|  | } |
|  | } |
|  | populationSort = (ArrayList<Individu>) population.clone(); |
|  | Collections.sort(populationSort, new Individu.CompAcc()); |
|  |  |
|  | doSelection(); |
|  | doCrossover(); |
|  | doMutation(); |
|  | doEvaluation(); |
|  | System.out.println(""); |
|  | } |
|  | long endTime = System.currentTimeMillis(); |
|  | long totalTime = endTime - startTime; |
|  | System.out.println("Running time : "+totalTime+" ms"); |
|  | } |

*Kode Sumber 4.20. Kode Sumber Program Utama*

# BAB V UJI COBA DAN EVALUASI

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



## Lingkungan Uji Coba

Lingkungan uji coba menjelaskan lingkungan yang digunakan untuk menguji implementasi metode gabungan *Genetic Algorithm, K-Means,* dan *K-Nearest Neighbor* pada Tugas Akhir ini. Lingkungan uji coba meliputi perangkat keras dan perangkat lunak yang dijelaskan sebagai berikut:

Perangkat keras

Prosesor: Intel® Core™ i3-23100M CPU @ 2.10GHz

*Memory*(RAM): 4,00 GB

Tipe sistem: 32-bit sistem operasi

Perangkat lunak

Sistem operasi: *Windows* 7 *Home Professional.*

Perangkat pengembang: *Eclipse*.

## Data *Training* dan Data *Testing*

Pembagian data *training* dan data *testing* menggunakan metode *5-crossvalidation*, di mana data keseluruhan dibagi menjadi 5 bagian. Visualisasi *5-crossvalidation* dapat dilihat pada Gambar 5.1.

Pada *crossvalidation* yang pertama, bagian pertama dari data pasien akan digunakan sebagai data *testing*, sedangkan sisanya menjadi data *training*. Setelah itu, akan dilakukan proses pengklasteran menggunakan *K-Means* pada data *training* yang sebelumnya sudah dipisahkan antara kelompok data positif dan negatif. Data *testing* diuji kedekatan kelasnya antar kelompok menggunakan *K-Nearest Neighbor* yang kemudian data tersebut akan dilabeli dengan label kelompok terdekat. Lalu, dari kelas yang diprediksi dihitung berapa jumlah data yang berlabel sama dengan kelas. Pada *crossvalidation* yang kedua, bagian kedua dari data pasien akan digunakan sebagai data *testing*, sedangkan sisanya menjadi data *training*. Begitu terus hingga semua bagian data pasien pernah digunakan sebagai data *testing*. Hasil dari tahap *crossvalidation* ini adalah jumlah data yang berlabel sama dengan kelas sebenarnya yang kemudian dibagi dengan seluruh data untuk mendapatkan akurasi.

62 Data Pasien

12 Data Pasien

12 Data Pasien

12 Data Pasien

12 Data Pasien

14 Data Pasien

Data *Testing*

Data *Training*

*Gambar 5.1. Visualisasi 5-Crossvalidation*

## 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 terdiri dari 3 pengujian yaitu:

Skenario perhitungan akurasi dan *running time* metode gabungan GA, *K-Means*, dan *K-Nearest Neighbor* dengan menggunakan PCA sebagai reduksi dimensi fitur.

Skenario perhitungan akurasi dan *running time* metode gabungan GA, *K-Means*, dan *K-Nearest Neighbor* tanpa menggunakan PCA.

Skenario perhitungan akurasi dan *running time* metode klasifikasi (*K-Nearest Neighbor*) biasa dengan beberapa percobaan parameter *k* dan menggunakan PCA sebagai reduksi dimensi fitur.

### Skenario Uji Coba 1

Skenario uji coba 1 adalah perhitungan akurasi dan *running time* metode gabungan GA, *K-Means*, dan *K-Nearest Neighbor* dengan menggunakan PCA sebagai reduksi dimensi fitur. Penulis mengambil hanya 61 atribut saja dari total 2000 atribut. Hal ini didasarkan pada nilai kumulatif varians yang menunjukkan bahwa ketika 61 atribut saja yang digunakan, maka data tetap akan mempunyai 99,98% informasi atau hanya kehilangan 0,02% informasi saja. Setelah itu data yang sudah direduksi fiturnya akan masuk ke dalam proses GA. Parameter-parameter yang digunakan pada proses GA telah dijelaskan pada Tabel 4.1.

Setelah dilakukan percobaan, maka harus dilakukan evaluasi untuk melihat sejauh mana kesesuaian program dengan data uji yang disediakan. Untuk menentukan kesesuaian keluaran dengan program dan analisa yang telah dibuat diperlukan pengujian akurasi dan *running time*. Perhitungan akurasi dilakukan menggunakan persamaan 3.1. Gambar 5.2 adalah grafik hasil akurasi skenario uji coba 1. Grafik hasil *running time* skenario uji coba 1 dapat dilihat pada Gambar 5.3.

*Gambar 5.2. Grafik Hasil Akurasi Skenario Uji Coba 1*

*Gambar 5.3. Grafik Hasil Running Time Skenario Uji Coba 1*

Pada akhir proses percobaan, didapatkan rata-rata nilai akurasi sebesar 89.52% dengan rata-rata hasil *running time* sebesar 46.92 sekon. Dalam 10 kali proses *running,* model terbaik yang pernah terbentuk adalah diikutsertakannya 39 atribut, yaitu atribut ke-1, 2, 3, 4, 5, 6, 7, 10, 11, 12, 13, 15, 16, 17, 22, 24, 25, 26, 28, 29, 30, 31, 32, 33, 35, 36, 37, 38, 39, 44, 45, 47, 50, 52, 53, 54, 56, 57, dan 58 dengan nilai akurasi sebesar 93.55% dalam waktu 59.12 sekon. Hasil skenario uji coba 1 secara lengkap dapat dilihat pada LAMPIRAN.

### Skenario Uji Coba 2

Skenario uji coba 2 merupakan perhitungan akurasi metode GA, *K-Means*, dan *K-Nearest Neighbor* tanpa menggunakan PCA. Dataset yang digunakan sebagai masukan program adalah dataset dengan 2000 atribut dan 62 sampel.

Setelah dilakukan percobaan, maka harus dilakukan evaluasi untuk melihat sejauh mana kesesuaian program dengan data uji yang disediakan. Untuk menentukan kesesuaian keluaran dengan program dan analisa yang telah dibuat diperlukan pengujian akurasi. Pengujian akurasi dapat dilakukan dengan melihat kesesuaian hasil keluaran pada program dengan hasil *actual class* nya. Perhitungan akurasi dilakukan menggunakan persamaan 3.1. Grafik hasil akurasi skenario uji coba 2 dapat dilihat pada Gambar 5.4. dan grafik hasil *running time* skenario uji coba 2 dapat dilihat pada Gambar 5.5.

Pada akhir proses percobaan, didapatkan rata-rata nilai akurasi sebesar 92.90% dengan rata-rata hasil *running time* sebesar 31.12 menit. Dalam 10 kali proses *running,* akurasi terbaik yang pernah didapat sebesar 95.16% dalam waktu 30.97 menit. Hasil skenario uji coba 2 secara lengkap dapat dilihat pada LAMPIRAN

*Gambar 5.4. Grafik Hasil Akurasi Skenario Uji Coba 2*

*Gambar 5.5. Grafik Hasil Running Time Skenario Uji Coba 2*

### Skenario Uji Coba 3

Skenario uji coba 3 merupakan perhitungan akurasi dan *running time* metode klasifikasi (*K-Nearest Neighbor*) biasa dengan beberapa percobaan parameter *k* dan menggunakan PCA sebagai reduksi dimensi fitur. Dataset yang digunakan adalah dataset yang telah direduksi dimensi fiturnya sehingga tersisa hanya 61 atribut dan 62 sampel.

Setelah dilakukan percobaan, maka harus dilakukan evaluasi untuk melihat sejauh mana kesesuaian program dengan data uji yang disediakan. Untuk menentukan kesesuaian keluaran dengan program dan analisa yang telah dibuat diperlukan pengujian akurasi. Pengujian akurasi dapat dilakukan dengan melihat kesesuaian hasil keluaran pada program dengan hasil *actual class* nya. Grafik hasil akurasi skenario uji coba 2 dapat dilihat pada Gambar 5.6. dan grafik hasil running time skenario uji coba 2 dapat dilihat pada Gambar 5.7.

Pada akhir proses percobaan, didapatkan rata-rata nilai akurasi sebesar 77.42% dengan rata-rata hasil *running time* sebesar 1.17 sekon. Dari 10 variasi parameter *k* yang digunakan, didapatkan nilai akurasi terbaik yaitu 87.10% dengan waktu 1.34 sekon pada saat nilai *k*=1.

*Gambar 5.6. Grafik Hasil Akurasi Skenario Uji Coba 3*

*Gambar 5.7. Grafik Hasil Running Time Skenario Uji Coba 3*

## Analisis Hasil Uji Coba

Setelah dihitung rata-rata akurasi, pertanyaan yang sering muncul adalah apakah nilai rata-rata akurasi tersebut betul-betul representasi data yang baik dari akurasi hasil percobaan yang telah dilakukan. Oleh karena itu perlu pula dilakukan perhitungan standar deviasi untuk mengetahui rentang atau variasi hasil. Adapun rumus untuk menghitung standar deviasi adalah sebagai berikut:

|  |  |
| --- | --- |
|  | (5.1) |

dimana *S* adalah standar deviasi, *mean* adalah rata-rata data, dan *n* adalahjumlah data. Standar deviasi dapat menggambarkan seberapa jauh bervariasinya data. Jika nilai standar deviasi jauh lebih besar dari *mean,* maka nilai *mean* kurang merepresentasikan performa untuk semua data. Sedangkan jika nilai standar deviasi sangat kecil dibandingkan dengan *mean,* maka nilai mean dapat digunakan sebagai representasi dari keseluruhan data. Nilai standar deviasi dari hasil akurasi masing-masing skenario ditunjukkan pada Tabel 5.1.

*Tabel 5.1. Tabel Perbandingan Hasil Akurasi dan Running Time*

|  |  |  |  |
| --- | --- | --- | --- |
|  | Skenario Uji Coba 1 | Skenario Uji Coba 2 | Skenario Uji Coba 3 |
| Mean Akurasi | 89.52% | 92.90% | 77.42% |
| Standar Deviasi Akurasi | 2.18 | 1.13 | 4.81 |
| Running Time | 46.92 s | 31.129 m | 1.17 s |

Berdasarkan Tabel 5.1., nilai standar deviasi yang dihasilkan sangat jauh lebih kecil daripada *mean.* Hal itu menunjukkan bahwa dalam kasus ini, *mean* merupakan representasi yang baik dari hasil percobaan. Selain itu, standar deviasi juga menyimpan informasi kefluktuatifan data. Semakin tinggi nilai standar deviasi, maka dapat dikatakan data semakin fluktuatif. Di antara tiga skenario yang diujicobakan, maka tingkat akurasi pada skenario uji coba 3 paling fluktuatif di antara dua skenario yang lain. Ditunjukkan pula bahwa skenario 1 mampu mereduksi waktu *running* hampir 40x lebih cepat dibandingkan dengan skenario uji coba 2.

***[Halaman ini sengaja dikosongkan]***

# BAB VIZZZZZ 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.

## Kesimpulan

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

1. Metode gabungan *Genetic Algorithm, K-Means,* dan *K-Nearest Neighbor* dengan menggunakan PCA untuk mereduksi dimensi fitur dapat menghasilkan model terbaik untuk pengklasifikasian penyakit kanker *colon.*
2. Metode gabungan *Genetic Algorithm, K-Means,* dan *K-Nearest Neighbor* dengan menggunakan PCA untuk mereduksi dimensi fitur mampu menghasilkan model dengan tingkat akurasi 89.52% dalam waktu 46.92 sekon. Hal ini menunjukkan bahwa metode tersebut mampu mengurangi waktu *running* secara signifikan dengan selisih akurasi hanya 3.38%.
3. Akurasi terbaik yang didapatkan adalah 93.55% dengan pengikutsertaan 39 atribut, yaitu atribut ke-1, 2, 3, 4, 5, 6, 7, 10, 11, 12, 13, 15, 16, 17, 22, 24, 25, 26, 28, 29, 30, 31, 32, 33, 35, 36, 37, 38, 39, 44, 45, 47, 50, 52, 53, 54, 56, 57, dan 58.

## Saran

Saran yang diberikan untuk pengembangan aplikasi ini adalah:

1. Untuk meningkatkan nilai akurasi, dapat dicoba untuk mengubah parameter GA yang digunakan, seperti jumlah populasi dan generasi.
2. Ada baiknya dibuatkan *user interface* yang memudahkan pengguna dalam memasukkan data dan melihat hasil keluarannya.

# DAFTAR PUSTAKA

|  |  |
| --- | --- |
| [1] | "Kanker Usus Besar," 27 Juni 2013. [Online]. Available: http://id.wikipedia.org/wiki/Kanker\_usus\_besar. [Accessed 9 September 2014]. |
| [2] | "Kanker Usus Besar," [Online]. Available: http://prodia.co.id/penyakit-dandiagnosa/. [Accessed 9 September 2014]. |
| [3] | B. Al-Shboul and S.-H. Myaeng, "Initializing K-Means using Genetic Algorithm," *World Academy of Science, Engineering and Technology,* vol. 54, 2009. |
| [4] | K.-j. Kim and H. Ahn, "A recommender system using GA K-means clustering," *Expert Systems with Applications,* vol. 34, pp. 1200-1209, 2008. |
| [5] | C. V. and N. Sumathi, "A Modified Genetic Algorithm Initializing K-Means Clustering," *Global Journal of Computer Science and Technology,* vol. 11, no. 2, 2011. |
| [6] | F. K. Wardhani, E. Suryani and A. Mukhlason, "Penerapan metode GA-Kmeans untuk pengelompokan pengguna pada Bapersip Provinsi Jawa Timur," *JURNAL TEKNIK POMITS,* vol. 1, no. 1, pp. 1-6, 2012. |
| [7] | M. A. Jabbar, B. L. Deekshatulu and P. Chandra, "Classification of Heart Disease Using K- Nearest Neighbor and," *International Conference on Computational Intelligence: Modeling Techniques and Applications,* vol. 10, pp. 85-94, 2013. |
| [8] | "Gejala Kanker Usus Besar (Kolon), Penyebab dan Pencegahan," [Online]. Available: http://gejalapenyakitmu.blogspot.com/2013/07/gejala-kanker-ususbesar-. [Accessed 30 September 2014]. |
| [9] | "Accuracy and Precision," [Online]. Available: http://en.wikipedia.org/wiki/Accuracy\_and\_precision. [Accessed 30 September 2014]. |
| [10] | "Using Matlab in Java," [Online]. Available: https://hive.asu.edu/minc/images/originalphotos/groupphotos/196/251/Using%20matlab%20in%20Java.pdf. [Accessed 20 Oktober 2014]. |
| [11] | "Matlab Control," [Online]. Available: https://code.google.com/p/matlabcontrol/. [Accessed 20 Oktober 2014]. |
| [12] | "Principal Component Analysis (#1)," 21 Januari 2012. [Online]. Available: http://tyangluhtu.wordpress.com/2012/01/21/principal-component-analysis/. [Accessed 5 November 2014]. |
| [13] | "Bioinformatics Research Group," [Online]. Available: http://www.upo.es/eps/bigs/datasets.html. [Accessed 29 September 2014]. |
| [14] | "Introduction to Genetic Algorithm," [Online]. Available: http://www.obitko.com/tutorials/genetic-algorithms/crossover-mutation.php. [Accessed 5 November 2014]. |
| [15] | "Stack Overflow," [Online]. Available: http://stackoverflow.com/questions/2784514/sort-arraylist-of-custom-objects-by-property. [Accessed 24 November 2014]. |
| [16] | "Evolutionary Algorithms 3 Selection," [Online]. Available: http://www.geatbx.com/docu/algindex-02.html#P363\_18910. [Accessed 24 November 2014]. |
| [17] | "Daniweb," 2013. [Online]. Available: https://www.daniweb.com/software-development/java/threads/447515/random-numbers-without-duplicates. [Accessed 24 November 2014]. |
| [18] | "Stikom Surabaya," [Online]. Available: http://sir.stikom.edu/10/5/BAB%20II.pdf. [Accessed 26 November 2014]. |
| [19] | M. S. Mohamad, S. Deris, S. M. Yatim and M. R. Othman, "FEATURE SELECTION METHOD USING GENETIC ALGORITHM FOR THE CLASSIFCATION OF SMALL AND HIGH DIMENSION DATA," *First International Symposium on Information and Communications Technologies,* 2004. |
| [20] | B. A. Minartiningtyas, "Informatika - Artikel Teknik Informatika dan Sistem Informasi," 2 Januari 2013. [Online]. Available: http://informatika.web.id/principal-component-analysis-pcaproyeksi-eigen.htm. [Accessed 5 Januari 2015]. |
| [21] | "Kecerdasan Buatan," [Online]. Available: http://file.upi.edu/Direktori/FPMIPA/JUR.\_PEND.\_FISIKA/196302071991031-WASLALUDDIN/Handout\_InsCer.pdf. [Accessed 27 12 2014]. |
| [22] | "Data Mining Research," 10 06 2007. [Online]. Available: http://www.dataminingblog.com/standardization-vs-normalization/. [Accessed 12 11 2014]. |

***[Halaman ini sengaja dikosongkan]***

# LAMPIRAN

*Tabel A.1. Rekap Hasil Keluaran Program Skenario Uji Coba 1 pada Percobaan 1*

|  |  |  |  |
| --- | --- | --- | --- |
| **Percobaan 1** | | | |
| **Generasi** | **Individu** | **Nilai K** | **Akurasi (%)** |
| 1 | 1011110100110010101001110000010111010000100010111010101010001 | 5 | 87.1 |
| 2 | 1011110100110010101001110000010111010000100010111010101010001 | 5 | 87.1 |
| 3 | 1011110100110010101001110000010111010000100010111010101010001 | 5 | 87.1 |
| 4 | 1011110100110010101001110000010111010000100010111010101010001 | 5 | 87.1 |
| 5 | 1011110100110010101001110000010111010000100010111010101010001 | 5 | 87.1 |
| 6 | 1011110100110010101001110000010111010000100010111010101010001 | 5 | 87.1 |
| 7 | 1011110100110010101001110000010111010000100010111010101010001 | 5 | 87.1 |
| 8 | 1011110100110010101001110000010111010000100010111010101010001 | 5 | 87.1 |
| 9 | 1011110100110010101001110000010111010000100010111010101010001 | 5 | 87.1 |
| 10 | 1011110100110010101001110000010111010000100010111010101010001 | 5 | 87.1 |
| ***Running Time* (sekon)** | | | **45.12** |

*Tabel A.2. Rekap Hasil Keluaran Program Skenario Uji Coba 1 pada Percobaan 2*

|  |  |  |  |
| --- | --- | --- | --- |
| **Percobaan 2** | | | |
| **Generasi** | **Individu** | **Nilai K** | **Akurasi (%)** |
| 1 | 1011110100111011111111000011100111101011111110000110010010001 | 3 | 88.71 |
| 2 | 1011110100111011111111000011100111101011111110000110010010001 | 3 | 88.71 |
| 3 | 1011110100111011111111000011100111101011111110000110010010001 | 3 | 88.71 |
| 4 | 1011110100111011111111000011100111101011111110000110010010001 | 3 | 88.71 |
| 5 | 1011110100111011111111000011100111101011111110000110010010001 | 3 | 88.71 |
| 6 | 1011110100111011111111000011100111101011111110000110010010001 | 3 | 88.71 |
| 7 | 1011110100111011111111000011100111101011111110000110010010001 | 3 | 88.71 |
| 8 | 1011110100111011111111000011100111101011111110000110010010001 | 3 | 88.71 |
| 9 | 1011110100111011111111000011100111101011111110000110010010001 | 3 | 88.71 |
| 10 | 1011110100111011111111000011100111101011111110000110010010001 | 3 | 88.71 |
| ***Running Time* (sekon)** | | | **40.064** |

***Tabel A.3.* *Rekap Hasil Keluaran Program Skenario Uji Coba 1 pada Percobaan 3***

|  |  |  |  |
| --- | --- | --- | --- |
| **Percobaan 3** | | | |
| **Generasi** | **Individu** | **Nilai K** | **Akurasi (%)** |
| 1 | 1111111110001100001100110101110111101001000101101110001110111 | 4 | 87.1 |
| 2 | 1111111110001100001100110101110111101001000101101110001110111 | 4 | 87.1 |
| 3 | 1111111110001100001100110101110111101001000101101110001110111 | 4 | 87.1 |
| 4 | 1111111110001100001100110101110111101001000101101110001110111 | 4 | 87.1 |
| 5 | 1111111110001100001100110101110111101001000101101110001110111 | 4 | 87.1 |
| 6 | 1111111110001100001100110101110111101001000101101110001110111 | 4 | 87.1 |
| 7 | 1111111110001100001100110101110111101001000101101110001110111 | 4 | 87.1 |
| 8 | 1111111110001100001100110101110111101001000101101110001110111 | 4 | 87.1 |
| 9 | 1111111110001100001100110101110111101001000101101110001110111 | 4 | 87.1 |
| 10 | 1111111110001100001100110101110111101001000101101110001110111 | 4 | 87.1 |
| ***Running Time* (sekon)** | | | **40.522** |

*Tabel A.4.* *Rekap Hasil Keluaran Program Skenario Uji Coba 1 pada Percobaan 4*

|  |  |  |  |
| --- | --- | --- | --- |
| **Percobaan 4** | | | |
| **Generasi** | **Individu** | **Nilai K** | **Akurasi (%)** |
| 1 | 0011011100000011100100110001001110000010111101110001011001011 | 5 | 88.71 |
| 2 | 0011011100000011100100110001001110000010111101110001011001011 | 5 | 88.71 |
| 3 | 0011110110011110101100110100010000101000000010110000111100110 | 3 | 91.93 |
| 4 | 0011110110011110101100110100010000101000000010110000111100110 | 3 | 91.93 |
| 5 | 0011110110011110101100110100010000101000000010110000111100110 | 3 | 91.93 |
| 6 | 0011110110011110101100110100010000101000000010110000111100110 | 3 | 91.93 |
| 7 | 0011110110011110101100110100010000101000000010110000111100110 | 3 | 91.93 |
| 8 | 0011110110011110101100110100010000101000000010110000111100110 | 3 | 91.93 |
| 9 | 0011110110011110101100110100010000101000000010110000111100110 | 3 | 91.93 |
| 10 | 0011110110011110101100110100010000101000000010110000111100110 | 3 | 91.93 |
| ***Running Time* (sekon)** | | | **42.321** |

*Tabel A.5.* *Rekap Hasil Keluaran Program Skenario Uji Coba 1 pada Percobaan 5*

|  |  |  |  |
| --- | --- | --- | --- |
| **Percobaan 5** | | | |
| **Generasi** | **Individu** | **Nilai K** | **Akurasi (%)** |
| 1 | 1110111001010111111100101100001101101101100001111110100100010 | 5 | 88.71 |
| 2 | 1111111001111011100001011101111101111110000110100101110111000 | 3 | 93.55 |
| 3 | 1111111001111011100001011101111101111110000110100101110111000 | 3 | 93.55 |
| 4 | 1111111001111011100001011101111101111110000110100101110111000 | 3 | 93.55 |
| 5 | 1111111001111011100001011101111101111110000110100101110111000 | 3 | 93.55 |
| 6 | 1111111001111011100001011101111101111110000110100101110111000 | 3 | 93.55 |
| 7 | 1111111001111011100001011101111101111110000110100101110111000 | 3 | 93.55 |
| 8 | 1111111001111011100001011101111101111110000110100101110111000 | 3 | 93.55 |
| 9 | 1111111001111011100001011101111101111110000110100101110111000 | 3 | 93.55 |
| 10 | 1111111001111011100001011101111101111110000110100101110111000 | 3 | 93.55 |
| ***Running Time* (sekon)** | | | **59.115** |

*Tabel A.6.* *Rekap Hasil Keluaran Program Skenario Uji Coba 1 pada Percobaan 6*

|  |  |  |  |
| --- | --- | --- | --- |
| **Running 6** | | | |
| **Generasi** | **Individu** | **Nilai K** | **Akurasi (%)** |
| 1 | 1111110100001011001000000001101101100001111001110110001100000 | 3 | 85.48 |
| 2 | 0111101110000100100011101010101101111011011010110111111101000 | 4 | 88.71 |
| 3 | 0111101110000100100011101010101101111011011010110111111101000 | 4 | 88.71 |
| 4 | 0111101110000100100011101010101101111011011010110111111101000 | 4 | 88.71 |
| 5 | 0111101110000100100011101010101101111011011010110111111101000 | 4 | 88.71 |
| 6 | 0111101110000100100011101010101101111011011010110111111101000 | 4 | 88.71 |
| 7 | 0111101110000100100011101010101101111011011010110111111101000 | 4 | 88.71 |
| 8 | 0111101110000100100011101010101101111011011010110111111101000 | 4 | 88.71 |
| 9 | 0111101110000100100011101010101101111011011010110111111101000 | 4 | 88.71 |
| 10 | 0111101110000100100011101010101101111011011010110111111101000 | 4 | 88.71 |
| ***Running Time* (sekon)** | | | **45.401** |

*Tabel A.7.* *Rekap Hasil Keluaran Program Skenario Uji Coba 1 pada Percobaan 7*

|  |  |  |  |
| --- | --- | --- | --- |
| **Running 7** | | | |
| **Generasi** | **Individu** | **Nilai K** | **Akurasi (%)** |
| 1 | 1001101111110011000111001001110111110000010111001000010011110 | 3 | 88.71 |
| 2 | 1001101111110011000111001001110111110000010111001000010011110 | 3 | 88.71 |
| 3 | 1001101111110011000111001001110111110000010111001000010011110 | 3 | 88.71 |
| 4 | 1001101111110011000111001001110111110000010111001000010011110 | 3 | 88.71 |
| 5 | 1001101111110011000111001001110111110000010111001000010011110 | 3 | 88.71 |
| 6 | 1001101111110011000111001001110111110000010111001000010011110 | 3 | 88.71 |
| 7 | 1001101111110011000111001001110111110000010111001000010011110 | 3 | 88.71 |
| 8 | 1001101111110011000111001001110111110000010111001000010011110 | 3 | 88.71 |
| 9 | 1001101111110011000111001001110111110000010111001000010011110 | 3 | 88.71 |
| 10 | 1001101111110011000111001001110111110000010111001000010011110 | 3 | 88.71 |
| ***Running Time* (sekon)** | | | **52.32** |

*Tabel A.8.* *Rekap Hasil Keluaran Program Skenario Uji Coba 1 pada Percobaan 8*

|  |  |  |  |
| --- | --- | --- | --- |
| **Running 8** | | | |
| **Generasi** | **Individu** | **Nilai K** | **Akurasi (%)** |
| 1 | 1111001001000110000010000011011101010111000001111000100101011 | 5 | 90.32 |
| 2 | 1111001001000110000010000011011101010111000001111000100101011 | 5 | 90.32 |
| 3 | 1111001001000110000010000011011101010011000001111000100101011 | 5 | 91.93 |
| 4 | 1111001001000110000010000011011101010011000001111000100101011 | 5 | 91.93 |
| 5 | 1111001001000110000010000011011101010011000001111000100101011 | 5 | 91.93 |
| 6 | 1111001001000110000010000011011101010011000001111000100101011 | 5 | 91.93 |
| 7 | 1111001001000110000010000011011101010011000001111000100101011 | 5 | 91.93 |
| 8 | 1111001001000110000010000011011101010011000001111000100101011 | 5 | 91.93 |
| 9 | 1111001001000110000010000011011101010011000001111000100101011 | 5 | 91.93 |
| 10 | 1111001001000110000010000011011101010011000001111000100101011 | 5 | 91.93 |
| ***Running Time* (sekon)** | | | **51.097** |

*Tabel A.9.* *Rekap Hasil Keluaran Program Skenario Uji Coba 1 pada Percobaan 9*

|  |  |  |  |
| --- | --- | --- | --- |
| **Running 9** | | | |
| **Generasi** | **Individu** | **Nilai K** | **Akurasi (%)** |
| 1 | 1011100011010101000111110111110010100010101001001011000000000 | 5 | 88.71 |
| 2 | 1011100011010101000111110111110010100010101001001011000000000 | 5 | 88.71 |
| 3 | 1011100011010101000111110111110010100010101001001011000000000 | 5 | 88.71 |
| 4 | 1011100011010101000111110111110010100010101001001011000000000 | 5 | 88.71 |
| 5 | 1011100011010101000111110111110010100010101001001011000000000 | 5 | 88.71 |
| 6 | 1011100011010101000111110111110010100010101001001011000000000 | 5 | 88.71 |
| 7 | 1011100011010101000111110111110010100010101001001011000000000 | 5 | 88.71 |
| 8 | 1011100011010101000111110111110010100010101001001011000000000 | 5 | 88.71 |
| 9 | 1011100011010101000111110111110010100010101001001011000000000 | 5 | 88.71 |
| 10 | 1011100011010101000111110111110010100010101001001011000000000 | 5 | 88.71 |
| ***Running Time* (sekon)** | | | **51.33** |

*Tabel A.10. Rekap Hasil Keluaran Program Skenario Uji Coba 1 pada Percobaan 10*

|  |  |  |  |
| --- | --- | --- | --- |
| **Running 10** | | | |
| **Generasi** | **Individu** | **Nilai K** | **Akurasi (%)** |
| 1 | 1011011101110000011100011000111100110110101011010101111011010 | 6 | 88.71 |
| 2 | 1011011101110000011100011000111100110110101011010101111011010 | 6 | 88.71 |
| 3 | 1011011101110000011100011000111100110110101011010101111011010 | 6 | 88.71 |
| 4 | 1011011101110000011100011000111100110110101011010101111011010 | 6 | 88.71 |
| 5 | 1011011101110000011100011000111100110110101011010101111011010 | 6 | 88.71 |
| 6 | 1011011101110000011100011000111100110110101011010101111011010 | 6 | 88.71 |
| 7 | 1011011101110000011100011000111100110110101011010101111011010 | 6 | 88.71 |
| 8 | 1011011101110000011100011000111100110110101011010101111011010 | 6 | 88.71 |
| 9 | 1011011101110000011100011000111100110110101011010101111011010 | 6 | 88.71 |
| 10 | 1011011101110000011100011000111100110110101011010101111011010 | 6 | 88.71 |
| ***Running Time* (sekon)** | | | **41.961** |

*Tabel A.11.* *Rekap Hasil Keluaran Program 10 Kali Percobaan pada Skenario Uji Coba 1*

|  |  |  |  |
| --- | --- | --- | --- |
| **Running** | **Individu** | **Nilai K** | **Akurasi (%)** |
| 1 | 1011110100110010101001110000010111010000100010111010101010001 | 5 | 87.1 |
| 2 | 1011110100111011111111000011100111101011111110000110010010001 | 3 | 88.71 |
| 3 | 1111111110001100001100110101110111101001000101101110001110111 | 4 | 87.1 |
| 4 | 0011110110011110101100110100010000101000000010110000111100110 | 3 | 91.93 |
| 5 | 1111111001111011100001011101111101111110000110100101110111000 | 3 | 93.55 |
| 6 | 0111101110000100100011101010101101111011011010110111111101000 | 4 | 88.71 |
| 7 | 1001101111110011000111001001110111110000010111001000010011110 | 3 | 88.71 |
| 8 | 1111001001000110000010000011011101010011000001111000100101011 | 5 | 91.93 |
| 9 | 1011100011010101000111110111110010100010101001001011000000000 | 5 | 88.71 |
| 10 | 1011011101110000011100011000111100110110101011010101111011010 | 6 | 88.71 |
| **Rata - Rata Akurasi (%)** | | | **89.52** |
| **Standar Deviasi Akurasi** | | | **2.182** |
| ***Running Time* (sekon)** | | | **46.92** |

*Tabel A.12. Rekap Hasil Keluaran Program 10 Kali Percobaan pada Skenario Uji Coba 2*

|  |  |  |  |
| --- | --- | --- | --- |
| **Percobaan** | **Nilai K** | **Akurasi (%)** | ***Running Time* (menit)** |
| 1 | 5 | 93.55 | 26.60 |
| 2 | 3 | 93.55 | 30.86 |
| 3 | 4 | 91.93 | 32.21 |
| 4 | 6 | 95.16 | 30.97 |
| 5 | 4 | 93.55 | 28.28 |
| 6 | 4 | 91.93 | 30.31 |
| 7 | 2 | 91.93 | 29.93 |
| 8 | 5 | 93.55 | 28.20 |
| 9 | 5 | 91.93 | 50.85 |
| 10 | 5 | 91.93 | 23.08 |
| **Rata-Rata** | | **92.901** | **31.13** |
| **Standar Deviasi** | | **1.130** | **7.412** |

*Tabel A.13. Rekap Hasil Keluaran Program 10 Kali Percobaan pada Skenario Uji Coba 3*

|  |  |  |
| --- | --- | --- |
| **Nilai K** | **Akurasi (%)** | **Running Time (sekon)** |
| 1 | 87.1 | 1.341 |
| 3 | 80.65 | 2.147 |
| 5 | 80.65 | 1.531 |
| 7 | 77.42 | 1.134 |
| 9 | 79.03 | 1.912 |
| 11 | 75.81 | 1.911 |
| 13 | 75.81 | 0.344 |
| 15 | 75.81 | 0.256 |
| 17 | 70.97 | 0.547 |
| 19 | 70.97 | 0.588 |
| **Rata-rata** | **77.422** | **1.17** |
| **Standar Deviasi** | **4.809** | **0.704** |

*Tabel A.14. Model Terbaik yang Dihasilkan*

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Nilai *k* | | | | | 3 | | |
| *Crossvalidation* 1 | | | | ***Crossvalidation* 4** | | | |
|  | Data ke- | | |  | Data ke- | | |
| Centroid Positif | 32 | 27 | 6 | Centroid Positif | 17 | 11 | 19 |
| Centroid Negatif | 6 | 10 | 12 | Centroid Negatif | 5 | 14 | 12 |
| *Crossvalidation* 2 | | | | ***Crossvalidation* 5** | | | |
|  | Data ke- | | |  | Data ke- | | |
| Centroid Positif | 24 | 15 | 5 | Centroid Positif | 12 | 2 | 27 |
| Centroid Negatif | 13 | 10 | 11 | Centroid Negatif | 1 | 5 | 9 |
| *Crossvalidation* 3 | | | | **Atribut yang Digunakan** | | | |
|  | Data ke- | | | 1, 2, 3, 4, 5, 6, 7, 10, 11, 12, 13, 15, 16, 17, 22, 24, 25, 26, 28, 29, 30, 31, 32, 33, 35, 36,37,38,39, 44,45,47 , 50, 52, 53, 54, 56,57,58 | | | |
| Centroid Positif | 7 | 16 | 2 |

BIODATA PENULIS

Dini Putri Mandasari, lahir di Surabaya, pada tanggal 16 November 1992. Penulis menempuh pendidikan mulai dari SD Al-Hikmah Surabaya (1999-2005), SMP Al-Hikmah Surabaya (2005-2008), SMA Al-Hikmah Surabaya (2008-2011) dan S1 Teknik Informatika ITS (2011-2015).

Selama masa kuliah, penulis aktif dalam organisasi Himpunan Mahasiswa Teknik Computer (HMTC). Diantaranya adalah menjadi staff departemen hubungan luar himpunan mahasiswa teknik computer ITS 2012-2013 dan sekretaris umum himpunan mahasiswa teknik computer ITS 2013-2014. Penulis juga aktif dalam kegiatan kepanitiaan Schematics. Diantaranya penulis pernah menjadi sekretaris umum II Schematics 2012 dan staff kesekretariatan Schematics 2013.

Selama kuliah di teknik informatika ITS, penulis mengambil bidang minat Komputasi Cerdas Visual (KCV). Penulis pernah menjadi asisten dosen mata kuliah manajemen basis data. Komunikasi dengan penulis dapat melalui email: **diniputrimandasari@gmail.com**.