



TESIS -KI092312

GRAMMATICAL EVOLUTION UNTUK EKSTRAKSI FITUR DENGAN PENGUKURAN MULTI FITNESS

Go Frendi Gunawan
5111201033

DOSEN PEMBIMBING
Prof. Dr. Ir. Joko Lianto Buliali, M.Sc.

PROGRAM MAGISTER
BIDANG KEAHLIAN KOMPUTASI CERDAS DAN VISUALISASI
JURUSAN TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2013



TESIS -KI092312

GRAMMATICAL EVOLUTION FOR FEATURE EXTRACTION WITH MULTI FITNESS EVALUATION

Go Frendi Gunawan
5111201033

SUPERVISOR
Prof. Dr. Ir. Joko Lianto Buliali, M.Sc.

MASTER PROGRAM
THE EXPERTISE FIELD OF INTELLIGENT COMPUTING AND
VISUALIZATION
DEPARTMENT OF INFORMATICS
FACULTY OF INFORMATION TECHNOLOGY
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2013

GRAMMATICAL EVOLUTION UNTUK EKSTRAKSI FITUR DENGAN PENGUKURAN MULTI FITNESS

Nama mahasiswa : Go Frendi Gunawan
NRP : 5111201033
Pembimbing : Prof. Dr. Ir. Joko Lianto Buliali, M.Sc.

ABSTRAK

Ekstraksi fitur merupakan salah satu topik yang berpengaruh dalam penyelesaian masalah klasifikasi. Sampai saat ini, tidak ada cara yang baku untuk menentukan fitur-fitur terbaik dari data. Dalam tesis ini, dikembangkan suatu pendekatan *grammatical evolution* dengan pengukuran *multi fitness* (disebut GE Tatami) guna memperoleh fitur-fitur terbaik dari data. Metode tersebut bertugas menciptakan $n-1$ buah fitur baru yang sanggup memisahkan data secara hirarkikal, di mana n adalah jumlah kelas dalam data.

Beberapa metode telah dicoba dalam penelitian ini, antara lain algoritma genetika, *grammatical evolution* dengan pengukuran *fitness* global, *grammatical evolution* dengan pengukuran *multi fitness*, *grammatical evolution* dengan pengukuran *fitness Tatami*, dan *grammatical evolution* yang dikembangkan oleh Gavrilis.

Hasil penelitian menunjukkan bahwa pada data-data sintesis dengan menggunakan *decision tree classifier*, metode Tatami menunjukkan hasil yang lebih baik dari keempat metode lainnya. Data sintesis tersebut dapat dipisahkan secara hirarkikal menggunakan fitur-fitur yang di *generate*. Namun metode Tatami menunjukkan hasil yang kurang baik jika fitur-fitur ideal gagal terbentuk. Metode ini juga gagal saat digunakan SVM sebagai *classifier*.

Kata kunci: ekstraksi fitur, *grammatical evolution*, klasifikasi, *multi-fitness*.



Halaman ini sengaja dikosongkan

GRAMMATICAL EVOLUTION FOR FEATURE EXTRACTION WITH MULTI FITNESS EVALUATION

By : Go Frendi Gunawan
Student Identity Number : 5111201023
Supervisor : Prof. Dr. Ir. Joko Lianto Buliali, M.Sc.

ABSTRACT

Feature Extraction is a significant topic in classification problem. Until now, there is no standard way to determine best features of data. In this thesis, grammatical evolution with multiple fitness evaluation approach (named as GE Tatami) has been developed to extract best features of data. The method generates $n-1$ features to separate data hierarchically, with n is number of classes.

Some methods have been evaluated in this research, including genetics algorithm, grammatical evolution with global fitness measurement, grammatical evolution with multi fitness measurement, grammatical evolution with Tatami fitness measurement, and Gavrilis's grammatical evolution.

It is shown in the experiment that Tatami method produces better results compared to the four other methods for synthesis data using decision tree classifier. The synthesis data is hierarchically separable. However Tatami method fails to boost SVM's accuracy. This method also fails when ideal features cannot be found.

Keywords: feature extraction, grammatical evolution, classification, multi-fitness.



Halaman ini sengaja dikosongkan

KATA PENGANTAR

Segala puji syukur bagi Tuhan Yang Maha Esa, Guru atas segala guru, Raja atas segala raja, dan Programmer atas segala programmer. Hanya karena bantuan dan perkenanan-Nya lah buku tesis ini dapat diselesaikan dengan baik

Penulis mengucapkan terimakasih yang sedalam-dalamnya bagi kedua orang tua penulis, Hadi Gunawan dan Hioe Linda atas dukungan moral yang diberikan selama pengerjaan tesis. Serta adik penulis, Paula Natalia yang telah membantu penulis untuk memenuhi beberapa syarat birokratif.

Penulis juga berterima kasih kepada Bapak Dr. Waskitho Wibisono, S.Kom., M.Eng. selaku Ketua Program Magister Teknik Informatika yang telah memberi dukungan dan arahan dalam menyelesaikan permasalahan akademik.

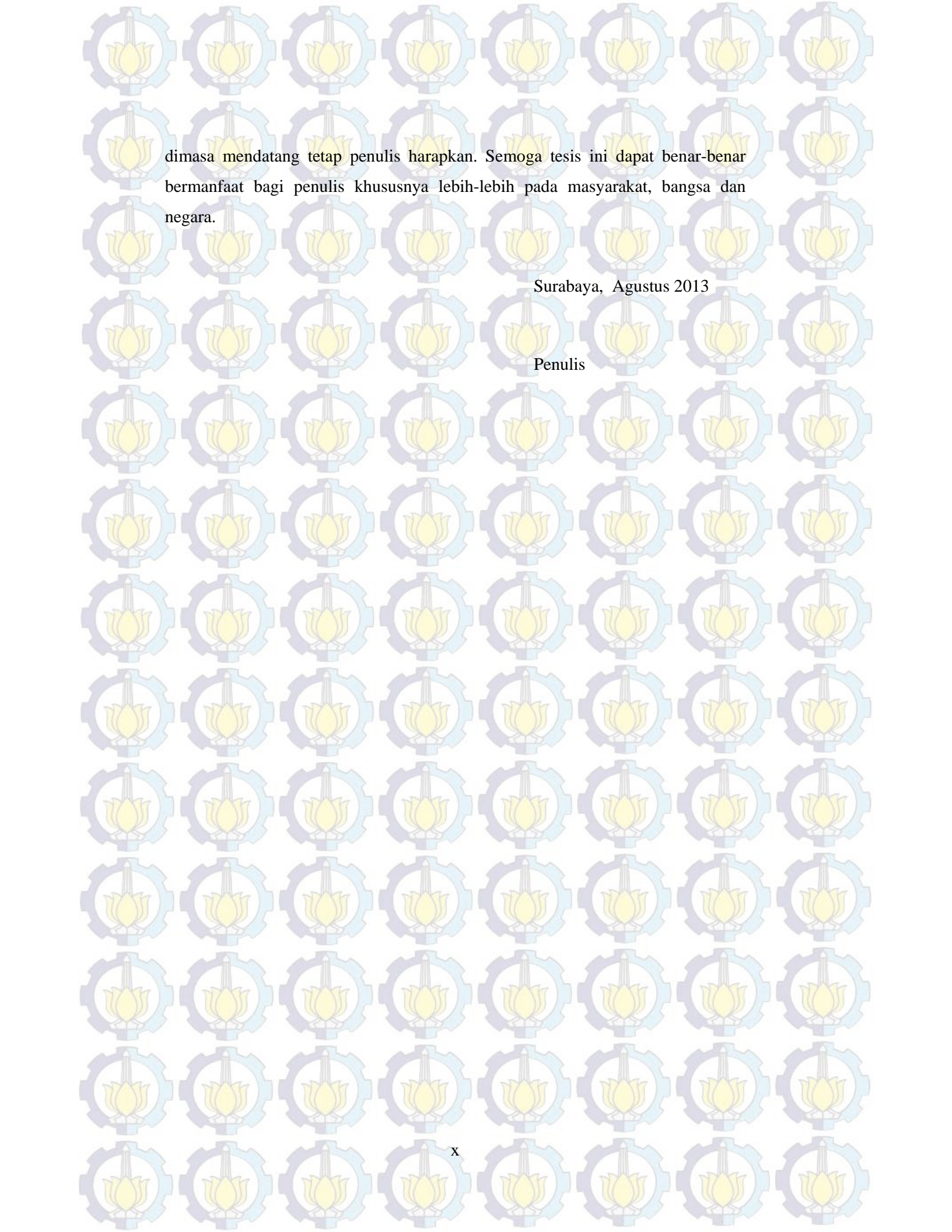
Terimakasih yang sedalam-dalamnya juga penulis tujukan untuk Dosen Pembimbing Bapak Prof. Dr. Ir. Joko Lianto Buliali, M.Sc. yang telah dengan sabar membimbing, memberikan ilmu, meluangkan waktu dan pikiran dalam proses pengerjaan tesis ini serta membuka wawasan penulis akan luasnya ilmu pengetahuan.

Bapak Dr. Ir. R.V. Hari Ginardi, M.Sc., M.Kom., Ibu Bilqis Amaliah, S.Kom, M.Kom., dan Ibu Isye Ariesianti, S.Kom, M.Phil. selaku dosen penguji yang telah banyak memberikan motivasi dan saran yang mendukung terselesaikannya tesis ini. Serta seluruh dosen S2 Teknik Informatika ITS yang telah memberikan wawasan serta ilmu pengetahuan baru bagi penulis selama menempuh masa studi pascasarjana.

Kepada teman seperjuangan dan seangkatan, khususnya Sonny Gosaria, Putra Prima, Naser Jawas dan teman-teman lain yang telah berbagi dan saling mendukung serta menyemangati dalam masa masa perkuliahan hingga masa penulisan thesis.

Kepada STIKI selaku pihak yang menugaskan dan membiayai studi S2 ini sampai selesai.

Penulis menyadari bahwa dalam laporan tesis ini masih banyak kekurangannya, karena itu masukan, saran demi perbaikan dan penerapan tesis ini



dimasa mendatang tetap penulis harapkan. Semoga tesis ini dapat benar-benar bermanfaat bagi penulis khususnya lebih-lebih pada masyarakat, bangsa dan negara.

Surabaya, Agustus 2013

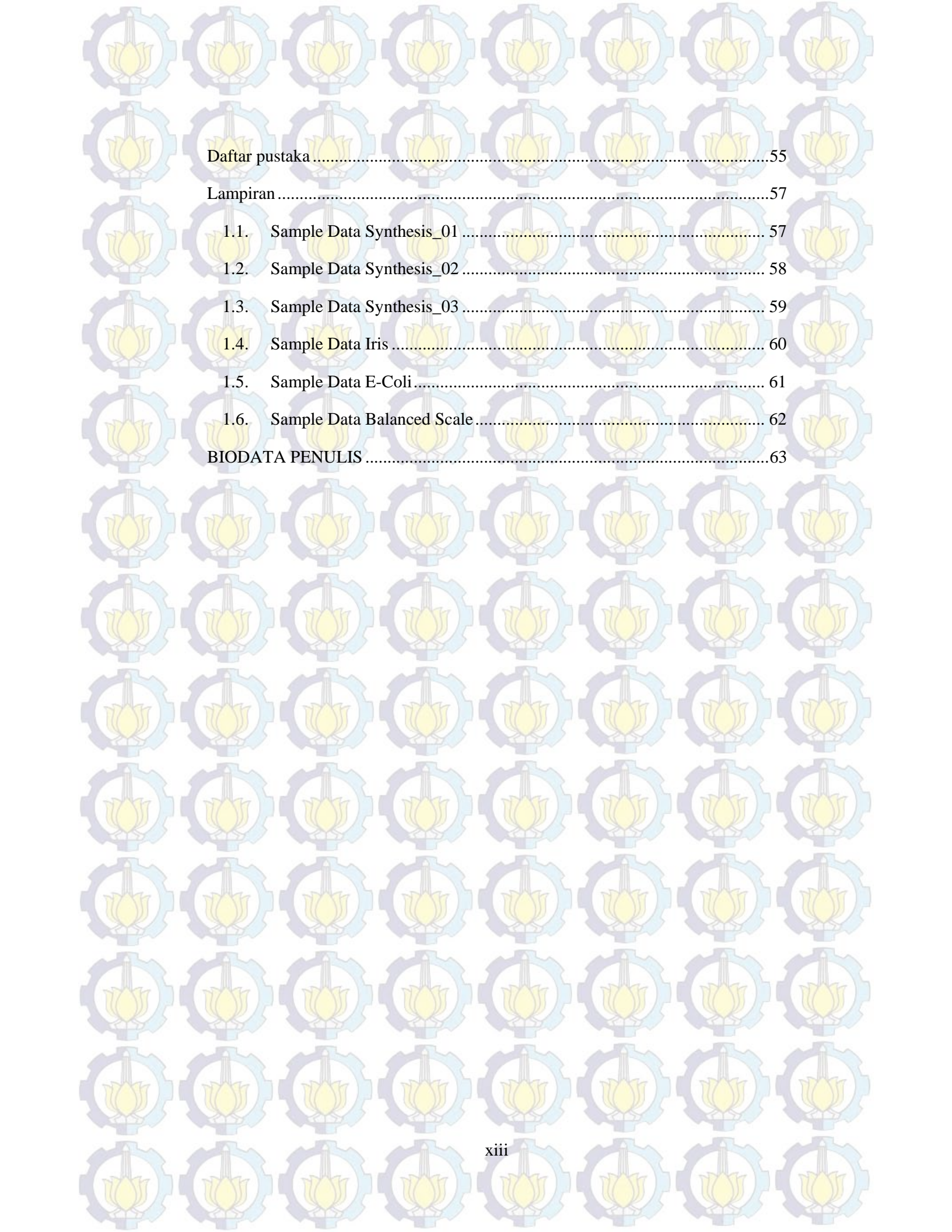
Penulis

DAFTAR ISI

Halaman

JUDUL	i
LEMBAR PENGESAHAN	iii
ABSTRAK	v
ABSTRACT	vii
KATA PENGANTAR	ix
DAFTAR ISI	xi
DAFTAR GAMBAR	xiv
DAFTAR TABEL	xvii
BAB 1 PENDAHULUAN	1
1.1 Latar Belakang	1
1.2 Perumusan Masalah	3
1.3 Tujuan dan Manfaat	3
1.4 Manfaat Penelitian	3
1.5 Kontribusi Penelitian	4
1.6 Batasan Penelitian	4
BAB 2 KAJIAN PUSTAKA DAN DASAR TEORI	5
2.1 Ekstraksi Fitur	5
2.2 Grammatical Evolution	9
2.2.1 Grammar	10
2.2.2 Transformasi Genotip Menjadi Fenotip	11
2.3 Decision Tree	13
BAB 3 METODA PENELITIAN	15
3.1 Langkah-Langkah Penelitian	15

3.2	Rancangan Sistem.....	16
3.3	Detail Sistem.....	18
3.3.1	Pendefinisian <i>Grammar</i>	18
3.3.2	Pembuatan Fitur.....	19
3.3.3	Normalisasi Proyeksi Data	20
3.3.4	Pemilihan Fitur Terbaik.....	21
3.3.4.1	Metode GA Select.....	22
3.3.4.2	Metode GE Global	23
3.3.4.3	Metode GE Multi	23
3.3.4.4	Metode GE Tatami	29
3.3.4.5	Metode GE Gavrilis	34
3.3.5	Pengukuran Performa Fitur	34
BAB 4 HASIL PENELITIAN DAN PEMBAHASAN		35
4.1	Pengujian	35
4.1.1	Dataset Sintesis 01.....	35
4.1.2	Dataset Sintesis 02.....	37
4.1.3	Dataset Sintesis 03.....	38
4.1.4	Dataset Iris.....	42
4.1.5	Dataset E-Coli	43
4.1.6	Dataset Balanced-Scale	44
4.1.7	Rata-Rata Hasil Pengujian.....	45
4.2	Analisis	46
4.2.1	Karakteristik GE Tatami.....	46
4.2.2	Kelemahan GE Tatami	51
BAB 5 KESIMPULAN DAN SARAN.....		53
5.1	Kesimpulan	53
5.2	Saran	54



Daftar pustaka	55
Lampiran	57
1.1. Sample Data Synthesis_01	57
1.2. Sample Data Synthesis_02	58
1.3. Sample Data Synthesis_03	59
1.4. Sample Data Iris	60
1.5. Sample Data E-Coli	61
1.6. Sample Data Balanced Scale	62
BIODATA PENULIS	63



Halaman ini sengaja dikosongkan

DAFTAR TABEL

	Halaman
Tabel 2.1 Contoh Data Numerik	6
Tabel 2.2 Contoh Grammar	11
Tabel 2.3 Segmen-Segmen Genotip	11
Tabel 3.1 Prediksi Classifier dan Fakta	22
Tabel 4.1 Hasil Pengujian pada Dataset Sintesis 01	36
Tabel 4.2 Hasil Pengujian pada Dataset Sintesis 02	38
Tabel 4.3 Hasil Pengujian pada Dataset Sintesis 03	41
Tabel 4.4 Hasil Pengujian pada Dataset Iris	42
Tabel 4.5 Hasil Pengujian pada Dataset E-Coli	43
Tabel 4.6 Hasil Pengujian pada Dataset Balanced-Scale	44
Tabel 4.7 Rata-Rata Hasil Pengujian pada Keenam Dataset	46

DAFTAR GAMBAR

	Halaman
Gambar 1.1 Pemisahan Cluster dengan Metode Tatami.....	2
Gambar 1.2 Pemisahan Cluster Secara Hirarkikal.....	3
Gambar 2.1 Ruang Fitur yang Terbentuk Berdasarkan Data pada Tabel 2.1	7
Gambar 2.2 Kurva yang Dibentuk oleh SVM untuk Memisahkan Ketiga Kelas ...	8
Gambar 2.3 Ruang Fitur yang Terbentuk dengan Memanfaatkan Fitur x^2+y^2	8
Gambar 2.4 Alur Umum Transformasi Genotip Menjadi Fenotip pada Grammatical Evolution dan Sistem Biologi. (Conor, 2006).....	10
Gambar 3.1 Skema Metode Penelitian.....	16
Gambar 3.2 Flowchart Sistem.....	17
Gambar 3.3 Grammar yang Digunakan.	18
Gambar 3.4 Fungsi Transformasi untuk Mengubah Genotip Menjadi Fitur.	20
Gambar 3.5 Fungsi Proyeksi dan Normalisasi Data	21
Gambar 3.6 Perubahan Nilai Fitness Maksimum dalam 100 Generasi	25
Gambar 3.7 Flowchart GE Multi (Bagian 1)	26
Gambar 3.8 Flowchart GE Multi (Bagian 2)	27
Gambar 3.9 Flowchart GE Multi (Bagian 3)	28
Gambar 3.10 Flowchart GE Multi (Bagian 4)	29
Gambar 3.11 Perubahan Nilai Fitness Maksimum dalam 100 Generasi pada Perulangan Kedua GE Tatami.....	32
Gambar 3.12 Flowchart GE Tatami.	33
Gambar 4.1 Proyeksi data terhadap fitur $(f1)/(f3)$	40

Gambar 4.2 Proyeksi data terhadap fitur $\sqrt{\sqrt{f_4} * (\sqrt{f_1 + f_1} / 2))} - (n_1 + n_1) / 2$ 41

Gambar 4.3 Proyeksi Data terhadap Fitur $(f_2) / (f_1)$ 48

Gambar 4.4 Proyeksi Data terhadap Fitur $(f_5) / (f_4)$ 48

Gambar 4.5 Proyeksi Data terhadap Fitur $(f_2) / (f_1)$ dan Fitur $(f_1) / (f_3)$ dengan Decision Tree Classifier 49

Gambar 4.6 Decision Tree yang Terbentuk Berdasarkan Penggunaan Fitur-Fitur yang di-generate oleh GE Tatami. 50

Gambar 4.7 Proyeksi Data terhadap Fitur $(f_2) / (f_1)$ dan Fitur $(f_1) / (f_3)$ dengan SVM Classifier. 51

BAB 1

PENDAHULUAN

1.1 Latar Belakang

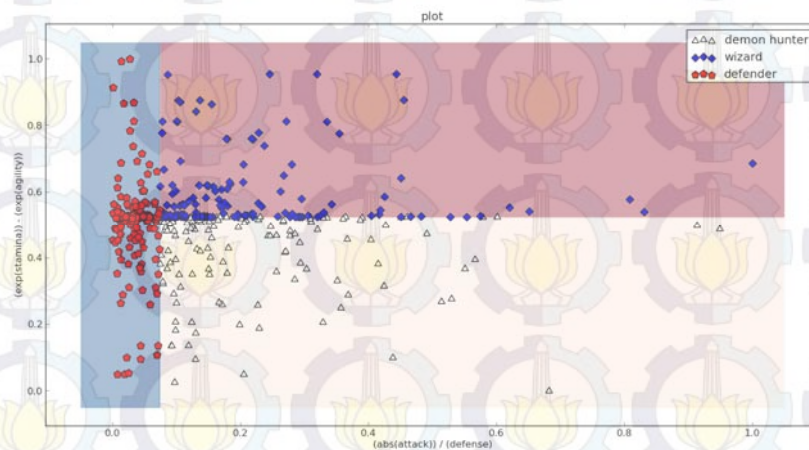
Ekstraksi fitur merupakan salah satu hal yang paling berpengaruh dalam pemecahan masalah klasifikasi. Pemilihan fitur yang tidak baik akan mengakibatkan kesulitan dalam memisahkan kelas-kelas data. Kegagalan pemisahan kelas-kelas data akan berdampak pada turunnya akurasi dalam proses klasifikasi.

Dalam penelitian sebelumnya (Gunawan, Gosaria, & Arifin, 2012), telah dicoba suatu pendekatan ekstraksi fitur dengan menggunakan *grammatical evolution*. Dalam penelitian tersebut, terdapat sebuah kelemahan dikarenakan hanya dilakukan 1 tolak ukur global untuk pengukuran *fitness value*. Hal ini mengakibatkan fitur-fitur yang sebenarnya cukup baik secara khusus, justru tersingkirkan karena nilai *fitness* globalnya rendah. Penelitian-penelitian lain seperti (Gavrilis, Tsoulous, Georgoulas, & Glavas, 2005) dan (Gavrilis, Tsoulous, & Dermatas, Selecting and Constructing Features Using Grammatical Evolution, 2008) juga menggunakan satu nilai *fitness* terhadap satu set fitur. (Li, Zhang, Tian, Mi, Liu, & Ruo, 2011) dan (Guo, Rivero, Dorado, Munteanu, & Pazos, 2011) juga melakukan hal yang hampir sama terhadap kasus yang berbeda.

Dalam penelitian ini diusulkan suatu cara baru dalam penilaian *fitness*. Penilaian *fitness* tersebut akan dilakukan dengan cara mengukur keterpisahan satu kelas terhadap kelas-kelas lain pada tiap dimensi. Metode tersebut, selanjutnya dinamakan *Tatami* karena kemiripannya dengan bentuk lantai tradisional Jepang. Dalam metode ini, untuk memisahkan n buah kelas, maka dibutuhkan maksimal $n-1$ buah fitur (dimensi).

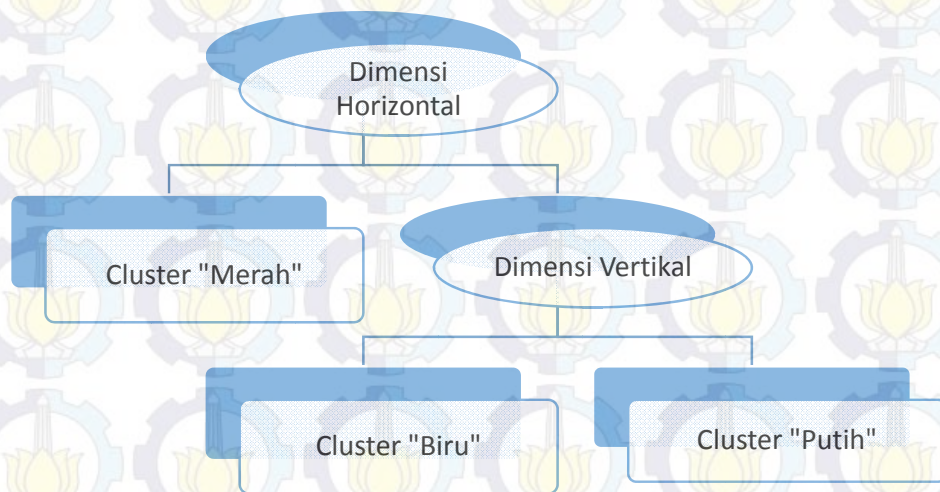
Sebagai contoh, pada Gambar 1.1 terdapat tiga buah cluster yang masing-masing direpresentasikan dengan warna merah, biru dan putih. Pemisahan ketiga cluster tersebut, dapat dilakukan dengan menggunakan 2 buah fitur (dimensi). Dimensi horizontal bertugas untuk memisahkan cluster merah dan kedua cluster lain. Pada dimensi horizontal ini, cluster biru dan putih tidak terpisahkan.

Sedangkan pada dimensi vertikal, cluster biru dan putih terpisahkan, walaupun kedua cluster tersebut tidak terpisah dari cluster merah. Dengan menggunakan kedua dimensi ini, maka akan terbentuk ruang fitur baru di mana cluster merah, biru dan putih terpisah secara linear.



Gambar 1.1 Pemisahan Cluster dengan Metode Tatami.

Adapun pada Gambar 1.1, tampak bahwa pemisahan ketiga cluster tersebut membentuk struktur hirarkikal, seperti yang ditunjukkan dalam gambar 1.2. Struktur hirarkikal tersebut mirip dengan struktur *decision tree*. Sehingga, diharapkan GE Tatami dapat menghasilkan sejumlah fitur terbaik, untuk membantu *decision tree classifier* memisahkan kelas-kelas yang ada secara optimal.



Gambar 1.2 Pemisahan Cluster Secara Hirarkikal

1.2 Perumusan Masalah

Adapun perumusan masalah berdasarkan latar belakang dari penelitian ini adalah:

- Bagaimana menentukan fitur-fitur optimal untuk masalah klasifikasi dengan menggunakan *decision treeclassifier*.
- Bagaimana menerapkan skenario pengukuran *fitness* untuk semua kelas.
- Bagaimana melakukan pengujian atas fitur-fitur yang sudah di-generate.

1.3 Tujuan dan Manfaat

Tujuan penelitian ini adalah menghasilkan dan menguji suatu metode baru (GE Tatami) yang berbasis grammatical evolution untuk mengekstrasi fitur pada data numerik, khususnya yang dimungkinkan untuk terpisah secara hirarkikal.

1.4 Manfaat Penelitian

Hasil penelitian dapat digunakan sebagai salah satu langkah *preprocessing* sebelum melakukan proses klasifikasi data numerik dengan menggunakan *decision tree*.

Dengan melakukan ekstraksi fitur sebagai bagian dalam tahap *preprocessing*, diharapkan proses klasifikasi data dengan fitur-fitur yang tidak

memiliki korelasi langsung terhadap keterpisahan kelas tetap dapat dilakukan dengan baik. Contoh penggunaan disajikan dalam bab empat.

1.5 Kontribusi Penelitian

Kontribusi penelitian ini adalah memberikan mekanisme baru untuk pengekstraksian fitur dengan menggunakan grammatical evolution, khususnya pada data numeric yang dimungkinkan untuk terpisah secara hirarkikal.

1.6 Batasan Penelitian

Batasan masalah pada penelitian ini adalah:

- a. Data yang diproses adalah data numerik. Jika ada data yang bersifat non-numerik, maka perlu dilakukan pengkodean ke dalam bentuk angka terlebih dahulu.
- b. Data yang diproses tidak memiliki *missing attribute*. Artinya, untuk setiap data, nilai semua fiturnya diketahui.
- c. *Grammar* yang digunakan hanya meliputi operator dan fungsi-fungsi matematika umum saja (+, -, *, /, exp, abs, sqr, dan sqrt).
- d. *Classifier* yang digunakan adalah *decision tree*.

BAB 2

KAJIAN PUSTAKA DAN DASAR TEORI

Pada bab ini akan dibahas beberapa teori dasar yang menunjang dalam pembuatan Tesis.

2.1 Ekstraksi Fitur

Ekstraksi fitur adalah suatu proses untuk mencari transformasi atau pemetaan dari fitur-fitur original ke ruang fitur baru yang dapat memperbesar keterpisahan antar kelas (Guo, Rivero, Dorado, Munteanu, & Pazos, 2011).

Banyak peneliti menyetujui bahwa ekstraksi fitur adalah proses terpenting dan tersulit pada masalah pengenalan pola dan klasifikasi. Pemilihan fitur yang paling tepat, mungkin merupakan tugas tersulit dalam pengenalan pola (Micheli-Tzanakou, 2000). Ekstraksi fitur yang ideal akan menghasilkan sebuah representasi yang sangat memudahkan pekerjaan *classifier* (Duda, Hart, & Stork, 2000). Dalam banyak kasus, ekstraksi fitur dilakukan oleh manusia, berdasarkan pengetahuan atau pengalaman, bahkan intuisi para peneliti (Guo, Rivero, Dorado, Munteanu, & Pazos, 2011).

Adapun fitur-fitur hasil ekstraksi bisa dikatakan baik, jika berhasil memisahkan data berdasarkan kelas yang diharapkan dengan tingkat kesalahan sekecil mungkin.

Untuk menjelaskan tujuan dari ekstraksi fitur, pada tabel 2.1 ditampilkan contoh data numerik. Data tersebut terdiri dari 2 fitur original, yakni x dan y . Masing-masing baris dalam tabel digolongkan dalam 3 buah kelas, yakni A, B dan C.

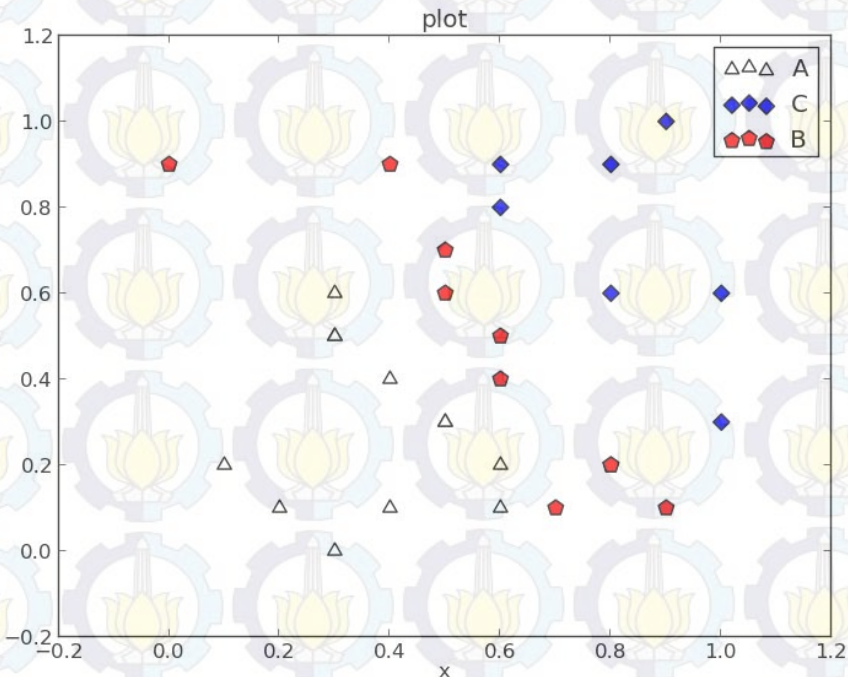
Jika data numerik pada tabel 2.1 direpresentasikan dalam bentuk grafis (ruang fitur) seperti yang disajikan pada gambar 2.1, dengan fitur x sebagai dimensi horizontal, dan y sebagai dimensi vertikal, maka akan tampak bahwa penggunaan dimensi x dan y dapat menciptakan ruang fitur yang sanggup memisahkan kelas A, B dan C.

Adapun demikian, penggunaan dimensi x dan y secara terpisah akan mengakibatkan data-data pada kelas A, B dan C saling overlap (menempati posisi

yang sama). Sebagai contoh, terdapat data dari kelas A, B dan C yang sama-sama memiliki nilai $x = 0,6$

Tabel 2.1 Contoh Data Numerik

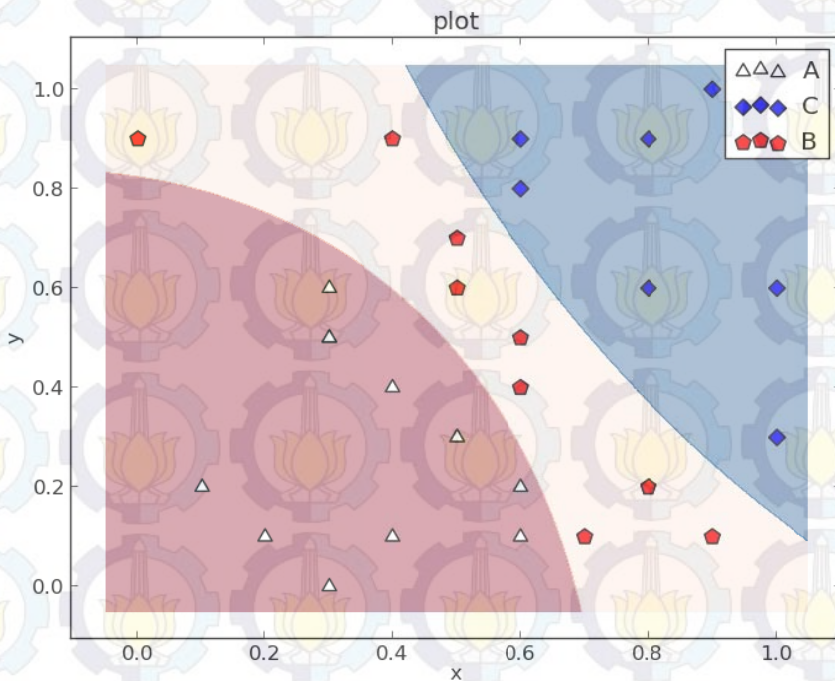
Fitur Asli		Kelas
x	y	
3	5	A
4	9	B
6	2	A
1	2	A
8	2	B
5	3	A
10	3	C
0	9	B
9	10	C
6	8	C
5	3	A
8	6	C
9	1	B
6	1	A
5	6	B
3	0	A
6	9	C
7	1	B
6	4	B
4	1	A
6	5	B
3	6	A
8	9	C
3	5	A
8	2	B
5	7	B
4	4	A
10	6	C
2	1	A



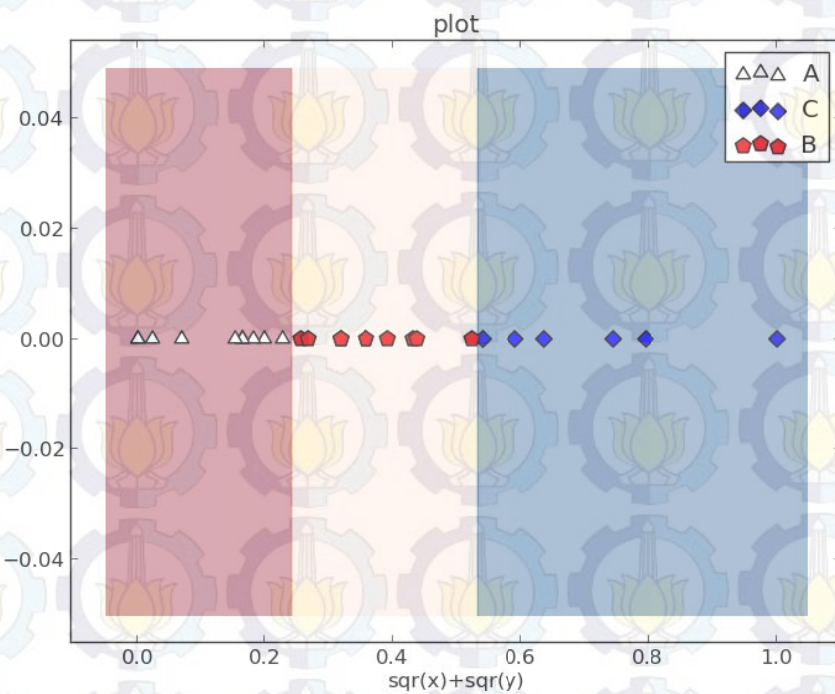
Gambar 2.1 Ruang Fitur yang Terbentuk Berdasarkan Data pada Tabel 2.1

Sekalipun ruang fitur pada gambar 2.1 telah memungkinkan data untuk dipisahkan berdasarkan kelas, namun garis pemisah antara ketiga fitur tersebut ternyata berbentuk garis non-linear yang secara matematis akan lebih kompleks dibandingkan garis linear. *Classifier* seperti SVM (Support Vector Machine) sebenarnya mampu menemukan kurva pemisah non-linear tersebut. Gambar 2.2 menunjukkan kurva pemisah yang dibentuk oleh SVM dengan kernel rbf.

Tujuan dari ekstraksi fitur adalah untuk menciptakan sesedikit mungkin fitur yang dapat memisahkan kelas-kelas secara cukup baik dan sederhana. Fitur yang dihasilkan dari proses ekstraksi dapat merupakan suatu fungsi matematika yang menggunakan subset dari fitur-fitur original sebagai komponennya, semisal x^2+y^2 . Gambar 2.3 menunjukkan bahwa penggunaan fitur x^2+y^2 ternyata telah cukup untuk memisahkan kelas A, B dan C secara sederhana (dengan menggunakan dua buah garis linear).



Gambar 2.2 Kurva yang Dibentuk oleh SVM untuk Memisahkan Ketiga Kelas



Gambar 2.3 Ruang Fitur yang Terbentuk dengan Memanfaatkan Fitur x^2+y^2

2.2 Grammatical Evolution

Grammatical Evolution adalah pengembangan dari *Genetics Programming* (yang merupakan pengembangan dari algoritma genetika), yang merupakan suatu algoritma untuk mendapatkan satu set program dalam bahasa tertentu. Dengan memanfaatkan *context-free grammar* yang ditulis dalam *Backus Naur form*, *grammatical evolution* mampu memisahkan representasi fenotip dan genotip dalam individu (Harper & Blair, 2006).

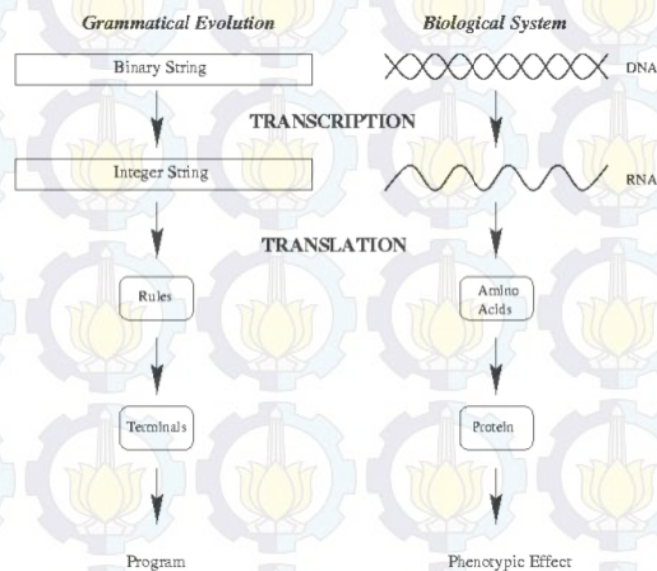
Pada *grammatical evolution*, sebuah individu memiliki dua buah representasi. Representasi yang pertama adalah representasi genotip, sedangkan representasi yang kedua adalah representasi fenotip.

Representasi genotip berupa sekumpulan angka sebagaimana layaknya pada algoritma genetika. Genotip dapat berupa angka biner maupun desimal. Representasi genotip pada *grammatical evolution* akan ditransformasikan menjadi representasi fenotip.

Representasi fenotip pada *grammatical evolution* dapat berupa fungsi matematika, kode program komputer, atau apapun, tergantung pada *grammar* yang digunakan.

Sama halnya seperti dalam algoritma genetika, pada *grammatical evolution* juga terdapat *fitness function* untuk mengukur kebaikan dari setiap individu. Untuk kasus ekstraksi fitur, umumnya tingkat akurasi *classifier* digunakan sebagai *fitness function*.

Alur umum *grammatical evolution* dan perbandingannya dengan transformasi genotip menjadi fenotip dalam biologi disajikan pada gambar 2.4.



Gambar 2.4 Alur Umum Transformasi Genotip Menjadi Fenotip pada Grammatical Evolution dan Sistem Biologi. (Conor, 2006)

2.2.1 Grammar

Untuk mentransformasikan representasi genotip menjadi representasi fenotip dibutuhkan sebuah *grammar*. *Grammar* di sini sebenarnya mirip dengan *grammar* dalam bahasa natural. Hanya saja, direpresentasikan dalam bentuk *backus naur form* (BNF). Dalam sebuah *grammar* terdapat beberapa bagian penting, antara lain:

- T : Terminal set. Merupakan node-node yang sudah tidak mungkin dievolusikan
- N : Non-terminal set. Merupakan node-node yang masih mungkin dievolusikan
- P : Production rules. Merupakan keseluruhan *grammar*
- S : Start symbol. Merupakan salah satu anggota N yang digunakan sebagai node

Semisal, didefinisikan production rules (P) seperti pada tabel 2.2, maka +, -, *, /, x, y, 1 merupakan anggota dari himpunan Terminal Set (T). Node-node yang menjadi anggota T, merupakan node-node yang sudah tidak mungkin dapat dievolusikan. Sementara itu <expr>, <op>, <var>, <num> digolongkan sebagai

Non-terminal Set (N). Node-node tersebut masih mungkin berevolusi menjadi node lain. Node $\langle \text{expr} \rangle$ berfungsi sebagai start symbol (S), artinya node $\langle \text{expr} \rangle$ merupakan node awal.

Tabel 2.2 Contoh *Grammar*

Notasi Node	Node	Aturan Produksi	Notasi Aturan
A	$\langle \text{expr} \rangle$	$\langle \text{expr} \rangle \langle \text{op} \rangle \langle \text{expr} \rangle$	A0
		$\langle \text{num} \rangle$	A1
		$\langle \text{var} \rangle$	A2
B	$\langle \text{op} \rangle$	+	B0
		-	B1
		*	B2
		/	B3
C	$\langle \text{var} \rangle$	x	C0
		y	C1
D	$\langle \text{num} \rangle$	1	D0

2.2.2 Transformasi Genotip Menjadi Fenotip

Transformasi genotip ke fenotip memanfaatkan *grammar* yang ada dan operasi modulo (siswa bagi) untuk memilih aturan transformasi.

Semisal terdapat representasi genotip 11.01.00.10.01, Genotip tersebut dapat dibagi dalam beberapa segmen sesuai kebutuhan. Dalam contoh transformasi ini, setiap segmen terdiri dari dua digit biner. Pembagian genotip dalam segmen-segmen disajikan dalam tabel 2.3

Tabel 2.3 Segmen-Segmen Genotip

Indeks Segmen	Segmen
1	11
2	01
3	00
4	10
5	01

Proses transformasi diawali dengan start symbol (dalam hal ini $\langle \text{expr} \rangle$). Selanjutnya diambil segmen dari genotip (dalam hal ini 11). Segmen tersebut

dapat pula dinyatakan dalam bilangan decimal (dalam hal ini 3). Node $\langle \text{expr} \rangle$ memiliki 3 kemungkinan perubahan ($A0 : \langle \text{expr} \rangle \langle \text{op} \rangle \langle \text{expr} \rangle$, $A1 : \langle \text{num} \rangle$, dan $A2 : \langle \text{var} \rangle$). Untuk menentukan aturan mana yang akan digunakan, maka dilakukan operasi modulo (sisanya), di mana segmen genotip terpilih akan dibagi dengan jumlah kemungkinan evolusi. Karena $3 \bmod 3 = 0$, maka dipilihlah aturan $A0$, yakni $\langle \text{expr} \rangle \langle \text{op} \rangle \langle \text{expr} \rangle$. Dari langkah ini, diperoleh, $\langle \text{expr} \rangle \langle \text{op} \rangle \langle \text{expr} \rangle$ sebagai calon fenotip yang baru.

Proses transformasi dilanjutkan dengan mengambil segmen kedua dalam genotip, yaitu 01. Segmen tersebut dapat dinyatakan dalam bentuk decimal 1. Selanjutnya, diambil node non-terminal pertama dari calon fenotip yang didapat dalam langkah sebelumnya ($\langle \text{expr} \rangle \langle \text{op} \rangle \langle \text{expr} \rangle$), yakni $\langle \text{expr} \rangle$. Node tersebut memiliki 3 kemungkinan perubahan ($A0 : \langle \text{expr} \rangle \langle \text{op} \rangle \langle \text{expr} \rangle$, $A1 : \langle \text{num} \rangle$, dan $A2 : \langle \text{var} \rangle$). Sama seperti pada langkah sebelumnya segment genotip terpilih di modulo kan dengan jumlah kemungkinan perubahan. Karena $1 \bmod 3 = 1$, maka dipilihlah aturan $A1$, yakni $\langle \text{var} \rangle$. Maka calon fenotip $\langle \text{expr} \rangle \langle \text{op} \rangle \langle \text{expr} \rangle$ berubah menjadi $\langle \text{var} \rangle \langle \text{op} \rangle \langle \text{expr} \rangle$.

Node non-terminal pertama dari calon fenotip yang baru ($\langle \text{var} \rangle \langle \text{op} \rangle \langle \text{expr} \rangle$) adalah $\langle \text{var} \rangle$, sedangkan segmen ketiga genotip adalah 00 yang bisa direpresentasikan sebagai 0 dalam basis desimal. $\langle \text{var} \rangle$ memiliki 2 kemungkinan perubahan ($C0 : x$ dan $C1 : y$). Karena $0 \bmod 2 = 0$, maka dipilih aturan $C0$, sehingga $\langle \text{var} \rangle$ berubah menjadi x . Maka kini calon fenotip berubah menjadi $x \langle \text{op} \rangle \langle \text{var} \rangle$.

Sekarang node non-terminal pertama dari calon fenotip yang baru ($x \langle \text{op} \rangle \langle \text{var} \rangle$) adalah $\langle \text{op} \rangle$. Segmen keempat genotip adalah 10, yang bisa direpresentasikan sebagai 2 dalam basis desimal. $\langle \text{op} \rangle$ memiliki 4 aturan perubahan ($B0 : +$, $B1 : -$, $B2 : *$, $B3 : /$). Karena $2 \bmod 4 = 2$, maka dipilih aturan $B1$, sehingga $\langle \text{op} \rangle$ berubah menjadi $-$. Dengan demikian calon fenotip bertransformasi menjadi $x - \langle \text{expr} \rangle$.

Proses ini dilanjutkan terus sampai seluruh node telah bertransformasi menjadi anggota terminal set (T). Seandainya, dalam proses ini, segmen genotip telah habis terpakai sebelum semua node berubah menjadi terminal node, maka akan kembali digunakan segmen pertama.

Proses transformasi secara ringkas disajikan dalam tabel 2.4. Di sini diperoleh bahwa representasi fenotip dari 11.01.00.10.01 adalah 1+Y

2.3 Decision Tree

Decision Trees (DTs) merupakan salah satu metode yang umum digunakan untuk masalah regresi dan klasifikasi. Metode ini akan menghasilkan sebuah model yang dapat digunakan untuk menebak nilai variabel target. Model yang dihasilkan *decision tree* sebenarnya merupakan aturan inferensi sederhana yang didapat dari fitur-fitur yang ada (Pedregosa, et al., 2011)

Decision tree memiliki kecenderungan overfit yang tinggi, selain itu korelasi tiap atribut terhadap keterpisahan kelas memegang peranan yang sangat penting. Karakteristik ini menyebabkan ekstraksi fitur menjadi salah satu hal yang berdampak pada akurasi *decision tree*.



Halaman ini sengaja dikosongkan

BAB 3

METODA PENELITIAN

3.1 Langkah-Langkah Penelitian

Pada penelitian ini, terdapat beberapa tahapan penyelesaian yang akan dilakukan, yang masing-masing tahapan menggunakan suatu metode tertentu. Adapun tahapan dan metode yang digunakan adalah sebagai berikut (gambar 3.1):

1. Studi literatur dan pencarian dataset.

Proses ini terdiri atas pencarian referensi-referensi pendukung yang sesuai, baik dari buku, jurnal, maupun artikel. Proses tersebut dilanjutkan dengan pencarian data-data numerik yang tersedia di internet sesuai dengan batasan permasalahan. Selain data-data umum, juga akan dibuat beberapa data sintesis yang dibuat dengan program *spreadsheet*.

2. Menyusun *grammar* dan rancang bangun sistem.

Proses ini terdiri atas perancangan formula *grammar* dan algoritma umum dalam proses ekstraksi fitur. Untuk proses ekstraksi fitur, akan digunakan lima macam metode yang akan dibahas pada subbab 3.3.4:

- Metode GA Select
- Metode GE Global
- Metode GE Multi
- Metode GE Tatami
- Metode GE Gavrilis

3. Menyusun rancangan pengujian system.

Dalam proses ini ditentukan skenario pengujian. Pengujian yang dimaksud dapat berupa perbandingan hasil klasifikasi dengan ekstraksi fitur dalam penelitian ini, ekstraksi fitur dalam penelitian sebelumnya, dan tanpa ekstraksi fitur. Akurasi klasifikasi menggunakan *decision-tree classifier* akan digunakan sebagai

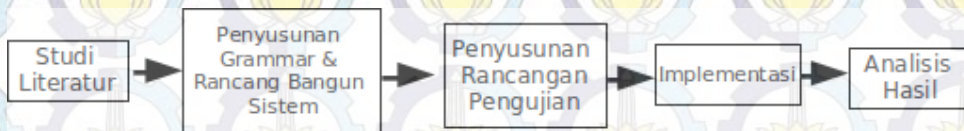
perbandingan. Khusus pada dataset synthesis 03, akan digunakan SVM dengan kernel linear sebagai pembandingan.

4. Mengimplementasikan system.

Sistem akan dibuat dalam bahasa pemrograman *Python* yang umum digunakan dalam kepentingan penelitian.

5. Menganalisis hasil yang diperoleh untuk menghasilkan kesimpulan.

Hasil ekstraksi fitur pada langkah nomor 4 akan diuji sesuai dengan rancangan pada langkah no 3. Selanjutnya akan disimpulkan apakah hasil penelitian lebih baik dari penelitian sebelumnya. Jika tidak lebih baik, maka akan dianalisis penyebab kegagalannya.



Gambar 3.1 Skema Metode Penelitian.

3.2 Rancangan Sistem

Secara umum, algoritma yang akan digunakan adalah sebagai berikut:

1. Input dataset.

Dataset yang ada ditransformasikan ke dalam bentuk yang sesuai, sehingga bisa diproses lebih lanjut oleh program

2. Menggunakan metode-metode pada subbab 3.1 langkah ke 2 guna mengekstraksi fitur.

Semua metode yang ada (GA Select, GE Global, GE Multi, GE Tatami, dan GE Gavrilis) akan digunakan untuk mengekstraksi fitur pada data yang telah diinputkan. Langkah ini dibagi dalam beberapa tahapan

a. *Generate* genotip.

Genotip individu pada semua metode berupa string biner yang di-*generate* secara acak. Pada semua metode selain

GA Select, genotip-genotip ini akan ditransformasikan menjadi fenotip.

b. Mengubah genotip menjadi fenotip (fitur-fitur baru).

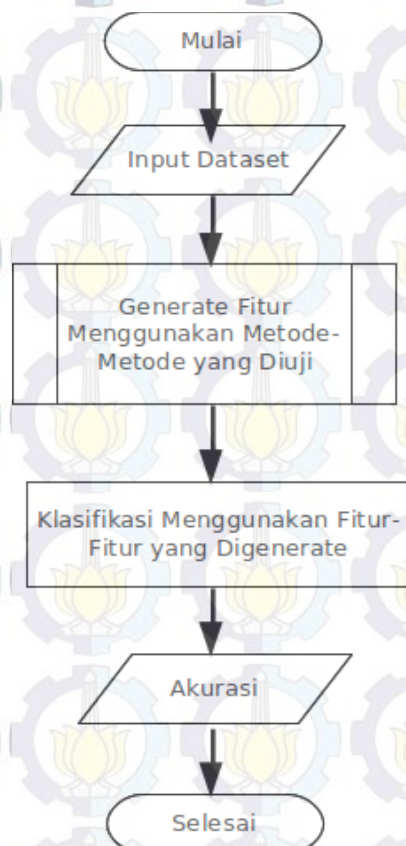
Sesuai dengan aturan *grammar* yang disediakan, setiap genotip akan diubah menjadi fenotip yang mewakili satu sebuah atau satu set fitur.

c. Menghitung nilai *fitness* dari setiap fenotip (fitur-fitur baru) terhadap setiap kelas.

Nilai *fitness* dari masing-masing fitur atau set fitur didapatkan dengan menggunakan akurasi decision tree pada ruang fitur yang tercipta.

d. Memilih fitur-fitur terbaik.

3. Membangun *feature space* berdasarkan fitur-fitur terbaik, dan melakukan proses klasifikasi.



Gambar 3.2 Flowchart Sistem.

3.3 Detail Sistem

Dalam subbab ini akan dijelaskan lebih lanjut mengenai detail rancangan sistem yang meliputi pendefinisian *grammar*, pembuatan fitur, normalisasi proyeksi data dan pemilihan fitur terbaik.

Proses ini dimulai dengan pendefinisian *grammar*. *Grammar* yang telah didefinisikan, kemudian akan digunakan untuk mentransformasi sejumlah genotip yang dihasilkan secara random menjadi sejumlah fenotip. Setiap fenotip akan dihitung nilai *fitness* nya. Selanjutnya semua fenotip akan diurutkan berdasarkan nilai *fitness*. Detail tahapan yang diperlukan untuk pembuatan fitur adalah sebagai berikut:

3.3.1 Pendefinisian Grammar

Dalam metode *grammatical evolution*, pendefinisian *grammar* merupakan bagian yang cukup penting. Pendefinisian *grammar* akan menentukan berbagai kemungkinan terciptanya fenotip. Setiap fenotip yang tercipta akan menjadi fitur-fitur baru yang siap dievaluasi berdasarkan nilai *fitness* nya.

Grammar yang digunakan dalam penelitian ini disajikan dalam gambar 3.3 (diimplementasikan dalam bahasa pemrograman *Python*):

```
1. self.variables = self.features
2. self.grammar = {
3.     '<expr>' : ['<var>', '<stmt>'],
4.     '<stmt>' : ['(<expr>) <op>'
5.     (<expr>)', '<func>(<expr>)', 'sqrt(sqr(<expr>+<expr>)/2)'],
6.     '<var>' : self.variables,
7.     '<op>' : ['+', '-', '*', '/'],
8.     '<func>' : ['exp', 'abs', 'sigmoid', 'sqr', 'sqrt', '-']
9. }
```

Gambar 3.3 Grammar yang Digunakan.

Grammar tersebut diharapkan dapat men-*generate* berbagai macam variasi matematika sederhana pada fitur-fitur original. Variabel *self.variables* berisi fitur-fitur data original. Pada *self.grammar* didefinisikan bahwa <expr> dapat berevolusi menjadi <var>, (<expr>) <op> (<expr>), atau <func>(<expr>). Sedangkan <var> dapat berevolusi menjadi fitur-fitur original. Demikian pula dengan <op> yang dapat berevolusi menjadi operator-operator matematika dan <func> yang dapat berevolusi menjadi salah satu dari fungsi-fungsi matematika terdefinisi. Proses evolusi sendiri akan bermula dari node <expr>

Perhitungan $\text{srt}(\text{sqr}(\langle \text{expr} \rangle + \langle \text{expr} \rangle)/2)$ digunakan untuk mengkombinasikan dua buah fitur menjadi sebuah fitur baru berupa garis dengan sudut 45 derajat terhadap kedua fitur sebelumnya.

3.3.2 Pembuatan Fitur

Proses pembuatan fitur tak lain adalah transformasi genotip (deretan angka acak yang telah di-generate) ke dalam bentuk fenotip menggunakan *grammatical evolution* dengan *grammar* terdefinisi. Proses ini telah dibahas dalam subbab 2.2.2. Pada implementasinya, proses ini didefinisikan dengan sebuah fungsi yang mengembalikan fitur baru dalam format data *string*.

Detail program ditunjukkan dalam gambar 3.4. Fungsi *_transform* menerima parameter *gene* yang bertipe data string dan berisi deretan angka biner. Kemudian dengan menggunakan parameter *gene* dan *grammar* yang telah didefinisikan sebelumnya, di-generate sebuah fitur (fenotip) baru. Fenotip tersebut berupa string yang berisi potongan kode program dalam bahasa Python.

```
1. def _transform(self, gene):
2.     # caching:
3.     if gene in self.genotype_dictionary:
4.         return self.genotype_dictionary[gene]
5.     # kedalaman maksimum = 20 (mencegah infinite loop)
6.     depth = 20
7.     gene_index = 0
8.     expr = self._start_node
9.     # dimulai dari level 0
10.    level = 0
11.    while level < depth:
12.        i=0
13.        new_expr = ''
14.        # parsing setiap karakter pada expr
15.        while i<len(expr):
16.            found = False
17.            for key in self._grammar:
18.                # ubah keyword berdasarkan aturan produksi
19.                if (expr[i:i+len(key)] == key):
20.                    found = True
21.                # jumlah kemungkinan transformasi
22.                possibility = len(self._grammar[key])
23.                # jumlah digit biner utk possibility
24.                digit_needed =
utils.bin_digit_needed(possibility)
25.                # jika akhir gen sudah tercapai
26.                if (gene_index+digit_needed)>len(gene):
27.                    # mulai dari depan lagi
28.                    gene_index = 0
29.                # bagian gen utk transformasi
30.                used_gene =
gene[gene_index:gene_index+digit_needed]
31.                gene_index = gene_index + digit_needed
32.                rule_index =
utils.bin_to_dec(used_gene)%possibility
```



```

33.         new_expr += self.grammar[key][rule_index]
34.         i += len(key)-1
35.         if not found:
36.             new_expr += expr[i:i+1]
37.             i += 1
38.         expr = new_expr
39.         level = level+1
40. # tambahkan ke cache
41.         self.genotype_dictionary[gene] = expr
42.         return expr

```

Gambar 3.4 Fungsi Transformasi untuk Mengubah Genotip Menjadi Fitur.

3.3.3 Normalisasi Proyeksi Data

Fitur yang telah di-generate pada subbab sebelumnya, selanjutnya digunakan untuk memproyeksikan data original. Proses ini didefinisikan dalam fungsi `get_projection`.

Untuk setiap record data yang ada, dilakukan proses evaluasi (didefinisikan pada `utils.execute`). Proses ini akan mengembalikan sebuah tuple yang berisi angka hasil evaluasi dan status error. Jika status error bernilai benar, maka ada kemungkinan bahwa hasil evaluasi tidak berupa angka (*Nan* atau *None*). Untuk meminimalisasi error hasil proyeksi, maka jika terjadi error, hasil evaluasi akan diasumsikan sebagai -1.

Selanjutnya, untuk semua data yang berhasil dievaluasi (tidak memunculkan error) akan dilakukan proses normalisasi. Proses normalisasi ini bertujuan untuk mengubah nilai minimum menjadi 0 dan nilai maksimum menjadi 1. Proses normalisasi ini bertujuan untuk memastikan bahwa setiap fitur memiliki *range* yang sama atau hampir sama.

Hasil proyeksi data direpresentasikan dalam bentuk *dictionary* dengan kelas sebagai *key*, dan list hasil proyeksi kelas tersebut sebagai *value*.

```

1. def get_projection(new_feature, old_features, all_data,
2.     used_data = None, used_target = None):
3.     used_projection, all_result, all_error = [], [], []
4.     # get all result
5.     for data in all_data:
6.         result, error = utils.execute(new_feature, data,
7.             old_features)
8.         all_error.append(error)
9.         if error:
10.            all_result.append(None)
11.        else:
12.            all_result.append(result)
13.    all_is_none = True
14.    for i in all_result:
15.        if i is not None:
16.            all_is_none = False

```



```

15.         break
16.     if all_is_none:
17.         min_result = 0
18.         max_result = 1
19.     else:
20.         min_result = min(x for x in all_result if x is not None)
21.         max_result = max(x for x in all_result if x is not None)
22.     if max_result-min_result>0:
23.         result_range = max_result-min_result
24.     else:
25.         result_range = LIMIT_ZERO
26.     if used_data is None: # include all data
27.         for i in xrange(len(all_result)):
28.             if all_error[i]:
29.                 all_result[i] = -1
30.             else:
31.                 all_result[i] = (all_result[i]-min_result)
32.         /result_range
33.         used_projection = all_result
34.     else:
35.         used_result = []
36.         for i in xrange(len(used_data)):
37.             result, error = utils.execute(new_feature,
38.             used_data[i], old_features)
39.             if error:
40.                 used_result.append(-1)
41.             else:
42.                 used_result.append((result-min_result)
43.                 /result_range)
44.         used_projection = used_result
45.     # pastikan isi projection hanya berupa angka (int atau float)
46.     for i in xrange(len(used_projection)):
47.         value = used_projection[i]
48.         if (not isinstance(value,float)) and
49.         (not isinstance(value,int)):
50.             value = -1
51.             if math.isnan(value):
52.                 value = -1
53.             used_projection[i] = round(value,2)
54.     if used_target is None:
55.         return used_projection
56.     group_projection = {}
57.     for i in xrange(len(used_projection)):
58.         group = used_target[i]
59.         if not group in group_projection:
60.             group_projection[group]=[]
61.         group_projection[group].append(used_projection[i])
62.     return group_projection

```

Gambar 3.5 Fungsi Proyeksi dan Normalisasi Data

3.3.4 Pemilihan Fitur Terbaik

Pemilihan fitur terbaik diperoleh dengan memanfaatkan lima buah metode. GE Multi dan GE Tatami merupakan metode yang diusulkan, sedangkan metode GA Select, GE Global dan GE Gavrilis digunakan sebagai pembanding.

Untuk pengukuran *fitness* individu, digunakan formula $(\text{true_positive}/(\text{true_positive}+\text{false_negative})+\text{true_negative}/(\text{true_negative}+\text{false_positive}))$

positive)) – 1. *True positive* adalah jumlah data yang oleh *classifier* diprediksi berada di dalam kelas tertentu dan ternyata memang benar berada dalam kelas tersebut. *True negative* adalah jumlah data yang oleh *classifier* diprediksi tidak berada di dalam kelas tertentu dan ternyata memang benar tidak berada dalam kelas tersebut. *False positive* adalah jumlah data yang oleh *classifier* diprediksi berada di dalam kelas tertentu namun ternyata tidak berada dalam kelas tersebut. *False negative* adalah jumlah data yang oleh *classifier* diprediksi tidak berada di dalam kelas tertentu namun ternyata berada dalam kelas tersebut. Dalam tabel 3.1 ditunjukkan hubungan antara *true positive*, *true negative*, *false positive*, *false negative* dan prediksi *classifier*.

Tabel 3.1 Prediksi *Classifier* dan Fakta

Prediksi <i>Classifier</i> VS Fakta	Berada dalam Kelas Tertentu	Tidak Berada dalam Kelas Tertentu
Diprediksi Berada dalam Kelas Tertentu	<i>True Positive</i>	<i>False Positive</i>
Diprediksi Tidak Berada dalam Kelas Tertentu	<i>False Negative</i>	<i>True Negative</i>

Penjelasan rinci mengenai masing-masing metode disajikan dalam subbab berikut:

3.3.4.1 Metode GA Select

Dalam metode GA Select, akan dipilih subset dari fitur original yang paling mampu memisahkan kelas dalam data secara optimum. Penilaian *fitness* dilakukan dengan memanfaatkan akurasi separator. Dalam implementasinya, untuk skenario ini digunakan algoritma genetika biasa.

Setiap individu dalam GA Select terdiri dari binary string. Jika karakter pertama pada binary string adalah 1, maka fitur asli pertama digunakan, sebaliknya jika karakter pertama pada binary string adalah 0, maka fitur pertama tidak digunakan. Demikian untuk karakter-karakter selanjutnya.

Semisal sebuah individu terdiri dari binary string 01110, dan terdapat 5 fitur f1-f5, maka:

- f1 tidak digunakan, karena karakter pertama adalah 0

- b. f2 digunakan, karena karakter kedua adalah 1
- c. f3 digunakan, karena karakter ketiga adalah 1
- d. f4 digunakan, karena karakter keempat adalah 1
- e. f5 tidak digunakan, karena karakter kelima adalah 0

Banyaknya fitur yang dapat di-*generate* dengan metode ini berkisar antara nol sampai dengan jumlah fitur original.

3.3.4.2 Metode GE Global

Dalam metode GE global, akan dipilih sebuah fitur yang mampu memisahkan semua kelas secara cukup baik. Penilaian *fitness* dilakukan dengan cara mengukur akurasi *classifier* terhadap data dengan menggunakan fenotip yang di-*generate* oleh grammatical evolution (proses grammatical evolution dijelaskan pada subbab 2.2). Metode GE Global merupakan implementasi dari penelitian sebelumnya (Gunawan, Gosaria, & Arifin, 2012)

Metode ini diharapkan menghasilkan 1 fitur terbaik yang dapat memisahkan semua kelas dalam data.

3.3.4.3 Metode GE Multi

Metode ini merupakan pengembangan dari GE Global. Dalam GE Multi, digunakan pengukuran *multi fitness* untuk memisahkan masing-masing kelas dengan keseluruhan kelas lain. Setiap individu dalam metode ini akan memiliki n buah nilai *fitness*, di mana n adalah jumlah kelas yang ada.

Banyaknya fitur yang bisa di-*generate* dalam metode ini adalah sebanyak jumlah kelas yang ada.

Semisal dalam sebuah dataset terdapat n buah kelas $\{C1, C2, C3, \dots Cn\}$, maka dalam GE Multi, akan terdapat n *fitness value* $\{f1, f2, f3, \dots fn\}$. Masing-masing *fitness value* menunjukkan keterpisahan sebuah kelas terhadap semua kelas lain. Dalam proses penentuan *fitness value*, kelas-kelas yang tidak selain kelas target akan disatukan kedalam satu kelas. Kemudian dilakukan proses klasifikasi dengan *classifier decision tree*. Proses ini dilakukan sebanyak n kali, sehingga diperoleh n *fitness value* untuk masing-masing individu.

Untuk menghitung $f1$, maka kelas $\{C2, C3, \dots Cn\}$ digabungkan menjadi $C \sim 1$. *Classifier* akan memisahkan $C1$ dan $C \sim 1$. Selanjutnya, akurasi *classifier* akan

dijadikan nilai *fitness* f1. Untuk menghitung f2, maka kelas {C1, C3,... Cn} digabungkan menjadi C~2. *Classifier* akan memisahkan C2 dan C~2. Selanjutnya, akurasi *classifier* akan dijadikan nilai *fitness* f2. Demikian seterusnya sampai fn.

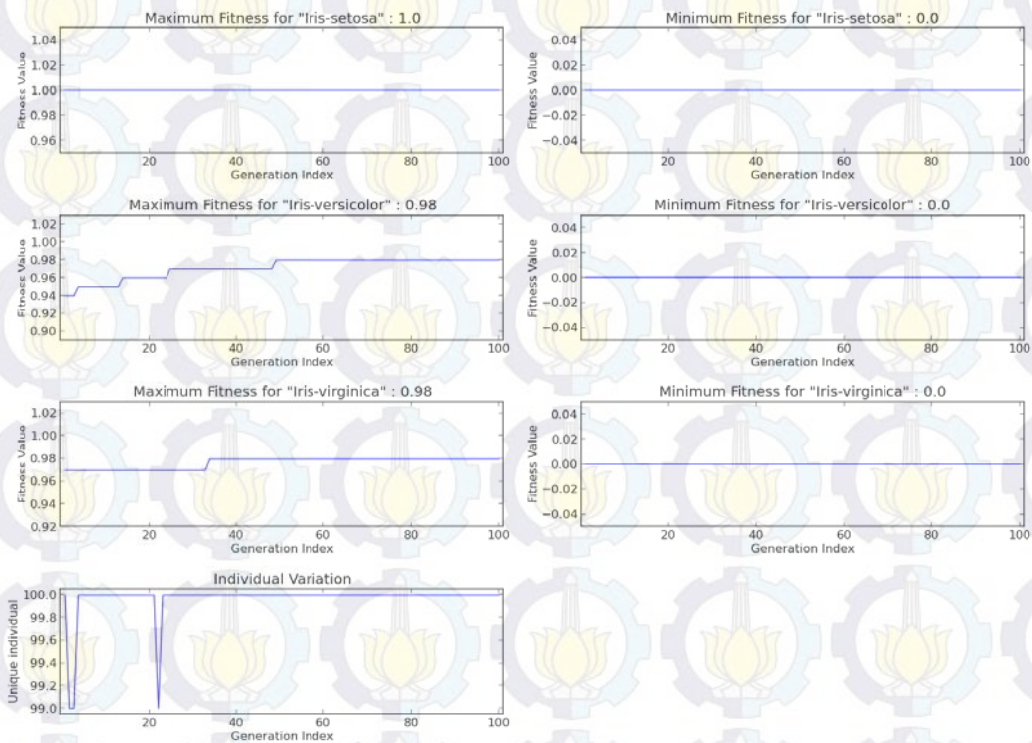
Di akhir proses GE Multi, dipilih satu individu yang memiliki f1 terbaik, 1 individu yang memiliki f2 terbaik, dan seterusnya, sehingga ditemukan n individu terbaik. Fenotip dari individu-individu terbaik ini selanjutnya digunakan sebagai fitur baru untuk proses klasifikasi.

Sebagai contoh, untuk dataset Iris yang terdiri dari 4 fitur (petal length, sepal length, petal width, dan sepal width) serta 3 kelas (iris-setosa, iris-virginica dan iris-versicolor), GE Multi menghasilkan 3 buah fitur:

- sepal_length
- $(\exp(\text{sepal_width})) * ((\text{sepal_length}) / (\text{petal_width}))$
- $\sqrt{\text{sqr}(\text{abs}(\sqrt{\text{sqr}(((\text{sepal_width}) + ((\text{sepal_width}) - (\sqrt{\text{sqr}((\text{petal_length}) - (\text{sepal_length}) + \text{sepal_width})/2)))) - (\text{petal_width}) + \text{petal_width}) / 2)) + \text{sepal_length}) / 2)}$

Fitur pertama memiliki *fitness* value tertinggi (bernilai 1,0) untuk memisahkan iris-setosa dengan kedua kelas lain. Fitur kedua memiliki *fitnessvalue* tertinggi (bernilai 0,98) untuk memisahkan iris-versicolor dengan kedua kelas lain. Fitur ketiga memiliki *fitness value* tertinggi untuk memisahkan iris-virginica (bernilai 0,98) dengan ketiga kelas lain.

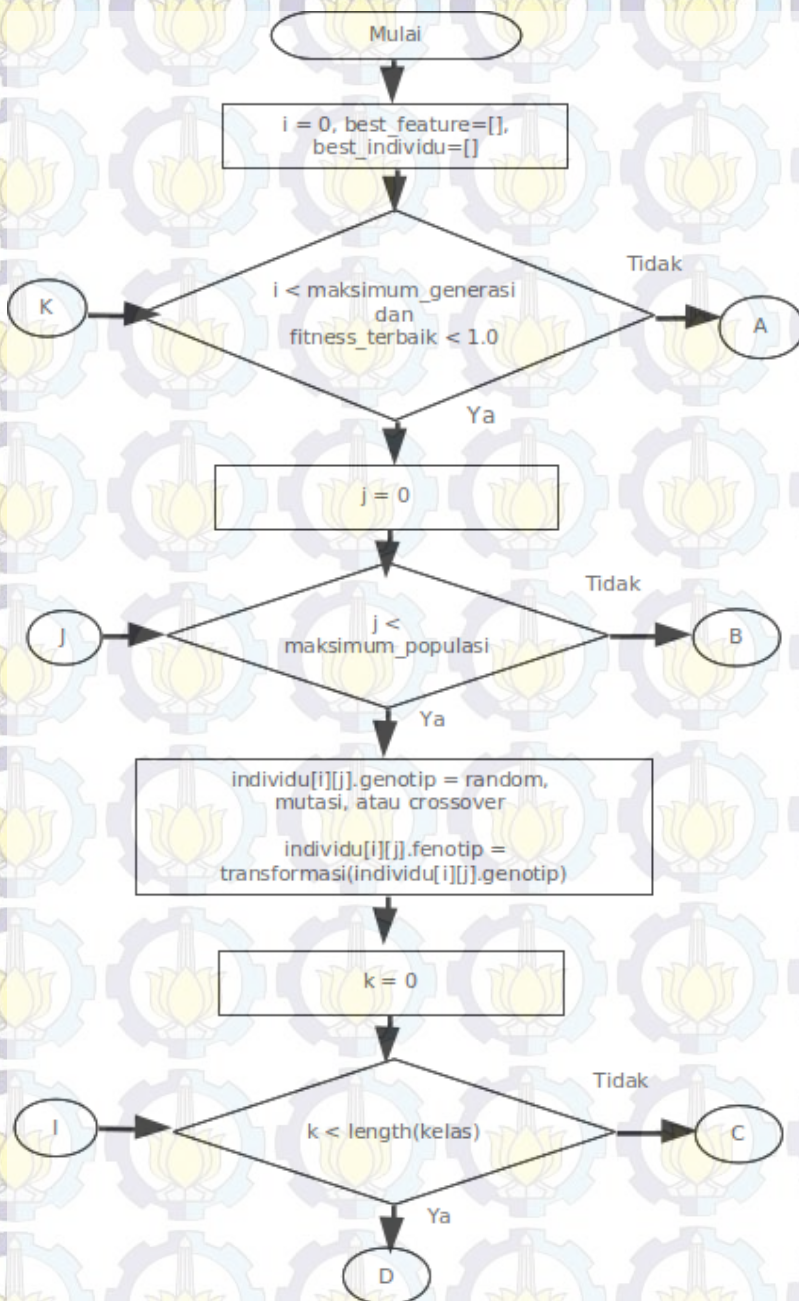
Data yang ada kemudian ditransformasi dalam menggunakan fitur-fitur baru yang telah di-generate GE Multi. Data hasil transformasi tadi kemudian digunakan sebagai input bagi *decision tree classifier*. Penggunaan ketiga fitur ini mengakibatkan akurasi decision tree meningkat menjadi 98,67%, dibandingkan dengan penggunaan fitur asli (sepal length, sepal width, petal length, dan petal width) yang hanya memberikan akurasi sebesar 96%.



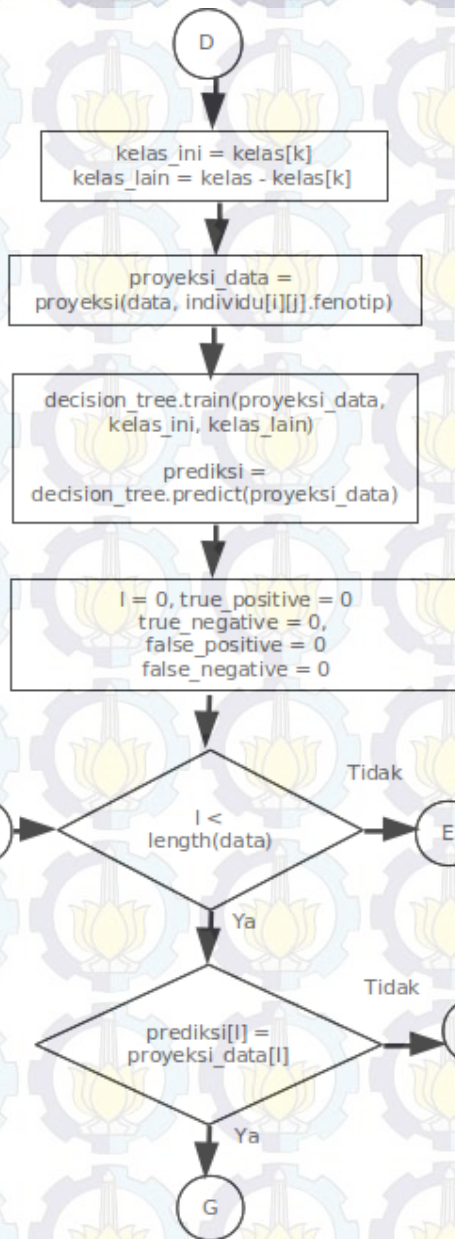
Gambar 3.6 Perubahan Nilai *Fitness* Maksimum dalam 100 Generasi

Gambar 3.6 menunjukkan perubahan nilai *fitness* terbaik pada tiap generasi. Panel-panel di bagian kiri menunjukkan nilai *fitness* maksimum yang didapat dari setiap generasi. Pada gambar tersebut tampak bahwa iris-setosa dapat terpisah secara mutlak dari kedua kelas lainnya, sedangkan iris-virginica dan iris-versicolor tidak dapat terpisah secara mutlak dari kelas lainnya.

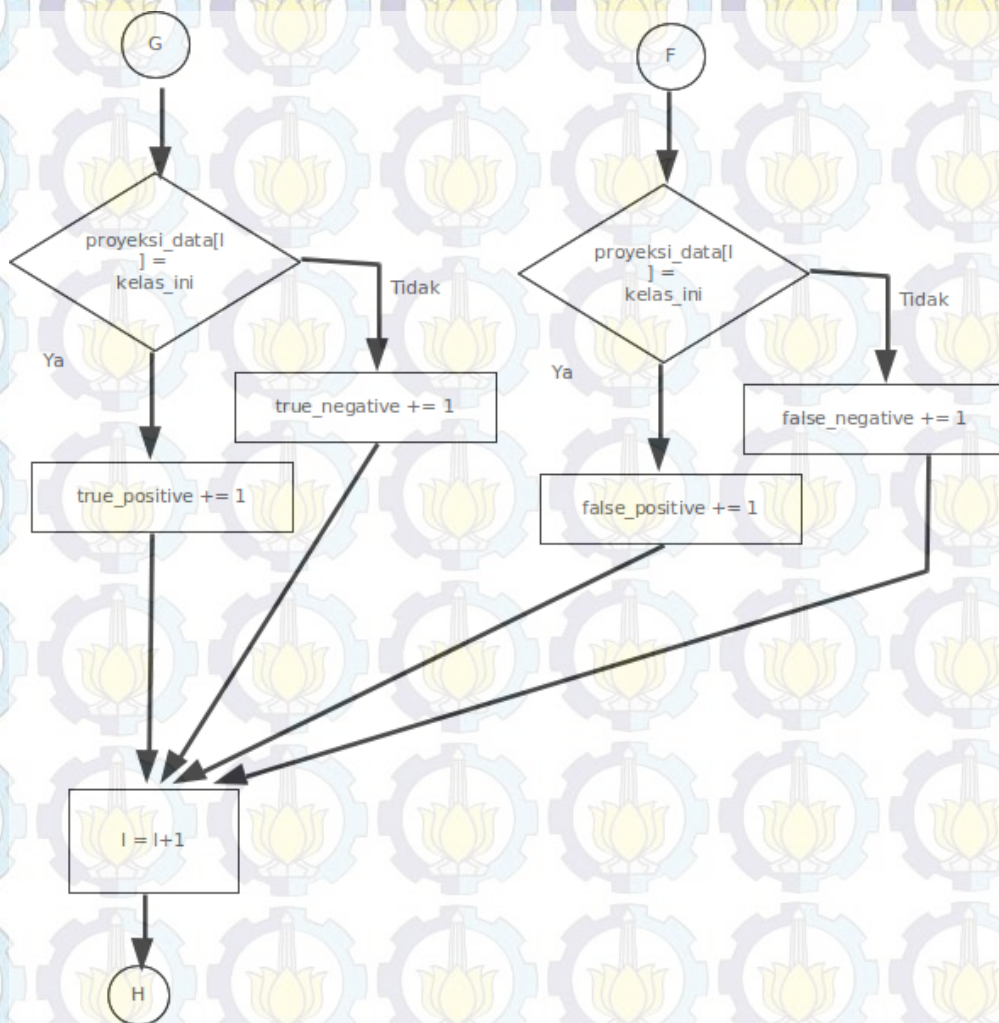
Flowchart metode GE Multi disajikan pada gambar 3.7 sampai gambar 3.10.



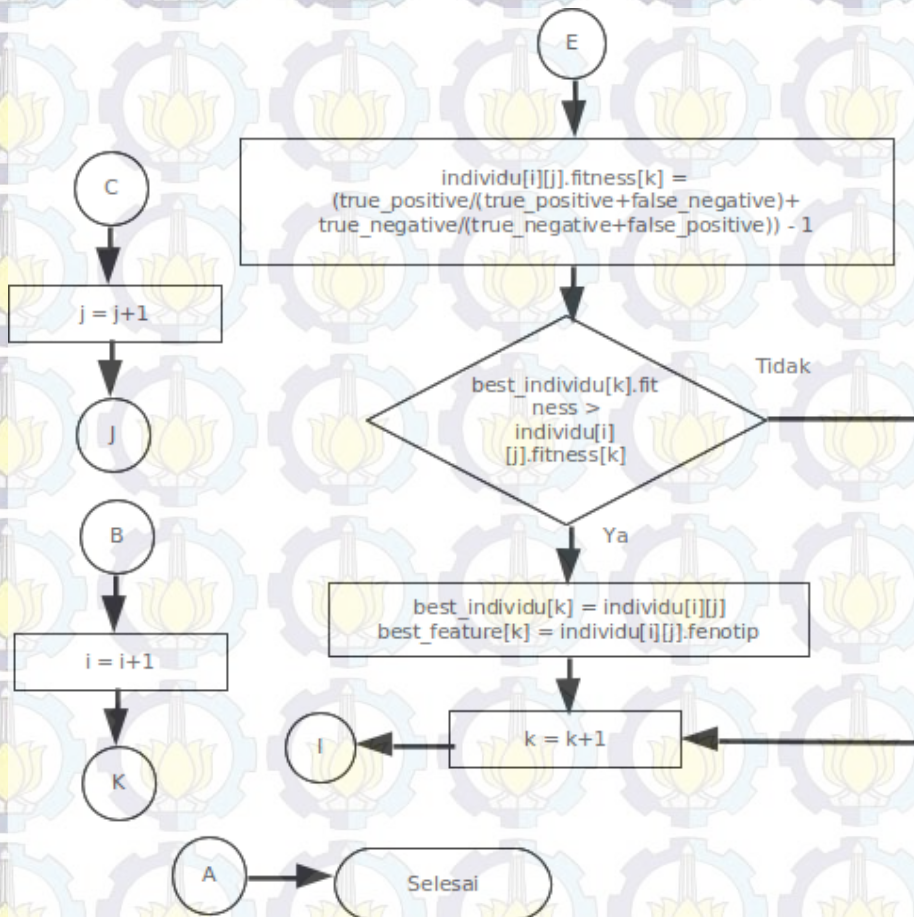
Gambar 3.7 Flowchart GE Multi (Bagian 1)



Gambar 3.8 Flowchart GE Multi (Bagian 2)



Gambar 3.9 Flowchart GE Multi (Bagian 3)



Gambar 3.10 Flowchart GE Multi (Bagian 4)

3.3.4.4 Metode GE Tatami

Metode GE Tatami merupakan pengembangan dari GE Multi. Pengembangan tersebut berasal dari hipotesa bahwa jika sebuah kelas telah terpisah dari semua kelas lainnya, maka pada proses selanjutnya, kelas yang sudah terpisah tersebut bisa diabaikan. Dalam skenario ini, akan tercipta $n-1$ buah fitur.

Dasar pemikiran yang melandasi GE Tatami adalah bahwa $n-1$ buah dimensi sebenarnya telah cukup untuk memisahkan n kelas pada data yang terpisah secara hirarkikal (penjelasan lebih lanjut pada subbab 4.8). Dalam ruang fitur yang terbentuk, akan dibutuhkan $n-1$ buah garis linear sederhana untuk

memisahkan n kelas tersebut. Gambar 1.1 pada Bab I menunjukkan bagaimana 3 buah kelas dapat terpisah dalam sebuah ruang fitur dua dimensi.

Dalam GE Tatami, untuk n buah kelas $\{C1, C2, C3, \dots, Cn\}$, dilakukan proses GE Multi untuk mencari individu-individu terbaik. Seperti yang telah dibahas dalam subbab sebelumnya, setiap individu dalam GE Multi memiliki n buah *fitnessvalue* $\{f1, f2, f3, \dots, fn\}$. Di sini dipilih satu *fitness value* terbesar dari semua individu. *Fitnessvalue* terbesar dari semua individu merepresentasikan kelas yang paling terpisah dari kelas lain. Kelas ini selanjutnya disimbolkan sebagai $C*1$. Fenotip dari individu terbaik dengan nilai *fitness* tertinggi selanjutnya disebut $F1$, dan merupakan bagian dari fitur baru yang akan digunakan dalam proses klasifikasi.

Dalam proses selanjutnya, data-data dalam kelas $C*1$ dihilangkan, sehingga diperoleh subset data baru yang terdiri dari $\{C1, C2, C3, \dots, Cn\} - C*1$. Subset data baru ini akan memiliki $n-1$ buah kelas. Kemudian kembali dilakukan GE Multi seperti sebelumnya. Perulangan kedua ini akan menghasilkan $F2$ yang juga merupakan bagian dari fitur baru yang akan digunakan dalam proses klasifikasi. Perulangan akan dilakukan sebanyak $n-1$ kali. Dalam setiap perulangan, jumlah kelas yang terlibat akan berkurang satu, sehingga pada perulangan ke $n-1$ hanya akan tersisa 2 kelas yang kemudian dipisahkan oleh F_{n-1} .

Sebagai contoh, untuk dataset Iris yang terdiri dari 4 fitur (petal length, sepal length, petal width, dan sepal width) serta 3 kelas (iris-setosa, iris-virginica dan iris-versicolor), GE Tatami menghasilkan 2 buah fitur:

- sepal_length
- $(\text{petal_length}) - (\text{sqr}(\text{sqr}(\text{sepal_length}) + \text{sqr}(\text{sqr}(\text{abs}(\text{petal_length}) + (\text{sqr}(\text{sqr}(\text{petal_length} + \text{petal_width}) / 2)) - (\text{abs}(-((\text{sepal_width}) - (\text{petal_width})))))) / 2)) / 2))$

Fitur pertama memiliki *fitnessvalue* tertinggi (bernilai 1,0) untuk memisahkan iris-setosa dengan kedua kelas lain. Fitur kedua memiliki *fitness value* tertinggi (bernilai 0,96) untuk memisahkan iris-virginica dan iris-versicolor (kelas iris-setosa diabaikan karena sudah terpisah pada fitur pertama).

Fitur pertama pada GE Tatami didapatkan dengan cara menjalankan proses GE Multi, seperti yang telah dijelaskan dalam subbab 3.3.4.3. Proses GE Multi tersebut menghasilkan 3 buah individu dengan 3 nilai *fitness* terbaik untuk setiap kelas:

- *sepal_length*
memiliki nilai *fitness* terbaik sebesar 1.0 untuk pemisahan antara iris setosa dengan kedua kelas lain
- $(\exp(\text{sepal_width})) * ((\text{sepal_length}) / (\text{petal_width}))$
memiliki nilai *fitness* terbaik sebesar 0.98 untuk pemisahan antara iris versicolor dengan kedua kelas lain.
- $\sqrt{\sqrt{(\text{abs}(\sqrt{\sqrt{((\text{sepal_width})^2 + ((\text{sepal_width})^2 - (\sqrt{\sqrt{((\text{petal_length}) - (\text{sepal_length}) + \text{sepal_width}/2)}))}) - ((\text{petal_width}) + \text{petal_width})/2)})) + \text{sepal_length})/2}}$
memiliki nilai *fitness* terbaik sebesar 0.98 untuk pemisahan antara iris virginica dengan kedua kelas lain.

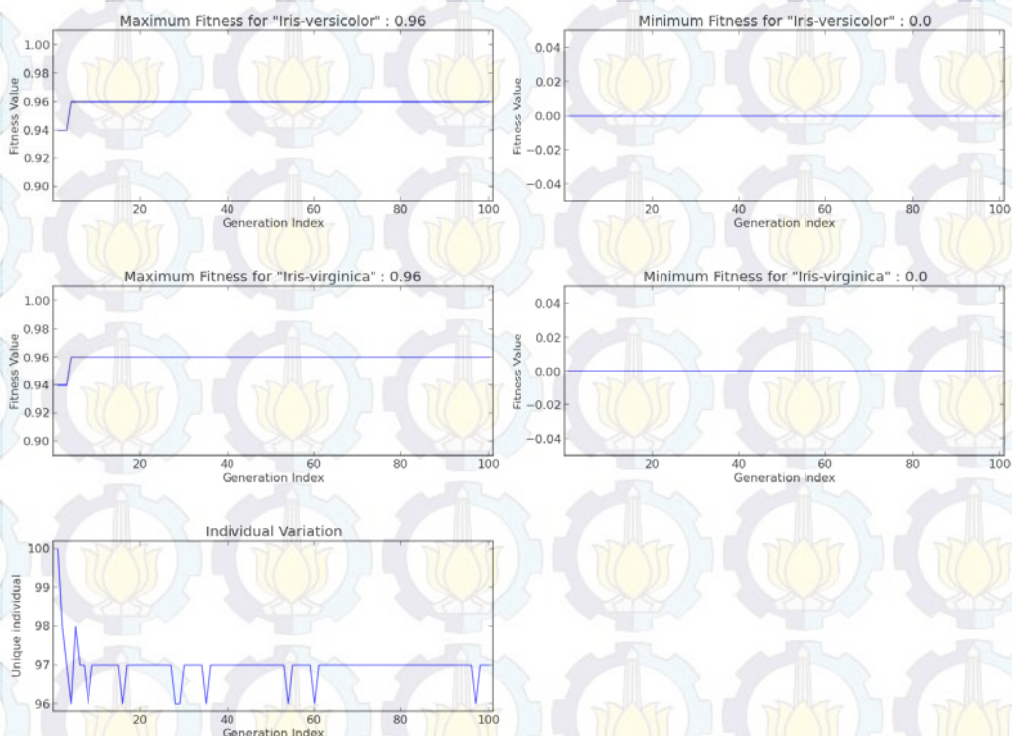
Pada proses ini, diperoleh bahwa nilai *fitness* tertinggi adalah 1.0 untuk memisahkan iris setosa dengan kedua kelas lain. Nilai *fitness* ini dimiliki oleh individu dengan fenotip *sepal_length*. Maka fitur *sepal_length* ditambahkan sebagai fitur baru dalam GE Tatami.

Kemudian proses ekstraksi fitur dilanjutkan dengan menghilangkan data yang memiliki kelas iris-setosa, dengan asumsi bahwa fitur *sepal_length* pada langkah pertama tadi telah memisahkan iris-setosa dengan kedua kelas lainnya. Maka data yang baru hanya terdiri dari dua buah kelas, yakni iris versicolor dan iris virginica.

Proses GE Multi kembali dijalankan pada data yang kini hanya terdiri dari dua kelas (iris virginica dan iris versicolor). Proses ini menghasilkan dua individu dengan dua nilai *fitness* terbaik. Kedua individu tersebut memiliki fenotip yang sama, yakni $(\text{petal_length}) - (\sqrt{\sqrt{(\text{sepal_length} + \sqrt{\sqrt{(\text{abs}(\text{petal_length}) + (\sqrt{\sqrt{(\text{petal_length} + \text{petal_width})/2}} - (\text{abs}((\text{sepal_width}) - (\text{petal_width})))))) / 2}})) / 2}}$ yang memiliki nilai *fitness* 0,96. Angka 0,96 merupakan nilai akurasi decision tree jika fitur tersebut digunakan pada data yang

kelas iris-setosa nya telah dihilangkan. Oleh sebab itu, terjadi penurunan nilai jika dibandingkan dengan *fitness* value yang dibuat oleh GE Multi pada langkah pertama. Ini wajar, karena jumlah data *false negative* ikut berkurang seiring dengan penghilangan kelas iris-setosa.

Langkah pertama pada GE Tatami, pada dasarnya adalah sama dengan GE Multi. Oleh sebab itu proses perubahan nilai *fitness* dari setiap generasi adalah sama dengan yang telah tergambar pada gambar 3.6. Sedangkan perubahan nilai *fitness* dari setiap generasi pada langkah kedua digambarkan pada gambar 3.11

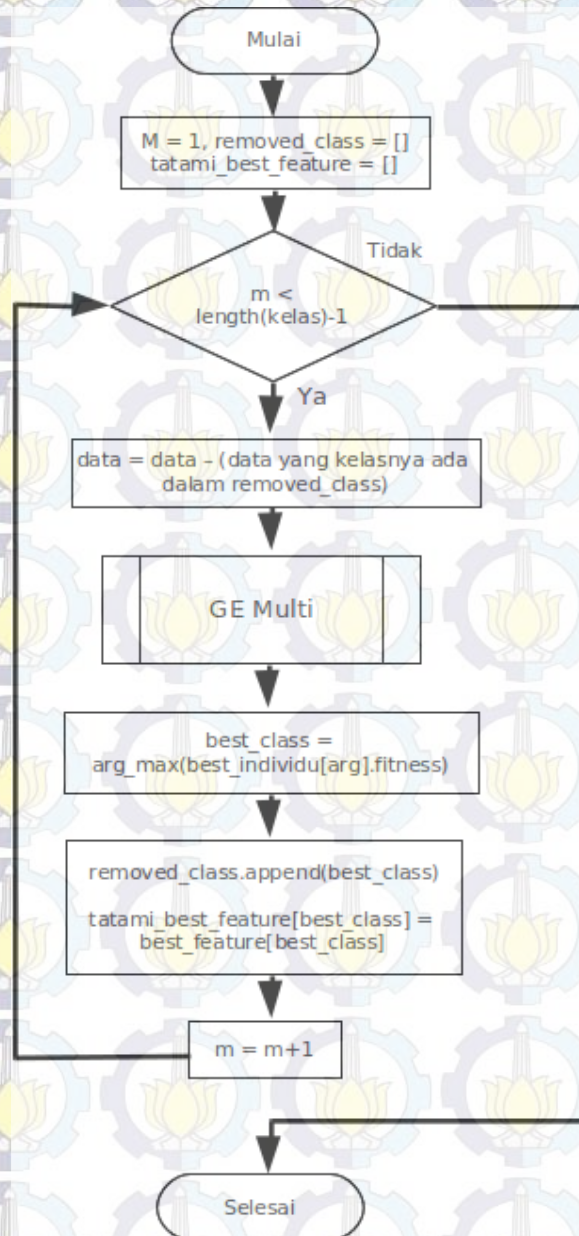


Gambar 3.11 Perubahan Nilai *Fitness* Maksimum dalam 100 Generasi pada Perulangan Kedua GE Tatami

Data yang ada kemudian ditransformasi dalam menggunakan fitur-fitur baru yang telah di-generate GE Tatami. Data hasil transformasi tadi kemudian digunakan sebagai input bagi *decision tree classifier*. Penggunaan ketiga fitur ini mengakibatkan akurasi decision tree meningkat menjadi 98,67%, dibandingkan dengan penggunaan fitur asli (sepal length, sepal width, petal length, dan petal

width) yang hanya memberikan akurasi sebesar 96%. Dalam kasus ini, penggunaan GE Tatami dan GE Multi menghasilkan nilai akurasi yang sama. Namun fitur yang dibuat oleh GE Tatami lebih sedikit (2 fitur saja).

Flowchart GE Tatami disajikan pada gambar 3.12.



Gambar 3.12 Flowchart GE Tatami.

3.3.4.5 Metode GE Gavrilis

Metode GE Gavrilis merupakan implementasi dari penelitian yang dilakukan oleh Gavrilis (Gavrilis, Tsoulous, & Dermatas, *Selecting and Constructing Features Using Grammatical Evolution*, 2008). Dalam metode ini akan di-*generate* set-set fitur yang masing-masing diwakili oleh satu individu. Berbeda dengan GE Multi dan GE Tatami, di sini satu individu mewakili satu set fitur.

Pengukuran *fitness* dalam metode ini dilakukan dengan mengukur akurasi *classifier* secara empiris.

Jumlah fitur yang di-*generate* dalam GE Gavrilis akan berkisar antara nol sampai tak terhingga.

3.3.5 Pengukuran Performa Fitur

Untuk pengukuran performa fitur sebagai acuan perbandingan atas semua metode di subbab 3.3.4, digunakan *Decision Tree classifier* yang merupakan salah satu algoritma umum dalam permasalahan klasifikasi. Semua metode yang telah dibahas pada subbab sebelumnya akan digunakan untuk men-*generate* sekumpulan fitur baru. Fitur-fitur tersebut akan digunakan sebagai data baru bagi *Decision Tree*. Diharapkan metode GE Tatami akan memperoleh hasil yang lebih baik dibandingkan dengan metode-metode lain.

Dalam pengujian akan digunakan berbagai macam data. Selain data-data sintesis yang sengaja dibuat untuk menguji hipotesa, percobaan juga akan dilakukan pada dataset iris, e.coli, dan balanced-scale yang telah umum dipakai dalam penelitian-penelitian sejenis. Data-data non-sintesis yang digunakan didapatkan dari website UCI-Machine Learning (<http://archive.ics.uci.edu/ml/>)

BAB 4

HASIL PENELITIAN DAN PEMBAHASAN

Percobaan diimplementasikan dengan menggunakan bahasa pemrograman Python 2.7 dan beberapa library eksternal. Adapun library eksternal yang digunakan adalah scipy, numpy, matplotlib dan scikit-learn (Pedregosa, et al., 2011).

Source code program dan hasil lengkap pengujian telah diletakkan di repository *github*. Repository tersebut berlisensi open-source dan bisa diakses secara publik di alamat <https://github.com/goFrendiAsgard/feature-extractor> dengan lisensi GNU, sehingga bebas dimodifikasi dan digunakan guna penelitian lebih lanjut.

4.1 Pengujian

Dalam percobaan yang dilakukan terdapat beberapa metode (GA Select, GE Global, GE Multi, GE Tatami dan GE Gavrilis) yang diujikan pada berbagai macam data. Selain itu, disertakan pula akurasi *classifier* tanpa ekstraksi fitur sebagai pembanding. Setiap data diuji dengan menggunakan *5 fold cross validation*.

4.1.1 Dataset Sintesis 01

Untuk kepentingan uji coba penelitian, maka dibuat beberapa buah dataset sintesis menggunakan aplikasi spreadsheet. Dalam penelitian ini digunakan libre-office.

Pada dataset sintesis 01, terdapat 3 buah kelas, yakni *defender*, *demon hunter* dan *wizard*. Ketiga kelas tersebut didapatkan dengan melakukan kalkulasi berdasarkan 4 fitur (*defense*, *attack*, *agility*, *stamina*). Keempat fitur yang ada bersifat random uniform dan memiliki range antara 0-10 dengan pembulatan satu angka di belakang koma. Dataset sintesis 01 terdiri dari 460 data yang terdiri dari 139 *defender*, 171 *demon hunter* dan 150 *wizard*. Data ini bisa diakses pada https://github.com/goFrendiAsgard/feature-extractor/blob/master/synthesis_01.csv.

Adapun Formula yang digunakan untuk menggolongkan kelas adalah sebagai berikut: $=IF(defense/attack \geq 1.4, "defender", IF(agility \geq stamina,$

"*demon hunter*", "*wizard*")). Pemilihan angka-angka pada formula semata-mata untuk membuat dataset *balanced* (memiliki jumlah data yang hampir sama untuk semua kelas). Formula tersebut dimaksudkan untuk membuat data yang hirarkikal. Melalui perbandingan fitur *defense* dan *attack*, kelas *defender* terpisah dari kedua kelas lainnya (*demon hunter* dan *wizard*). Melalui perbandingan fitur *agility* dan *stamina*, kelas *demon hunter* terpisah dari kelas *wizard*.

Hasil pengujian menunjukkan bahwa GE Multi dan GE Tatami memeberikan hasil yang cukup baik dengan jumlah fitur yang relatif sedikit. Pada fold 2, GE Gavrilis memperoleh akurasi tertinggi, namun memiliki jumlah fitur yang sangat banyak (48 fitur).

Hasil pengujian terhadap dataset sintesis 01 ditunjukkan dalam tabel 4.1

Tabel 4.1 Hasil Pengujian pada Dataset Sintesis 01

Percobaan		Tanpa Ekstraksi Fitur		Ekstraksi Fitur: GA Select		Ekstraksi Fitur: GE Global		Ekstraksi Fitur: GE Multi		Ekstraksi Fitur: GE Tatami		Ekstraksi Fitur: GE Gavrilis	
		Akurasi (%)	Jml. Fitur	Akurasi (%)	Jml. Fitur	Akurasi (%)	Jml. Fitur	Akurasi (%)	Jml. Fitur	Akurasi (%)	Jml. Fitur	Akurasi (%)	Jml. Fitur
Un-fold	Train	73.0	4	73.0	3	77.8	1	99.3	3	100	2	85.6	3
	Test	73.0		73.0		77.8		99.3		100		85.6	
	Total	73.0		73.0		77.8		99.3		100		85.6	
Fold 1	Train	75.9	4	75.9	3	78.9	1	100	3	100	2	84.5	61
	Test	67.7		67.7		35.5		76.6		81.1		83.3	
	Total	74.3		74.3		70.4		95.4		96.3		84.3	
Fold 2	Train	73.7	4	73.7	3	78.9	1	100	3	100	2	100	48
	Test	70.0		70.0		36.7		74.4		65.6		80.0	
	Total	73.0		73.0		70.6		95.0		93.2		96.0	
Fold 3	Train	71.6	4	71.6	3	77.5	1	100	3	100	2	85.1	3
	Test	75.5		75.5		25.5		86.6		86.7		86.7	
	Total	72.3		72.3		67.3		97.3		97.4		85.4	
Fold 4	Train	73.5	4	73.5	3	78.6	1	100	3	100	2	85.4	3
	Test	74.4		74.4		36.6		77.7		77.7		76.6	
	Total	73.7		73.7		70.4		95.6		95.6		83.7	
Fold 5	Train	75.6	4	75.6	3	81.0	1	100	3	100	2	87.0	2
	Test	70.0		42.2		42.2		94.4		82.2		72.2	
	Total	74.5		73.4		73.5		98.9		96.5		84.1	

4.1.2 Dataset Sintesis 02

Data sintesis 02 memiliki struktur yang hampir sama dengan data sintesis 01. Pada dataset ini terdapat 4 fitur dan 4 kelas.

Pada dataset sintesis 02, terdapat 3 buah kelas, yakni defender, demon hunter, monk dan wizard. Ketiga kelas tersebut didapatkan dengan melakukan kalkulasi berdasarkan 4 fitur (defense, attack, agility, stamina). Keempat fitur yang ada bersifat random uniform dan memiliki range antara 0-10 dengan pembulatan satu angka di belakang koma. Dataset sintesis 02 terdiri dari 613 data yang terdiri dari 184 defender, 148 demon hunter, 135 monk dan 146 wizard. Data ini bisa diakses pada alamat https://github.com/goFrendiAsgard/feature-extractor/blob/master/synthesis_02.csv.

Formula yang digunakan pada dataset sintesis 02 adalah sebagai berikut: $\text{=IF}(\text{defense}/\text{attack} \geq 1.6, \text{"defender"}, \text{IF}(\text{agility}/\text{stamina} \geq 1.3, \text{"demon hunter"}, \text{IF}(\text{stamina} > \text{defense}, \text{"monk"}, \text{"wizard"})))$. Sama seperti pada dataset sintesis 01, formula untuk membuat kelas pada data sintesis 02 juga dimaksudkan untuk membuat data yang hirarkikal. Melalui perbandingan fitur defense dan attack, kelas defender terpisah dari ketiga kelas lainnya (demon hunter, monk dan wizard). Melalui perbandingan fitur agility dan stamina, kelas demon hunter terpisah dari kelas monk dan wizard. Terakhir, melalui perbandingan fitur stamina dan defense, kelas monk dan wizard terpisah satu sama lain. Dibandingkan dengan data sintesis 01, data sintesis 02 ini memiliki struktur hirarkikal yang lebih dalam, dikarenakan jumlah kelasnya lebih banyak.

Pada Data sintesis 02, GE Multi tidak lagi memberikan performa sebaik pada dataset sintesis 01. Hal tersebut disebabkan karena dengan semakin banyaknya kelas, pemisahan satu kelas terhadap semua kelas lain akan menjadi semakin sulit. Sebaliknya GE Tatami justru menunjukkan hasil yang lebih baik, dikarenakan struktur hirarkikal yang lebih tampak.

Hasil pengujian terhadap dataset sintesis 02 ditunjukkan dalam tabel 4.2

Tabel 4.2 Hasil Pengujian pada Dataset Sintesis 02

Percobaan	n	Tanpa Ekstraksi Fitur		Ekstraksi Fitur: GA Select		Ekstraksi Fitur: GE Global		Ekstraksi Fitur: GE Multi		Ekstraksi Fitur: GE Tatami		Ekstraksi Fitur: GE Gavrilis	
		Akurasi (%)	Jml. Fitur	Akurasi (%)	Jml. Fitur	Akurasi (%)	Jml. Fitur	Akurasi (%)	Jml. Fitur	Akurasi (%)	Jml. Fitur	Akurasi (%)	Jml. Fitur
Un-fold	Train	77.9	4	77.9	4	70.8	1	100	4	100	3	90.3	12
	Test	77.9		77.9		70.8		100		100		90.3	
	Total	77.9		77.9		70.8		100		100		90.3	
Fold 1	Train	78.7	4	78.7	4	73.2	1	99.4	4	100	3	89.8	12
	Test	76.0		76.0		33.1		46.3		62.8		72.7	
	Total	78.1		78.1		65.2		88.9		92.6		86.4	
Fold 2	Train	76.4	4	76.4	4	70.5	1	100	4	100	3	89.4	12
	Test	77.6		77.6		27.2		72.7		74.3		85.9	
	Total	76.6		76.6		61.9		94.6		94.9		88.7	
Fold 3	Train	79.6	4	79.6	4	71.7	1	99.3	3	100	3	90.0	12
	Test	68.6		68.6		34.7		64.4		83.4		68.6	
	Total	77.4		77.4		64.4		92.5		96.7		85.8	
Fold 4	Train	79.0	4	79.0	4	70.7	1	100	4	100	3	90.2	12
	Test	72.7		72.7		40.5		71.0		61.1		80.1	
	Total	77.8		77.8		64.7		94.2		92.3		88.2	
Fold 5	Train	78.0	4	78.0	4	71.7	1	100	4	100	3	86.9	12
	Test	73.5		73.5		32.2		83.4		94.2		70.2	
	Total	77.1		77.1		63.9		96.7		98.8		83.6	

4.1.3 Dataset Sintesis 03

Dataset sintesis 03 merupakan dataset ideal untuk GE Tatami. Dataset ini terdiri dari 400 data yang terdiri dari 91 kelas A, 81 kelas B, 81 kelas C, 59 kelas D, dan 88 kelas E. Dalam dataset ini terdapat 5 kelas, A, B, C, D dan E dan 7 fitur (f1, f2, f3, f4, f5, n1, dan n2). Fitur n1 dan fitur n2 adalah fitur noise yang bersifat random uniform dengan ketelitian 1 angka di belakang koma. Keberadaan kedua fitur noise tersebut dimaksudkan untuk menguji apakah metode-metode yang ada dapat memfilter dan menghilangkan noise. Fitur f1 sampai f5 merupakan fitur utama yang diperoleh melalui serangkaian perhitungan.

Pembagian data ke dalam 5 kelas ditentukan berdasarkan 4 fitur tersembunyi m1, m2, m3, dan m4. Fitur m1, m2, m3 dan m4 bersifat random uniform dan memiliki range antara 0-10 dengan ketelitian 1 angka di belakang koma. Penentuan kelas menggunakan formula sebagai berikut: $=IF(m1 < 0.5, "A",$

$IF(m2 < 0.5, "B", IF(m3 < 0.5, "C", IF(m4 < 0.5, "D", "E"))))$). Fitur m1 memisahkan kelas A dengan keempat kelas lainnya (B, C, D, dan E). Fitur m2 memisahkan kelas B dengan ketiga kelas lain (C, D, dan E). Fitur m3 memisahkan kelas C dari kelas D dan E. Terakhir, fitur m4 berfungsi untuk memisahkan kelas D dan E.

Seperti yang telah diungkapkan, fitur m1 sampai m4 tidak ditunjukkan secara eksplisit kepada sistem, melainkan disembunyikan secara implisit melalui formula-formula matematis sederhana ke dalam lima fitur tampak f1 sampai f5. Hal ini dimaksudkan untuk menguji kemampuan GE Tatami dalam memperoleh kembali fitur-fitur utama tersembunyi (m1-m4) berdasarkan fitur-fitur tampak. Proses penyembunyian m1-m4 ke dalam f1-f5 dilakukan sebagai berikut:

- f1 diperoleh secara acak dengan formula $=ROUND(RAND() * 9.9 + 0.1, 3)$
- f2 diperoleh dengan menggunakan rumus $f2 = f1/m1$
- f3 diperoleh dengan menggunakan rumus $f3 = f2/m2$
- f4 diperoleh dengan menggunakan rumus $f4 = m3/f1$
- f5 diperoleh dengan menggunakan rumus $f5 = f4/m4$

Diharapkan GE Tatami akan berhasil menemukan m1-m4 dengan menggunakan konstruksi matematis dari f1-f5. Iterasi pertama dalam GE Tatami diharapkan mampu menemukan $f1/f2$ atau bentuk lain yang sebanding dengan m1 ($f2 = f1/m1$ sehingga $m1 = f1/f2$). Fitur ini seharusnya mampu memisahkan kelas A dengan keempat kelas lainnya (B, C, D dan E). Pada iterasi kedua, GE Tatami diharapkan mampu menemukan $f2/f3$ atau bentuk lain yang sebanding dengan m2 ($f3 = f2/m2$ sehingga $m2 = f2/f3$). Demikian seterusnya hingga m4-m5 atau fitur-fitur yang sebanding dengan itu ditemukan.

Data sintesis 03 dapat diakses melalui alamat https://github.com/goFrendiAsgard/feature-extractor/blob/master/synthesis_03.csv

Hasil pengujian menunjukkan bahwa untuk dataset sintesis 03, GE Tatami menunjukkan hasil yang sangat baik.

Berikut adalah keempat fitur yang berhasil di-generate oleh GE Tatami pada skenario unfold guna memisahkan kelima kelas yang ada:

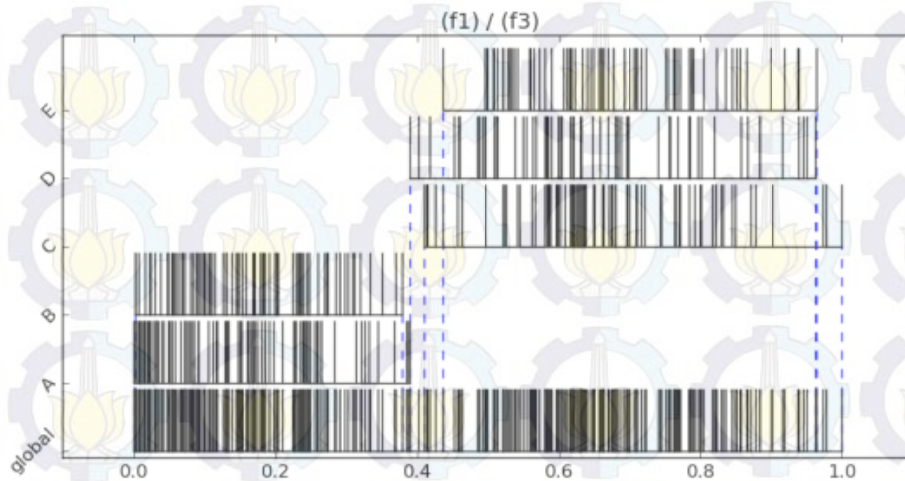
- $(f2) / (f1)$

- $(f1) / (f3)$
- $\text{sqrt}(\text{sqrt}(((f4) * (\text{sqrt}(\text{sqrt}(f1+f1)/2)))) - (n1)+n1)/2)$
- $(f5) / (f4)$

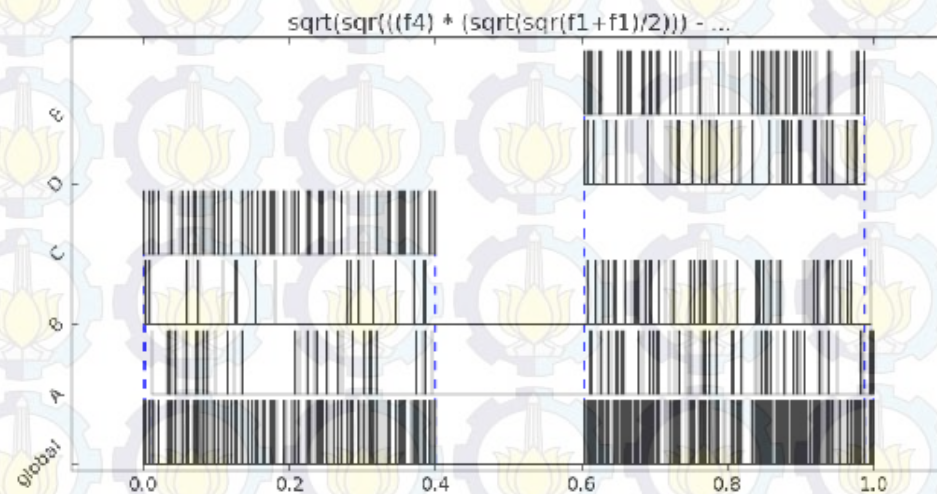
Dari keempat fitur tersebut, tampak bahwa fitur pertama $(f2) / (f1)$ sebanding dengan $m1$, sebab $m1 = f2/f1$. Fitur ke empat $(f5) / (f4)$ juga sebanding dengan $m4$, sebab $m4 = f5/f4$. Sementara itu, fitur kedua $(f1) / (f3)$ dan fitur ketiga $\text{sqrt}(\text{sqrt}(((f4) * (\text{sqrt}(\text{sqrt}(f1+f1)/2)))) - (n1)+n1)/2)$ walaupun masing-masing tidak sama dengan $m2$ dan $m3$, namun mampu memisahkan kelas-kelas yang sesuai dengan baik.

Fitur kedua sanggup memisahkan kelas B dengan kelas C, D dan E, seperti yang ditunjukkan dalam gambar 4.1. Fitur ketiga sanggup memisahkan kelas C dengan kelas D dan E, seperti yang ditunjukkan dalam gambar 4.2.

Hasil pengujian terhadap dataset sintesis 03 ditunjukkan dalam tabel 4.3



Gambar 4.1 Proyeksi data terhadap fitur $(f1)/(f3)$.



Gambar 4.2 Proyeksi data terhadap fitur $\sqrt{\sqrt{((f_4) * (\sqrt{\sqrt{(f_1 + f_1) / 2})}) - ((n_1) + n_1) / 2)}$.

Tabel 4.3 Hasil Pengujian pada Dataset Sintesis 03

Percobaan	Tanpa Ekstraksi	Ekstraksi Fitur: GA		Ekstraksi Fitur: GE		Ekstraksi Fitur: GE		Ekstraksi Fitur: GE		Ekstraksi Fitur: GE	
		Akurasi (%)	Jml. Fitur	Akurasi (%)	Jml. Fitur	Akurasi (%)	Jml. Fitur	Akurasi (%)	Jml. Fitur	Akurasi (%)	Jml. Fitur
Un-fold	Train	72.5	7	72.5	6	67.0	1	98.2	5	100	4
	Test	72.5		72.5		67.0		98.2		100	
	Total	72.5		72.5		67.0		98.2		100	
Fold 1	Train	74.4	7	74.4	6	67.9	1	99.0	5	100	4
	Test	26.5		26.5		27.8		45.5		63.2	
	Total	65.0		65.0		60.0		88.5		92.7	
Fold 2	Train	72.9	7	72.9	6	66.9	1	100.	5	100.	4
	Test	62.0		62.0		27.8		68.3		94.9	
	Total	70.7		70.7		59.2		93.7		99.0	
Fold 3	Train	71.3	7	71.3	6	69.4	1	99.6	5	100	4
	Test	29.1		29.1		24.0		55.7		65.8	
	Total	63.0		63.0		60.5		91.0		93.2	
Fold 4	Train	72.9	7	72.9	4	66.9	1	98.4	5	100	4
	Test	26.5		27.8		26.5		50.6		78.4	
	Total	63.7		64.0		59.0		89.0		95.7	
Fold 5	Train	72.2	7	72.2	6	68.8	1	99.3	5	100	4
	Test	24.0		25.3		48.1		59.4		63.2	
	Total	62.7		63.0		64.7		91.5		92.7	

4.1.4 Dataset Iris

Dataset iris merupakan dataset yang cukup banyak dipakai dalam penelitian. Data ini terdiri dari 3 kelas (Iris-Setosa, Iris-Versicolor, dan Iris-Virginica) serta 4 atribut fitur original (Sepal Length, Sepal Width, Petal Length, dan Petal Width) yang memiliki ketelitian 1 angka di belakang koma. Dataset iris bersifat multi-variate, terdiri dari 150 data (50 iris-setosa, 50 iris-versicolor, dan 50 iris-virginica). Dataset iris dapat didownload dari website UCI Machine Learning dengan alamat (<http://archive.ics.uci.edu/ml/datasets/Iris>)

Hasil pengujian terhadap data iris menunjukkan hasil yang hampir seimbang untuk GE Global, GE Multi, GE Tatami dan GE Gavrilis. Namun tampak bahwa GE Gavrilis memberikan hasil yang sedikit lebih unggul dibandingkan metode-metode lain.

Hasil pengujian terhadap dataset iris ditunjukkan dalam tabel 4.4

Tabel 4.4 Hasil Pengujian pada Dataset Iris.

Percobaan	Tanpa Ekstraksi Fitur	Ekstraksi Fitur: GA Select		Ekstraksi Fitur: GE Global		Ekstraksi Fitur: GE Multi		Ekstraksi Fitur: GE Tatami		Ekstraksi Fitur: GE Gavrilis	
		Akura-rasi (%)	Jml. Fitur	Akura-rasi (%)	Jml. Fitur	Akura-rasi (%)	Jml. Fitur	Akura-rasi (%)	Jml. Fitur	Akura-rasi (%)	Jml. Fitur
Un-fold	Train	96.04	2	98.6	1	98.6	3	98.6	2	98.6	1
	Test	96.0		98.6		98.6		98.6		98.6	
	Total	96.0		98.6		98.6		98.6		98.6	
Fold 1	Train	96.64	2	99.1	1	99.1	3	98.3	2	99.1	1
	Test	86.6		96.6		96.6		96.6		96.6	
	Total	94.6		98.6		98.6		98.0		98.6	
Fold 2	Train	95.84	2	100	1	99.1	3	99.1	2	100	3
	Test	96.6		96.6		83.3		66.6		96.6	
	Total	96.0		99.3		96.0		92.6		99.3	
Fold 3	Train	96.64	2	98.3	1	98.3	3	99.1	2	98.3	1
	Test	93.3		76.6		100		100		100	
	Total	96.0		94.0		98.6		99.3		98.6	
Fold 4	Train	95.84	2	99.1	1	99.1	3	99.1	2	99.1	1
	Test	96.6		93.3		96.6		96.6		96.6	
	Total	96.0		98.0		98.6		98.6		98.6	
Fold 5	Train	96.64	2	98.3	1	98.3	3	99.1	2	99.1	3
	Test	93.3		93.3		93.3		96.6		96.6	
	Total	96.0		97.3		97.3		98.6		98.6	

4.1.5 Dataset E-Coli

Data E-Coli merupakan dataset yang cukup banyak dipakai dalam penelitian. Data ini terdiri dari 7 atribut (mcg, gvh, lip, chg, aac, alm1, alm2) dan 6 kelas (cp, im, imU, om, omL, pp), yang terdiri dari 335 record (143 cp, 77 im , 52 pp, 35 imU, 20 om, 5 omL, 2 imL, 2 imS). Masing-masing atribut memiliki ketelitian satu angka di belakang koma. Dataset ini merupakan klasifikasi terhadap berbagai varian dari bakteri E-Coli. Dataset E-Coli dapat didownload pada alamat (<http://archive.ics.uci.edu/ml/datasets/Ecoli>).

Berbeda dengan pengujian-pengujian sebelumnya, di sini justru GA Select menghasilkan akurasi yang paling tinggi dibandingkan keempat skenario lain. Adapun demikian, saat semua data digunakan untuk training sekaligus testing, GE Tatami tampak berhasil memberikan akurasi yang paling tinggi. Dalam beberapa kasus, terlihat bahwa penggunaan fitur original justru memberikan hasil yang lebih baik (semisal pada fold 1, 2 dan 4). Hal ini dikarenakan bahwa pengekstraksian fitur menjadi jumlah yang lebih sedikit dari fitur original juga menyebabkan hilangnya sebagian informasi yang berguna dalam proses klasifikasi.

Hasil pengujian terhadap dataset e-coli ditunjukkan dalam tabel 4.5

Tabel 4.5 Hasil Pengujian pada Dataset E-Coli

Percobaan	Tanpa Ekstraksi Fitur	Ekstraksi Fitur: GA Select		Ekstraksi Fitur: GE Global		Ekstraksi Fitur: GE Multi		Ekstraksi Fitur: GE Tatami		Ekstraksi Fitur: GE Gavrillis	
		Akurasi (%)	Jml. Fitur	Akurasi (%)	Jml. Fitur	Akurasi (%)	Jml. Fitur	Akurasi (%)	Jml. Fitur	Akurasi (%)	Jml. Fitur
Un-fold	Train	97.0	7	84.5	1	96.7	8	97.6	7	97.0	12
	Test	97.0		84.5		96.7		97.6		97.0	
	Total	97.0		84.5		96.7		97.6		97.0	
Fold 1	Train	97.4	7	87.4	1	98.8	8	96.3	7	97.7	12
	Test	73.8		58.4		53.8		53.8		69.2	
	Total	92.8		81.8		90.1		88.1		92.2	
Fold 2	Train	96.6	7	86.7	1	97.7	8	98.5	7	97.7	12
	Test	81.5		63.0		49.2		1.54		58.4	
	Total	93.7		82.1		88.3		79.7		90.1	
Fold 3	Train	97.7	7	89.3	1	99.2	8	97.0	7	98.5	26
	Test	70.7		53.8		80.0		69.2		53.8	
	Total	92.5		82.4		95.5		91.6		89.8	

Fold 4	Train	95.9	7	96.3	5	87.0	1	98.5	8	97.4	7	98.1	12
	Test	73.8		73.8		69.2		63.0		43.0		56.9	
	Total	91.6		91.9		83.6		91.6		86.9		90.1	
Fold 5	Train	97.0	7	97.0	4	86.7	1	98.1	8	96.3	7	98.8	18
	Test	73.8		75.3		61.5		38.4		38.4		67.6	
	Total	97.4		97.4		87.4		98.8		96.3		97.7	

4.1.6 Dataset Balanced-Scale

Dataset balanced-scale terdiri dari 625 data yang masing-masing terdiri dari 4 fitur(left_weight, left_distance, right_weight, right_distance) dan 3 kelas(B, R, L). Masing-masing atribut bertipe bilangan bulat positif dengan range antara 1-5. Dataset ini terdiri dari 625 record yang terdiri dari 49 B, 288 L, dan 288 R. Dataset ini dibuat untuk tujuan pengujian psikologi dan dapat didownload pada alamat (<http://archive.ics.uci.edu/ml/datasets/Balance+Scale>).

Hasil pengujian menunjukkan bahwa secara umum GE Multi lebih unggul dibandingkan semua metode lain.

Tabel 4.6 Hasil Pengujian pada Dataset Balanced-Scale

Percobaan	Tanpa Ekstraksi Fitur	Akurasi (%)	Jml. Fitur	Ekstraksi Fitur: GA Select	Akurasi (%)	Jml. Fitur	Ekstraksi Fitur: GE Global	Akurasi (%)	Jml. Fitur	Ekstraksi Fitur: GE Multi	Akurasi (%)	Jml. Fitur	Ekstraksi Fitur: GE Tatami	Akurasi (%)	Jml. Fitur	Ekstraksi Fitur: GE Gavrilis	Akurasi (%)	Jml. Fitur
Un-fold	Train	70.8	4	70.8	3	84.8	1	91.6	1	91.6	2	82.5	9					
	Test	70.8		70.8		84.8		91.6		91.6		82.5						
	Total	70.8		70.8		84.8		91.6		91.6		82.5						
Fold 1	Train	68.7	4	70.9	3	100	1	100	1	92.0	2	81.0	4					
	Test	67.4		70.7		91.8		91.8		85.3		81.3						
	Total	68.4		70.8		98.4		98.4		90.7		81.1						
Fold 2	Train	70.1	4	71.9	3	85.4	1	92.2	1	92.6	2	83.8	9					
	Test	61.7		66.6		69.9		89.4		82.9		78.8						
	Total	68.4		70.8		82.4		91.6		90.7		82.8						
Fold 3	Train	69.3	4	70.5	3	90.0	1	99.0	2	92.0	2	83.6	126					
	Test	65.0		72.3		66.6		85.3		71.5		81.3						
	Total	68.4		70.8		85.4		96.3		88.0		83.2						
Fold 4	Train	72.5	4	72.5	3	86.6	1	100	1	91.8	2	82.2	2					
	Test	68.2		68.2		78.0		73.9		91.0		77.2						
	Total	71.6		71.6		84.9		94.8		91.6		81.2						
Fold 5	Train	71.1	4	71.1	3	84.6	1	94.6	3	100	2	82.8	1					
	Test	69.9		69.9		82.9		57.7		66.6		51.2						
	Total	70.8		70.8		84.3		87.3		93.4		76.6						

4.1.7 Rata-Rata Hasil Pengujian

Guna mendapatkan gambaran performa secara umum, maka dilakukan perhitungan terhadap rata-rata hasil pengujian terhadap keenam dataset yang digunakan.

Rata-rata akurasi yang diperoleh dari pengujian terhadap keenam dataset menunjukkan bahwa GE Multi memberikan hasil terbaik, sementara GE Tatami menempati peringkat kedua.

Adapun GE Tatami menunjukkan keunggulan mutlak pada data data sintesis 02 dan data sintesis 03. Kedua dataset tersebut dapat terpisah secara hirarkikal berdasarkan fitur-fitur yang di-generate. Keduanya juga memiliki jumlah kelas yang relatif cukup banyak (lebih dari tiga). Ini menunjukkan bahwa GE Tatami unggul dalam meningkatkan akurasi decision tree pada data-data yang dapat terpisah secara hirarkikal.

Pada data E-Coli, secara umum tampak bahwa penggunaan ekstraksi fitur seperti GE Global, GE Multi, GE Tatami, dan GE Gavrilis tidak meningkatkan akurasi decision tree, bahkan cenderung memperburuk akurasi (kecuali pada sesi training di mana rata-rata akurasi yang diberikan GE Tatami adalah sebesar 98,2%). Ini menunjukkan dua hal. Yang pertama adalah bahwa pada data-data dengan jumlah fitur yang relatif banyak (lebih banyak dari jumlah kelas), maka penggunaan ekstraksi fitur seperti GE Global, GE Multi, GE Tatami, dan GE Gavrilis, mungkin tidak akan banyak membantu. Yang kedua, adalah bahwa GE Tatami memiliki kecenderungan untuk menciptakan fitur-space yang overfit. Hal ini tampak dari perbedaan akurasi antara sesi training dan testing yang cenderung cukup besar.

Rata-rata hasil pengujian ditunjukkan pada tabel 4.7

Tabel 4.7 Rata-Rata Hasil Pengujian pada Keenam Dataset

Percobaan		Tanpa Ekstraksi Fitur		Ekstraksi Fitur: GA Select		Ekstraksi Fitur: GE Global		Ekstraksi Fitur: GE Multi		Ekstraksi Fitur: GE Tatami		Ekstraksi Fitur: GE Gavrilis	
		Akurasi (%)	Jml Fitur	Akurasi (%)	Jml Fitur	Akurasi (%)	Jml Fitur	Akurasi (%)	Jml Fitur	Akurasi (%)	Jml Fitur	Akurasi (%)	Jml Fitur
iris	Train	96.2	4	96.2	2	98.9	1	98.8	3	98.9	2	99.0	2
	Test	93.7		93.7		92.5		94.7		92.5		97.5	
	Total	95.7		95.7		97.6		98.0		97.6		98.7	
balance-scale	Train	70.4	4	71.3	3	88.6	1	96.2	2	93.3	2	82.7	25
	Test	67.2		69.8		79.0		81.6		81.5		75.4	
	Total	69.8		71.0		86.7		93.3		91.0		81.2	
ecoli	Train	96.9	7	97.0	6	86.9	1	98.2	8	97.2	7	98.0	15
	Test	78.4		78.2		65.1		63.5		50.6		67.2	
	Total	93.4		93.4		82.7		91.5		88.2		92.0	
synthesis_01	Train	73.9	4	73.9	3	78.8	1	99.8	3	100	2	87.9	20
	Test	71.8		71.8		42.42		84.8		82.2		80.7	
	Total	73.5		73.5		71.7		96.9		96.5		86.5	
synthesis_02	Train	78.3	4	78.3	4	71.4	1	99.8	4	100	3	89.4	12
	Test	74.4		74.4		39.7		73.0		79.3		78.0	
	Total	77.5		77.5		65.2		94.5		95.9		87.2	
synthesis_03	Train	72.7	7	72.7	6	67.8	1	99.1	5	100	4	81.9	19
	Test	40.1		40.5		36.9		63.0		77.6		50.8	
	Total	66.2		66.3		61.7		92.0		95.5		75.8	
All Average	Train	81.4	5	81.6	4	82.1	1	98.6	4	98.2	3	89.8	16
	Test	70.9		71.4		59.3		76.8		77.3		74.9	
	Total	79.3		79.6		77.6		94.4		94.1		86.9	

4.2 Analisis

Berdasarkan data-data pengujian yang didapat dalam subbab 4.1, dilakukan analisis untuk menjelaskan karakteristik dan kelemahan dari GE Tatami.

4.2.1 Karakteristik GE Tatami

Dari hasil percobaan, tampak bahwa GE Tatami menunjukkan hasil yang cukup baik pada data-data sintesis dan data iris. Dalam hal ini proses *grammatical evolution* berhasil menemukan fitur-fitur yang sanggup memisahkan data sesuai dengan hipotesis.

Pada dataset sintesis 03, tampak bahwa GE Tatami berhasil menemukan fitur-fitur yang sebanding (dalam hal pemisahan kelas) dengan fitur asli (m_1 , m_2 , m_3 dan m_4) berdasarkan fitur-fitur tampak (f_1 , f_2 , f_3 , f_4 dan f_5). Hubungan antara fitur asli dan fitur tampak telah dijelaskan pada subbab 4.3. Adapun fitur-fitur yang berhasil di-*generate* oleh GE Tatami adalah sebagai berikut:

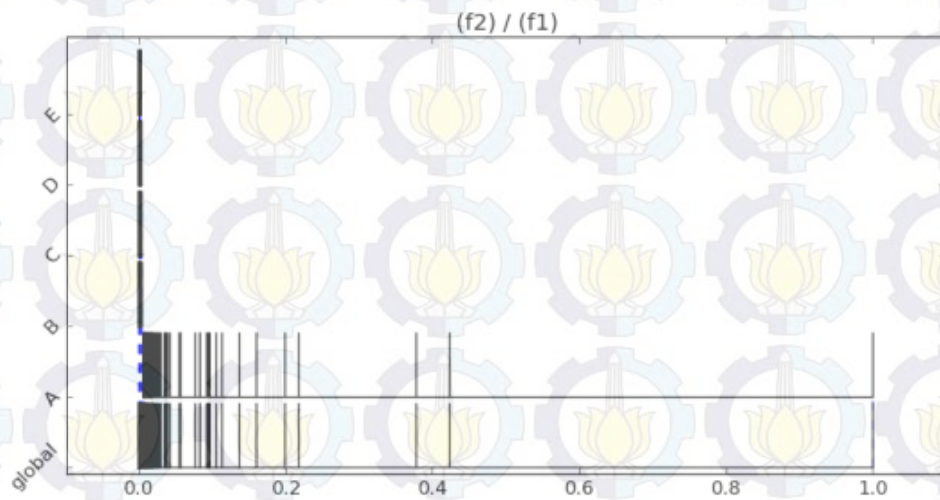
- $(f_2) / (f_1)$
- $(f_1) / (f_3)$
- $\text{sqrt}(\text{sqrt}(((f_4) * (\text{sqrt}(\text{sqrt}(f_1+f_1)/2)))) - (n_1)+n_1)/2)$
- $(f_5) / (f_4)$

Fitur $(f_2) / (f_1)$ sanggup memisahkan kelas A dan keempat kelas lainnya. Hal ini tampak seperti pada gambar 4.3. Pada gambar 4.3, tampak bahwa kelas B, C, D dan E berimpit di sebelah kiri, terpisah secara linear dari kelas A. Panjang garis vertikal menunjukkan banyaknya data yang menempati nilai yang sama.

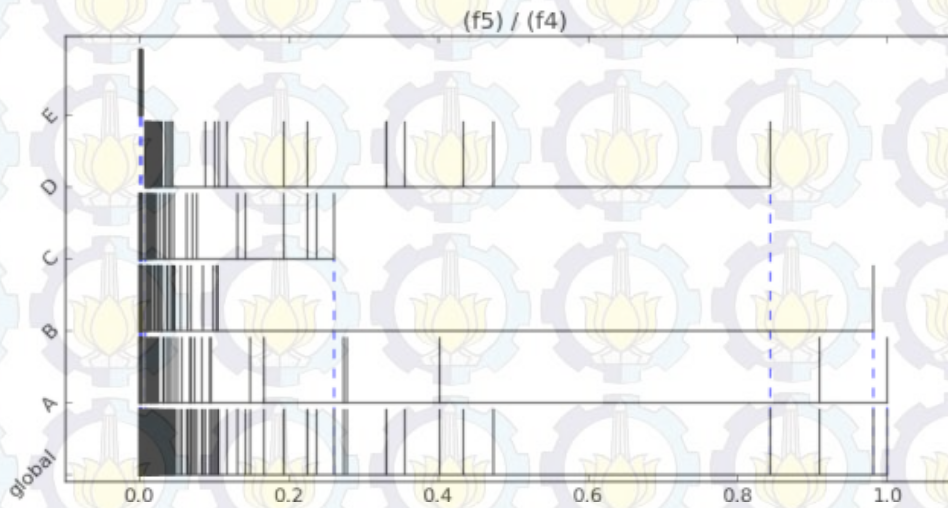
Setelah berhasil menemukan fitur $(f_2) / (f_1)$ yang memisahkan kelas A dan keempat kelas lain (B, C, D, dan E), GE Tatami mencari fitur yang paling baik memisahkan satu dari keempat kelas tersebut terhadap tiga kelas lainnya. Dalam proses ini ditemukan fitur $(f_1) / (f_3)$. Proyeksi data ke dalam fitur $(f_1) / (f_3)$ mengakibatkan kelas A tidak terpisah dengan kelas B. Namun hal ini tidak dipermasalahkan karena fitur sebelumnya, yakni $(f_2) / (f_1)$ telah memisahkan kelas A dengan semua kelas lain. Proyeksi data terhadap fitur $(f_1) / (f_3)$ telah digambarkan pada gambar 4.1.

Setelah kelas kedua fitur sebelumnya berhasil memisahkan kelas A dan B dari ketiga kelas lain (C, D, dan E), GE Tatami mencari fitur yang paling baik memisahkan satu dari ketiga kelas itu dari kedua kelas lainnya. Dalam proses ini ditemukan $\text{sqrt}(\text{sqrt}(((f_4) * (\text{sqrt}(\text{sqrt}(f_1 + f_1) / 2)))) - (n_1) + n_1) / 2)$ yang sanggup memisahkan C dari D dan E. Proyeksi data terhadap fitur ketiga ini telah digambarkan pada gambar 4.2.

Setelah hanya tersisa kelas D dan E, GE Tatami mencari fitur terakhir terakhir untuk memisahkan kedua kelas tersebut. Di sini ditemukan $(f_5) / (f_4)$. Fitur ini tampak berhasil memisahkan kelas D dan E dengan sangat baik, seperti yang ditunjukkan dalam gambar 4.4.



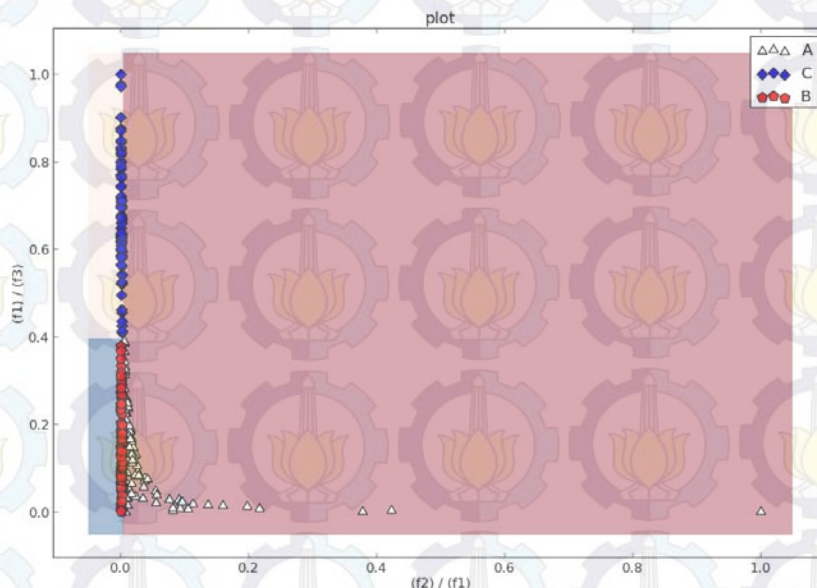
Gambar 4.3 Proyeksi Data terhadap Fitur $(f2) / (f1)$.



Gambar 4.4 Proyeksi Data terhadap Fitur $(f5) / (f4)$

Skenario GE Tatami tampak berhasil menciptakan feature space yang ideal bagi *classifierDecision Tree*, seperti ditunjukkan pada gambar 4.5. Fitur $(f2) / (f1)$ dan $(f3) / (f1)$, memberikan kemudahan bagi *decision tree* untuk membagi data sesuai dengan kelas yang ada.

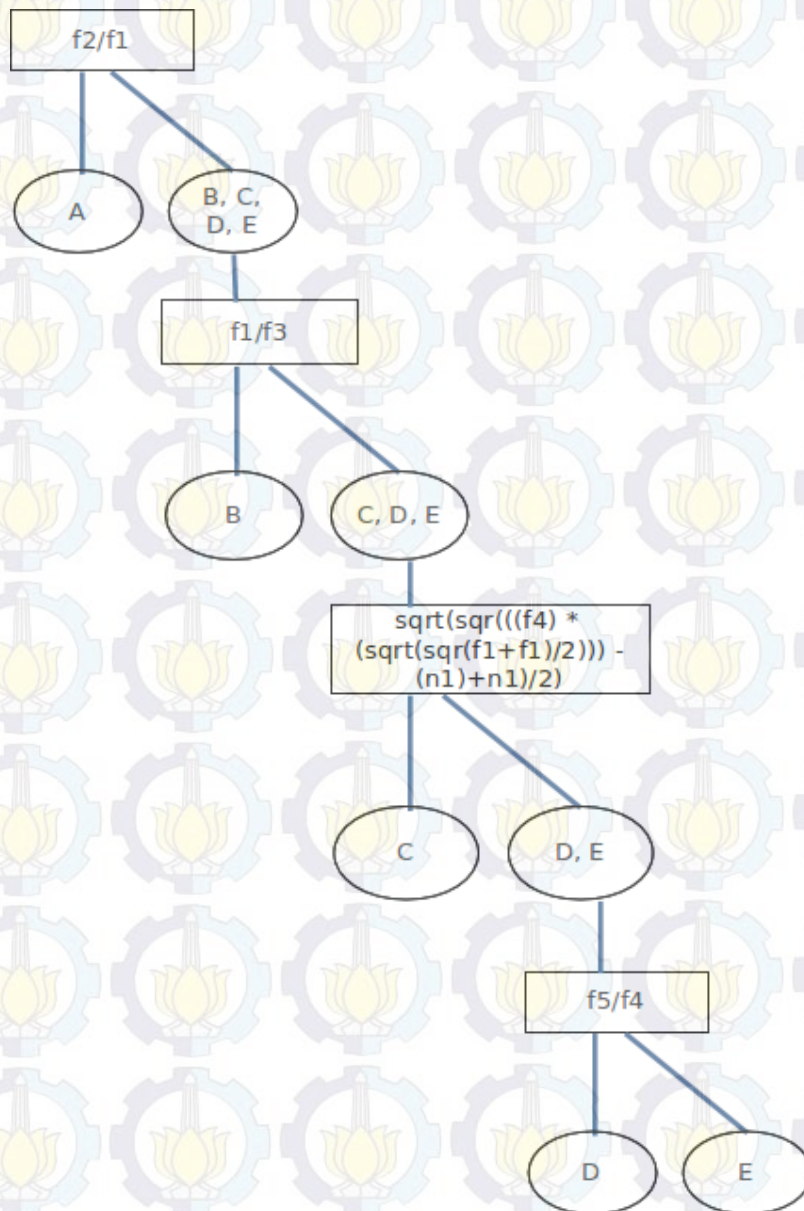
Cara kerja GE Tatami ini sangat sesuai dengan karakteristik *Decision Tree* yang memisahkan data secara hirarkikal seperti digambarkan dalam gambar 4.6.



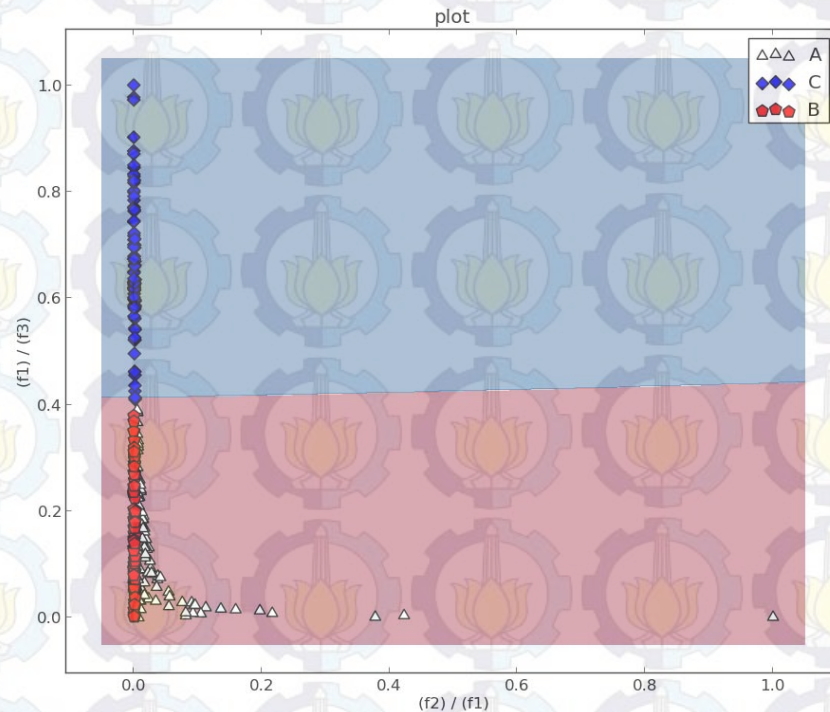
Gambar 4.5 Proyeksi Data terhadap Fitur $(f2) / (f1)$ dan Fitur $(f1) / (f3)$ dengan *Decision Tree Classifier*

Sekalipun GE Tatami menunjukkan hasil yang baik jika dikombinasikan dengan *Decision Tree*, namun tampaknya penggunaan *classifier* lain semisal SVM. Proyeksi data yang sama jika menggunakan *classifier* SVM ditunjukkan pada gambar 4.7. Fenomena ini menunjukkan bahwa Feature Space yang di-generate oleh GE Tatami hanya akan membantu untuk *classifier* yang memanfaatkan keterpisahan data per fitur tanpa mempedulikan pola dan bentuk sebaran data.

Dari cara kerja yang telah dipaparkan, tampak bahwa GE Tatami akan sangat unggul jika digunakan untuk mengekstraksi fitur pada data yang secara natur terpisah secara hirarkikal. Jika *grammar* yang dipakai sesuai, maka GE Tatami juga akan sanggup untuk mencari fitur-fitur pemisah data, sekalipun fitur-fitur pemisah data itu tidak ditunjukkan secara eksplisit (seperti pada kasus data sintesis 03)



Gambar 4.6 Decission Tree yang Terbentuk Berdasarkan Penggunaan Fitur-Fitur yang di-generate oleh GE Tatami.

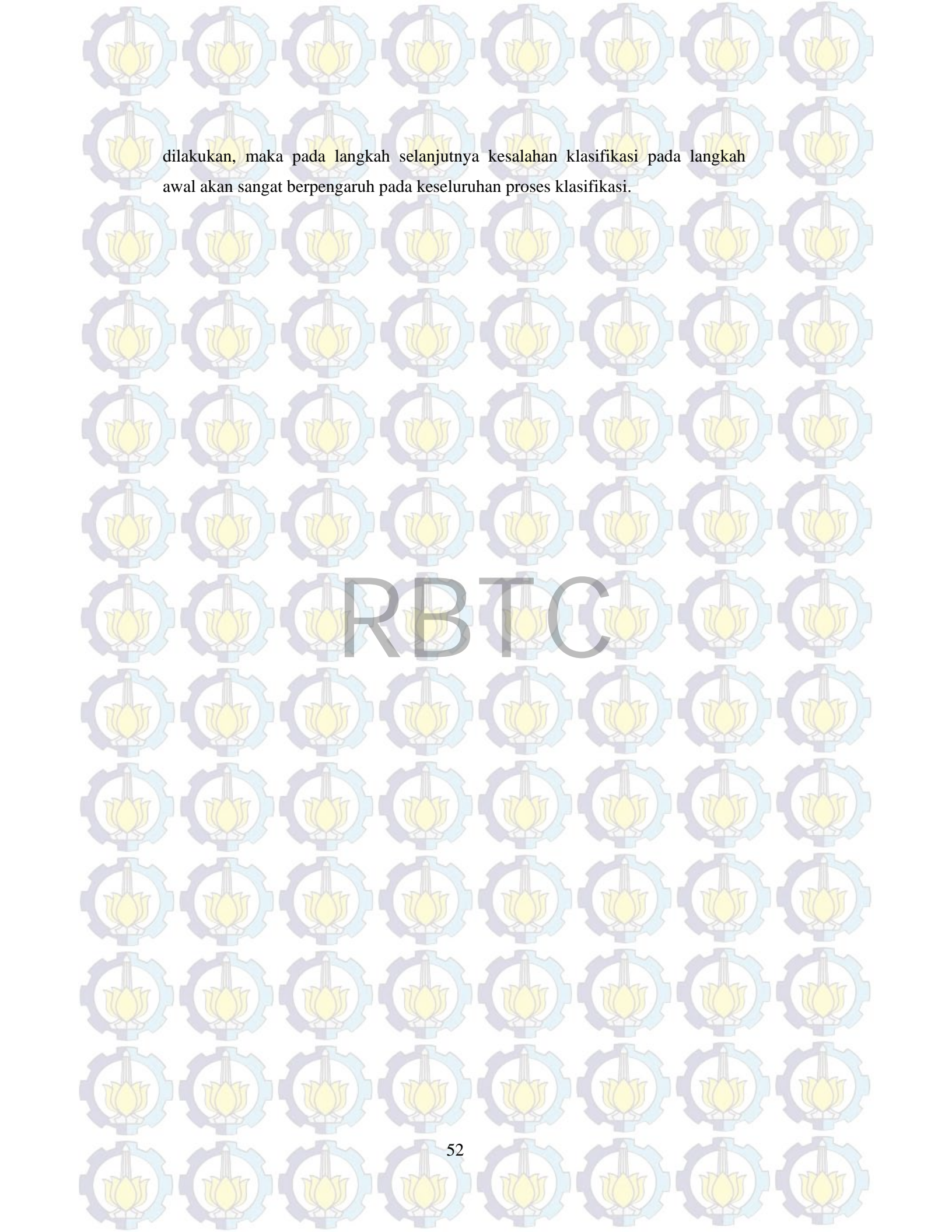


Gambar 4.7 Proyeksi Data terhadap Fitur $(f2) / (f1)$ dan Fitur $(f1) / (f3)$ dengan SVM Classifier.

4.2.2 Kelemahan GE Tatami

Pada semua kasus yang ada, GE Tatami menunjukkan hasil terbaik saat semua data digunakan sebagai training dan testing sekaligus. Namun pada pengujian cross-validation didapati bahwa performa GE Tatami menurun. Terkadang (seperti dalam Fold-2 di dataset E-Coli), sekalipun akurasi yang diberikan dalam training sangat tinggi, akurasi testing nya sangat rendah. Ini menunjukkan bahwa GE Tatami cenderung membuat fitur-fitur yang overfit.

Kelemahan GE Tatami yang lain adalah jika terjadi kegagalan pada langkah pertama, maka langkah-langkah selanjutnya akan menjadi tidak efektif. Pada langkah pertama (*Predefined process* GE Multi pada flowchart di gambar 3.10), GE Tatami harus berhasil menciptakan fitur yang memisahkan satu kelas tertentu dengan semua kelas lainnya secara cukup baik. Jika hal ini gagal



dilakukan, maka pada langkah selanjutnya kesalahan klasifikasi pada langkah awal akan sangat berpengaruh pada keseluruhan proses klasifikasi.

BAB 5

KESIMPULAN DAN SARAN

Bab ini memaparkan kesimpulan yang dapat diambil berdasarkan pada penelitian yang telah dilakukan. Dalam Bab 5 ini diuraikan juga tentang hal-hal yang perlu dipertimbangkan untuk pengembangan penelitian lebih lanjut. Penjelasan yang lebih terperinci tentang hal-hal tersebut diuraikan pada sub-bab berikut.

5.1 Kesimpulan

Hasil penelitian menunjukkan bahwa skenario GE Tatami berhasil membuat fitur-fitur yang membantu dalam proses klasifikasi untuk data sintesis dan iris. Namun metode tersebut gagal untuk menentukan fitur-fitur terbaik pada data *ecoli*. GE Tatami akan menghasilkan akurasi yang bagus jika ada proyeksi terhadap suatu fitur *ter-generate* yang menunjukkan keterpisahan menonjol antara satu kelas dengan semua kelas lainnya.

GE Tatami memberikan persyaratan yang lebih mudah dipenuhi daripada GE Global dan GE Multi. Pada GE Global, harus *ter-generate* sebuah fitur yang sanggup memisahkan setiap kelas. Pada GE Multi, harus *ter-generate* n buah fitur yang sanggup memisahkan n kelas dengan kelas-kelas lain. Sementara pada GE Tatami, jika ada satu kelas yang sudah berhasil dipisahkan dari kelas-kelas lain, maka untuk pencarian fitur berikutnya, kelas tersebut dapat diabaikan.

Walaupun GE Tatami memberikan persyaratan yang lebih mudah dipenuhi, namun kompleksitas yang diberikan lebih tinggi dari metode-metode lain. Hal ini dikarenakan GE Tatami perlu melakukan perhitungan ulang sebanyak jumlah kelas-1 kali.

GE Tatami juga menunjukkan kegagalan saat tidak berhasil *di-generate* fitur yang memisahkan satu kelas dengan kelas-kelas lain secara cukup menonjol. Selain itu, GE Tatami juga memiliki kecenderungan untuk membuat *feature space* yang overfit.

5.2 Saran

Hasil pengujian menunjukkan bahwa GE Tatami dan GE Multi memberikan hasil yang cukup baik. Dalam kasus-kasus ideal, GE Tatami tampak sanggup memberikan hasil yang sangat baik, namun dalam kasus-kasus lain, tampak bahwa GE Tatami menunjukkan hasil yang kurang baik. Sesuai dengan analisa yang dilakukan, kelemahan ini terjadi karena di langkah pertama, GE Tatami gagal memisahkan satu kelas dengan semua kelas lain. Hal ini selanjutnya berdampak pada proses selanjutnya. Oleh sebab itu, disarankan untuk menggabungkan fitur-fitur yang di-generate oleh GE Multi dan fitur-fitur yang di-generate oleh GE Tatami.

Dalam penelitian, *decision tree classifier* masih dilibatkan dalam perhitungan *fitness value*. Penggunaan *fitness function* yang lebih sederhana tanpa melibatkan *classifier* diharapkan dapat meningkatkan performa GE Multi dan GE Tatami.

Untuk penggunaan GE Tatami dalam kasus nyata, sebaiknya digunakan sample yang cukup banyak dikarenakan kecenderungannya untuk membuat fitur-fitur yang *overfit*.

DAFTAR PUSTAKA

- Conor, R. (2006). Grammatical Evolution Tutorial. *Gecco 2006*.
- Duda, R. O., Hart, P. E., & Stork, D. G. (2000). *Pattern Classification*. Wiley-Interscience.
- Gavrilis, D., Tsoulous, I. G., & Dermatas, E. (2008). Selecting and Constructing Features Using Grammatical Evolution. *Journal Pattern Recognition Letters*, 1358-1365.
- Gavrilis, D., Tsoulous, I. G., Georgoulas, G., & Glavas, E. (2005). Classification of Fetal Heart Rate Using Grammatical Evolution. *IEEE Workshop on Signal Processing Systems Design and Implementation*.
- Gunawan, G. F., Gosaria, S. C., & Arifin, A. Z. (2012). Grammatical Evolution For Feature Extraction In Local Thresholding Problem. *Jurnal Ilmu Komputer dan Informasi*, 5(2).
- Guo, L., Rivero, D., Dorado, J., Munteanu, C. R., & Pazos, A. (2011). Automatic feature extraction using genetic programming: An application to epileptic EEG classification. *Expert Systems with Applications*, 38, 10425-10436.
- Harper, R., & Blair, A. (2006). Dynamically Define Functions in Grammatical Evolution. *IEEE Congress of Evolutionary Computation*.
- Li, B., Zhang, P., Tian, H., Mi, S., Liu, D., & Ruo, G. (2011). A new feature extraction and selection scheme for hybrid fault diagnosis of gearbox. *Expert Systems with Applications*, 38, 10000-10009.
- Micheli-Tzanakou, E. (2000). *Supervised and Unsupervised Pattern Recognition: Feature Extraction in Computational Intelligence*. CRC Press.
- Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., et al. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12, 2825-2830.



Halaman ini sengaja dikosongkan

LAMPIRAN

1.1.Sample Data Synthesis_01

defense	attack	agility	stamina	class
9.9	9	7.3	7.4	wizard
10.4	9.6	5	1.6	demon hunter
6.4	1.4	3.6	6.3	defender
7	2.2	9.7	3.3	defender
7.3	1.4	8.7	7.6	defender
7.1	9.5	3.3	3.4	wizard
5.6	1.9	6.9	1.5	defender
4.8	2.4	9.2	7.2	defender
5	3.1	6.9	6.7	defender
8.5	6.6	4.5	10.4	wizard
3	7.5	8.5	3	demon hunter
7.1	1.7	10.2	4	defender
8.6	6	2.9	3.7	defender
8	4.6	7.1	1.9	defender
10.1	10.4	3.3	6.9	wizard
5.8	1	2.7	3.7	defender
7.7	10.6	1.6	5.8	wizard
3.5	4.1	10.5	2.3	demon hunter
7.2	6.1	3	1.9	demon hunter
2.9	9.1	10	2.8	demon hunter
2.5	3.5	9.8	1.2	demon hunter
5.8	5.5	8.8	2.8	demon hunter
6.1	6.9	1.7	8.6	wizard
5.9	10	6.7	9.4	wizard
11	5.6	4.4	6.5	defender
7.8	6.4	4.5	5	wizard
10.3	10.5	10.1	4.6	demon hunter
3.6	10.8	9.7	10.8	wizard
8.3	2.2	10.9	4.1	defender
3.9	2.2	9.5	9.8	defender
5.4	8.6	2.1	1.3	demon hunter
3.7	6.1	4.9	10.2	wizard
5.4	7.4	2.4	4.7	wizard
8.1	7.9	5.3	10.4	wizard
6.8	9.8	7.3	10.6	wizard
...

1.2.Sample Data Synthesis_02

defense	attack	agility	stamina	class
4.8	3.9	1.8	9.7	monk
8.9	2.5	10.5	7	defender
7.2	4.3	9.4	1.5	defender
4	9.9	5.9	6.8	wizard
5.5	8.7	3.4	4.9	wizard
8.4	4.8	1.5	3.1	defender
6.4	1.8	6.9	4.5	defender
2	9	1.5	1.3	wizard
1.4	10.8	9.1	4.3	demon hunter
1.8	4.9	3.5	3.7	wizard
4.5	1.5	8.2	3.8	defender
10.3	3.9	6.5	5.3	defender
4.1	8.7	6.9	8.4	wizard
9	8.1	2.5	3.1	wizard
6.8	1.6	1.8	4.2	defender
2.6	8.9	4.9	9.7	monk
4.5	2.7	5.6	7.3	defender
2.2	10.7	5.3	2.4	demon hunter
1.1	4	7.1	9.7	monk
2.9	7.2	9.9	3.1	demon hunter
7.2	9.3	9	1.2	demon hunter
8.8	2.2	7.4	7	defender
3.9	7.6	10	9	monk
4.5	2.6	7.9	5.2	defender
2.6	2.9	3.9	3.5	monk
1.6	4.2	8.1	10.4	monk
4.6	3.7	3.3	9.6	monk
7	9.2	1.7	10.3	monk
7.8	4.5	8.3	8.9	defender
2.1	10.3	2.7	2.2	wizard
2.5	3.3	1.5	2.6	wizard
1.1	8.9	5	4.6	wizard
8.8	3.2	5.7	9.6	defender
10.4	7.4	5.2	8.7	monk
9.8	5.3	7.9	5.2	defender
6.5	9.3	5	2	demon hunter
7	9.2	3.8	7.9	wizard
...

1.3.Sample Data Synthesis_03

f1	f2	f3	f4	f5	n1	n2	class
1.004	3.336875	11.29565	0.383855	7.429413	5.614	5.661	A
7.308	55.99563	56.69864	0.119703	0.329466	5.538	4.124	A
0.84	0.971972	1.372631	0.41778	1.744859	5.236	9.603	C
6.973	10.64605	11.94266	0.131109	5.778658	5.469	2.197	D
2.334	2.897753	3.179782	0.356137	1.03543	0.388	3.539	D
7.406	39.4607	39.72366	0.006252	0.006523	5.774	9.44	A
9.333	10.40953	11.76615	0.103944	17.05113	1.924	0.509	D
9.574	15.90112	110.7335	0.096251	0.109699	4.498	5.561	B
4.519	22.34915	89.98461	0.138601	0.180343	9.452	9.416	A
4.777	6.580423	7.52579	0.17734	0.190722	5.921	1.997	E
5.32	16.2231	21.43748	0.015583	0.834596	7.413	7.153	A
4.789	6.161582	6.293074	0.125942	0.332697	5.995	8.359	D
7.284	33.45157	44.96962	0.100392	0.721439	0.572	6.409	A
9.569	81.11533	104.2781	0.001426	0.003775	5.98	8.816	A
9.548	10.92337	142.6186	0.100586	0.28838	6.015	4.813	B
8.241	33.82372	479.1457	0.077071	0.118478	5.674	3.948	A
5.406	31.16762	32.82642	0.14485	1.404551	6.594	6.276	A
2.239	2.367671	13.48088	0.288409	0.467747	9.293	8.232	B
1.986	2.183807	2.661411	0.102755	0.14421	5.037	6.777	C
0.752	0.956239	2.511047	0.962074	19.23467	5.224	6.823	B
6.735	8.873998	37.68497	0.054965	0.258658	0.496	5.918	B
8.189	8.362252	36.28102	0.112378	0.638044	8.456	0.289	B
7.292	10.38659	1967.255	0.021248	0.025804	9.342	9.781	B
3.412	5.212485	6.270242	0.044326	0.188147	3.326	8.757	C
5.126	6.868544	161.2565	0.131215	0.155684	6.308	3.704	B
6.451	9.382496	15.26613	0.105607	0.155619	7.33	3.445	E
8.173	10.3741	17.15062	0.118841	0.363388	6.91	5.873	D
4.269	6.726389	8.447762	0.093529	0.105311	7.401	4.7	C
4.553	5.036589	6.551577	0.136641	0.159004	4.982	1.952	E
6.44	7.841033	727.3115	0.109334	0.152542	0.952	3.194	B
8.736	10.28948	15.42618	0.045318	0.135321	5.508	1.44	C
7.327	7.738717	8.844073	0.11993	0.421637	9.547	3.553	D
8.594	11.12612	17.43484	0.111697	0.720653	1.046	5.783	D
4.888	6.029918	8.623578	0.179089	0.451085	6.883	8.285	D
9.81	11.33042	12.01227	0.092807	0.112332	5.879	7.487	E
3.606	3.855979	4.353472	0.247321	0.346313	5.519	9.912	E
5.84	24.96811	39.27403	0.112376	0.297619	1.439	0.338	A
6.167	6.788922	8.882386	0.006027	0.006233	7.293	6.927	C
...

1.4.Sample Data Iris

petal_length	petal_width	sepal_length	sepal_width	class
5.1	3.5	1.4	0.2	Iris-setosa
4.9	3	1.4	0.2	Iris-setosa
4.7	3.2	1.3	0.2	Iris-setosa
4.6	3.1	1.5	0.2	Iris-setosa
5	3.6	1.4	0.2	Iris-setosa
5.4	3.9	1.7	0.4	Iris-setosa
4.6	3.4	1.4	0.3	Iris-setosa
5	3.4	1.5	0.2	Iris-setosa
4.4	2.9	1.4	0.2	Iris-setosa
4.9	3.1	1.5	0.1	Iris-setosa
7	3.2	4.7	1.4	Iris-versicolor
6.4	3.2	4.5	1.5	Iris-versicolor
6.9	3.1	4.9	1.5	Iris-versicolor
5.5	2.3	4	1.3	Iris-versicolor
6.5	2.8	4.6	1.5	Iris-versicolor
5.7	2.8	4.5	1.3	Iris-versicolor
6.3	3.3	4.7	1.6	Iris-versicolor
4.9	2.4	3.3	1	Iris-versicolor
6.6	2.9	4.6	1.3	Iris-versicolor
5.2	2.7	3.9	1.4	Iris-versicolor
5	2	3.5	1	Iris-versicolor
6.3	3.3	6	2.5	Iris-virginica
5.8	2.7	5.1	1.9	Iris-virginica
7.1	3	5.9	2.1	Iris-virginica
6.3	2.9	5.6	1.8	Iris-virginica
6.5	3	5.8	2.2	Iris-virginica
7.6	3	6.6	2.1	Iris-virginica
4.9	2.5	4.5	1.7	Iris-virginica
7.3	2.9	6.3	1.8	Iris-virginica
6.7	2.5	5.8	1.8	Iris-virginica
7.2	3.6	6.1	2.5	Iris-virginica
5.9	3	5.1	1.8	Iris-virginica
...

1.5.Sample Data E-Coli

mcg	gvh	lip	chg	aac	alm1	alm2	class
0.49	0.29	0.48	0.5	0.56	0.24	0.35	cp
0.07	0.4	0.48	0.5	0.54	0.35	0.44	cp
0.56	0.4	0.48	0.5	0.49	0.37	0.46	cp
0.59	0.49	0.48	0.5	0.52	0.45	0.36	cp
0.23	0.32	0.48	0.5	0.55	0.25	0.35	cp
0.06	0.61	0.48	0.5	0.49	0.92	0.37	im
0.44	0.52	0.48	0.5	0.43	0.47	0.54	im
0.63	0.47	0.48	0.5	0.51	0.82	0.84	im
0.23	0.48	0.48	0.5	0.59	0.88	0.89	im
0.34	0.49	0.48	0.5	0.58	0.85	0.8	im
0.43	0.4	0.48	0.5	0.58	0.75	0.78	im
0.46	0.61	0.48	0.5	0.48	0.86	0.87	im
0.85	0.53	0.48	0.5	0.53	0.52	0.35	imS
0.63	0.49	0.48	0.5	0.54	0.76	0.79	imS
0.75	0.55	1	1	0.4	0.47	0.3	imL
0.7	0.39	1	0.5	0.51	0.82	0.84	imL
0.72	0.42	0.48	0.5	0.65	0.77	0.79	imU
0.79	0.41	0.48	0.5	0.66	0.81	0.83	imU
0.83	0.48	0.48	0.5	0.65	0.76	0.79	imU
0.69	0.43	0.48	0.5	0.59	0.74	0.77	imU
0.79	0.36	0.48	0.5	0.46	0.82	0.7	imU
0.78	0.33	0.48	0.5	0.57	0.77	0.79	imU
0.78	0.68	0.48	0.5	0.83	0.4	0.29	om
0.63	0.69	0.48	0.5	0.65	0.41	0.28	om
0.67	0.88	0.48	0.5	0.73	0.5	0.25	om
0.61	0.75	0.48	0.5	0.51	0.33	0.33	om
0.67	0.84	0.48	0.5	0.74	0.54	0.37	om
0.74	0.9	0.48	0.5	0.57	0.53	0.29	om
0.73	0.84	0.48	0.5	0.86	0.58	0.29	om
0.75	0.76	0.48	0.5	0.83	0.57	0.3	om
0.77	0.57	1	0.5	0.37	0.54	0.01	omL
0.66	0.49	1	0.5	0.54	0.56	0.36	omL
0.71	0.46	1	0.5	0.52	0.59	0.3	omL
0.74	0.49	0.48	0.5	0.42	0.54	0.36	pp
0.7	0.61	0.48	0.5	0.56	0.52	0.43	pp
0.66	0.86	0.48	0.5	0.34	0.41	0.36	pp
0.73	0.78	0.48	0.5	0.58	0.51	0.31	pp
0.65	0.57	0.48	0.5	0.47	0.47	0.51	pp
...

1.6.Sample Data Balanced Scale

left_weight	left_distance	right_weight	right_distance	class
1	1	1	1	B
1	1	1	2	R
1	1	1	3	R
1	1	1	4	R
1	1	1	5	R
1	1	1	2	R
1	1	1	2	R
1	1	1	5	R
1	2	1	1	L
1	2	1	2	B
1	2	1	3	R
1	2	1	4	R
1	2	1	5	R
1	2	2	1	B
1	2	2	2	R
1	3	1	1	L
1	3	1	2	L
1	3	1	3	B
1	3	1	4	R
1	3	1	5	R
1	3	2	1	L
1	3	2	2	R
1	3	2	3	R
1	3	2	4	R
1	3	2	5	R
1	3	3	1	B
1	3	3	2	R
1	3	3	3	R
1	3	3	4	R
1	5	1	1	L
1	5	1	2	L
1	5	1	3	L
1	5	1	4	L
1	5	1	5	B
1	5	3	1	L
1	5	3	2	R
1	5	3	3	R
1	5	3	4	R
...

BIODATA PENULIS



Go Frendi Gunawan S.Kom adalah putra tunggal dari pasangan Hadi Gunawan dan Hioe Linda. Lahir di Malang pada tahun 1987. Menempuh pendidikan Sekolah Dasar di SDK Cor Jesu Malang, Sekolah Lanjutan Tingkat Pertama di SLTP Celaket 21 dan Lulus SMA dari SMAK Santa Maria Malang pada tahun 2005, dan melanjutkan studi S1 di STIKI, hingga lulus pada tahun 2009.

Penulis menaruh minat pada topik-topik yang terkait dengan kecerdasan buatan, sistem informasi geografis, dan pemrograman web.

Sehari-harinya penulis mengajar sebagai dosen di STIKI Malang, serta terlibat dalam berbagai proyek open source semisal No-CMS, kokoropy, dan groceryCRUD. Berbagai repository program tersebut telah dipublikasikan secara *free* di <https://github.com/goFrendiAsgard>. Penulis dapat dihubungi via e-mail dengan alamat gofrendiasgard@gmail.com.