

**PEMBANGKITAN POHON KEPUTUSAN DENGAN METODE
GENETIC PROGRAMMING PADA KASUS PENENTUAN
PENDERITA DIABETES MELITUS**

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:

Farizky Novanda Pramuditya

NIM: 135150200111009



**PROGRAM STUDI TEKNIK INFORMATIKA
JURUSAN TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2020**

PENGESAHAN

PEMBANGKITAN POHON KEPUTUSAN DENGAN METODE *GENETIC PROGRAMMING* PADA KASUS PENENTUAN PENDERITA DIABETES MELITUS

SKRIPSI

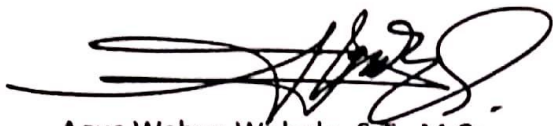
Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh :
Farizky Novanda Pramuditya
NIM: 135150200111009

Skripsi ini telah diuji dan dinyatakan lulus pada
9 Januari 2020
Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I

Dosen Pembimbing II



Agus Wahyu Widodo, S.T., M.Cs.
NIP: 19740805 200112 1 001



Bayu Rahayudi, S.T., M.T.
NIP: 19740712 200604 1 001

Mengetahui
Ketua Jurusan Teknik Informatika



Istetio Kurniawan, S.T., M.T., Ph.D.
NIP: 19710518 200312 1 001

PERNYATAAN ORISINALITAS

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, di dalam naskah skripsi ini tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu perguruan tinggi, dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis disitasi dalam naskah ini dan disebutkan dalam daftar referensi.

Apabila ternyata didalam naskah skripsi ini dapat dibuktikan terdapat unsur-unsur plagiasi, saya bersedia skripsi ini digugurkan dan gelar akademik yang telah saya peroleh (sarjana) dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku (UU No. 20 Tahun 2003, Pasal 25 ayat 2 dan Pasal 70).

Malang, 23 Januari 2020

Farizky Novanda Pramuditya

NIM: 135150200111009



ABSTRAK

Farizky Novanda Pramuditya, Pembangkitan Pohon Keputusan dengan Metode *Genetic Programming* pada Kasus Penentuan Penderita Diabetes Melitus

Pembimbing: Agus Wahyu Widodo S.T., M.Cs. dan Bayu Rahayudi S.T., M.T.

Diabetes melitus merupakan penyakit yang berbahaya karena dapat menyebabkan timbulnya penyakit lain pada tubuh. Penyakit yang dapat muncul antara lain stroke, serangan jantung, rusaknya fungsi mata, dan gagal ginjal. Oleh karena itu diperlukan proses yang dapat membantu diagnosis penyakit ini. Metode yang digunakan adalah *genetic programming* yang merupakan salah satu metode algoritma evolusi dengan representasi individu menggunakan pohon. Metode tersebut akan digunakan untuk menghasilkan pohon keputusan yang dapat digunakan untuk melakukan diagnosis penyakit. Pembentukan pohon keputusan tersebut akan sesuai dengan data latih yang digunakan. Parameter *genetic programming* seperti jumlah populasi, nilai iterasi maksimal, nilai *mutation rate* dan *crossover rate* akan menjadi variabel bebas. Parameter jumlah populasi, nilai *crossover rate* dan nilai *mutation rate* memberi pengaruh besar terhadap nilai *fitness* yang dihasilkan. Nilai iterasi maksimal tidak memberi pengaruh karena terjadinya konvergensi dini. Pohon keputusan yang dihasilkan menggunakan parameter terbaik kemudian digunakan untuk menentukan apakah pasien menderita diabetes atau tidak. Hasil yang didapatkan adalah pohon keputusan tersebut memberikan nilai akurasi sebesar 66,11%.

Kata kunci: *genetic programming*, pohon keputusan, diabetes melitus

ABSTRACT

Farizky Novanda Pramuditya, Decision Tree Induction Using Genetic Programming for Determining Diabetes Mellitus Patient

Supervisors: Agus Wahyu Widodo S.T., M.Cs. dan Bayu Rahayudi S.T., M.T.

Diabetes melitus is a disease that can cause many health complications including stroke, heart attack, kidney failure, and damage to eyes. Method that can help to diagnose this disease is sorely needed. Genetic programming is method used in this research. Genetic programming is one evolutionary algorithm that uses parse tree as its solution representation. This method will produce decision tree as its output which will be used to diagnose patients in the testing dataset. Genetic programming parameters like number of populations, maximum iterations, mutation and crossover rate will be changed in the testing part of this research. Number of populations, mutation and crossover rate affect fitness value. Maximum iterations don't affect fitness value because of fitness value stagnation in the beginning part of iterations. Decision tree produced with this method using the best parameters will be used for determining diabetes mellitus patient. Accuracy decision tree for determining diabetes mellitus patient from dataset is 66.11%.

Keywords: *genetic programming, decision tree, diabetes mellitus*

DAFTAR ISI

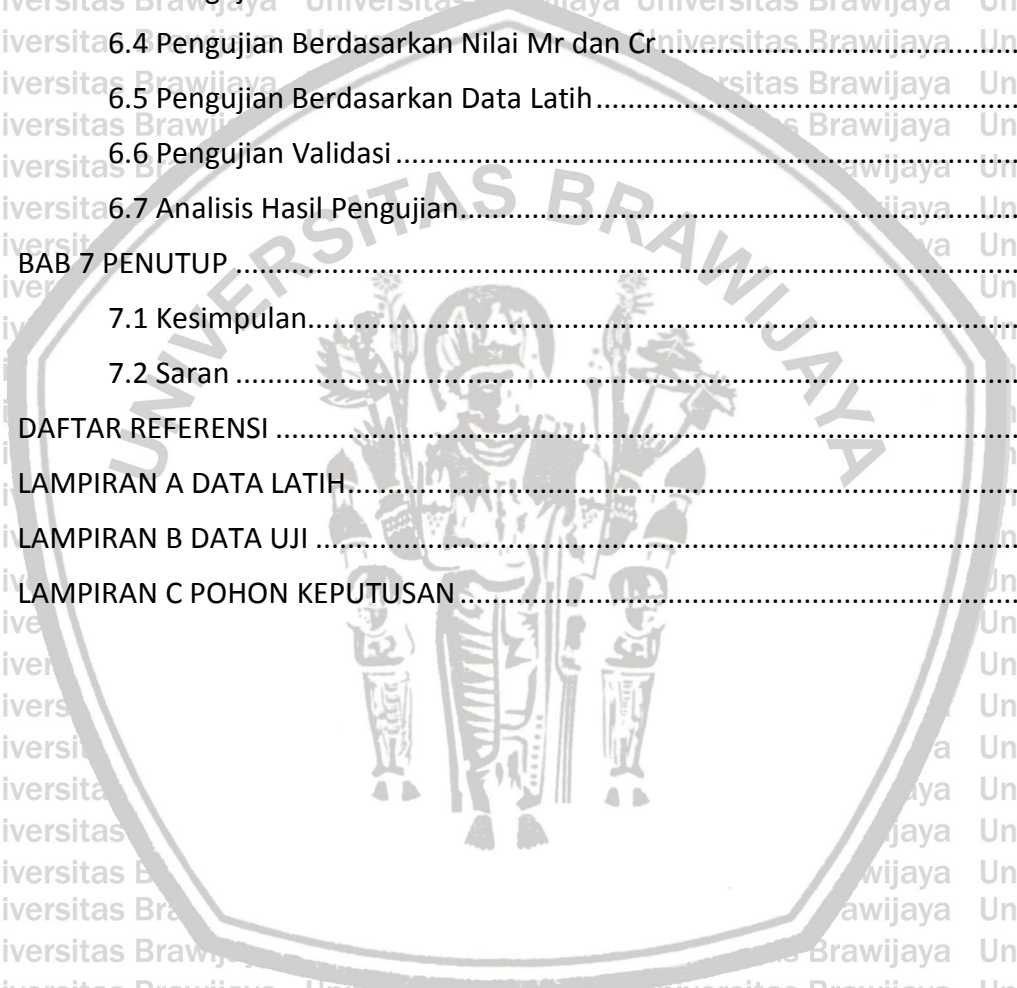
PENGESAHAN	ii
PERNYATAAN ORISINALITAS	iii
PRAKATA	iv
ABSTRAK	v
ABSTRACT	vi
DAFTAR ISI	vii
DAFTAR TABEL	x
DAFTAR GAMBAR	xi
DAFTAR SOURCE CODE	xii
DAFTAR LAMPIRAN	xiii
BAB 1 PENDAHULUAN	1
1.1 Latar Belakang	1
1.2 Rumusan Masalah	2
1.3 Tujuan	2
1.4 Manfaat	2
1.5 Batasan Masalah	2
1.6 Sistematika Pembahasan	3
BAB 2 LANDASAN KEPUSTAKAAN	4
2.1 Kajian Pustaka	4
2.2 Diabetes Melitus	4
2.3 Pohon Keputusan	5
2.4 Genetic Programming	5
2.4.1 Inisialisasi	6
2.4.2 Crossover	6
2.4.3 Mutasi	7
2.4.4 Seleksi	8
BAB 3 METODOLOGI PENELITIAN	9
3.1 Studi Literatur	9
3.2 Analisis Kebutuhan	9
3.3 Pengumpulan Data	9



3.4 Model Perancangan.....	10
3.5 Implementasi.....	10
3.6 Pengujian.....	11
3.7 Analisis.....	11
3.8 Kesimpulan.....	12
BAB 4 PERANCANGAN.....	13
4.1 Tahapan Penyelesaian Masalah.....	14
4.1.1 Inisialisasi.....	14
4.1.2 Mutasi.....	16
4.1.3 Reproduksi.....	17
4.1.4 <i>Crossover</i>	17
4.2 Perhitungan Manual.....	19
4.2.1 Skenario Perhitungan Manual.....	19
4.2.2 Representasi Individu.....	19
4.2.3 Inisialisasi Populasi Awal.....	20
4.2.4 Mutasi.....	20
4.2.5 Reproduksi.....	21
4.2.6 <i>Crossover</i>	22
4.3 Perancangan Pengujian.....	23
4.3.1 Perancangan Pengujian Berdasarkan Jumlah Populasi.....	23
4.3.2 Perancangan Pengujian Berdasarkan Nilai Bobot pada <i>Fitness</i>	24
4.3.3 Perancangan Pengujian Berdasarkan Jumlah Iterasi.....	24
4.3.4 Perancangan Pengujian Berdasarkan Nilai μ_r dan C_r	25
4.3.5 Perancangan Pengujian Berdasarkan Data Latih.....	26
BAB 5 IMPLEMENTASI.....	27
5.1 Spesifikasi Sistem.....	27
5.1.1 Spesifikasi Perangkat Keras.....	27
5.1.2 Spesifikasi Perangkat Lunak.....	27
5.2 Implementasi Sistem.....	27
5.2.1 Inisialisasi.....	27
5.2.2 Iterasi.....	31

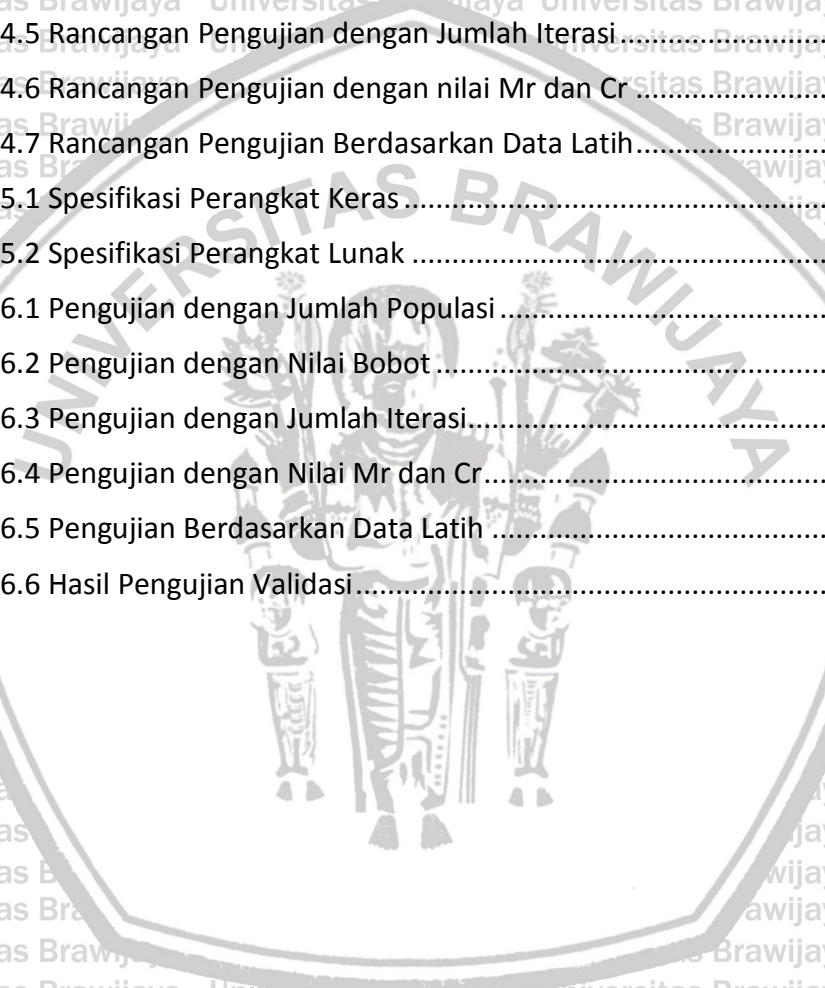


5.2.3 Mutasi	31
5.2.4 Reproduksi	32
5.2.5 Crossover	32
BAB 6 PENGUJIAN DAN ANALISIS	34
6.1 Pengujian Berdasarkan Jumlah Populasi	34
6.2 Pengujian Berdasarkan Nilai Bobot pada Perhitungan <i>Fitness</i>	35
6.3 Pengujian Berdasarkan Jumlah Iterasi	37
6.4 Pengujian Berdasarkan Nilai Mr dan Cr	38
6.5 Pengujian Berdasarkan Data Latih	39
6.6 Pengujian Validasi	40
6.7 Analisis Hasil Pengujian	41
BAB 7 PENUTUP	46
7.1 Kesimpulan	46
7.2 Saran	46
DAFTAR REFERENSI	47
LAMPIRAN A DATA LATIH	48
LAMPIRAN B DATA UJI	57
LAMPIRAN C POHON KEPUTUSAN	62



DAFTAR TABEL

Tabel 3.1 Fitur pada Data yang Dikumpulkan.....	10
Tabel 4.1 Data Training Perhitungan Manual	21
Tabel 4.2 Hasil Evaluasi Individu dengan Data Training.....	22
Tabel 4.3 Rancangan Pengujian dengan Jumlah Populasi	24
Tabel 4.4 Rancangan Pengujian dengan Nilai Bobot	24
Tabel 4.5 Rancangan Pengujian dengan Jumlah Iterasi.....	25
Tabel 4.6 Rancangan Pengujian dengan nilai Mr dan Cr.....	25
Tabel 4.7 Rancangan Pengujian Berdasarkan Data Latih.....	26
Tabel 5.1 Spesifikasi Perangkat Keras.....	27
Tabel 5.2 Spesifikasi Perangkat Lunak	27
Tabel 6.1 Pengujian dengan Jumlah Populasi	34
Tabel 6.2 Pengujian dengan Nilai Bobot.....	36
Tabel 6.3 Pengujian dengan Jumlah Iterasi.....	37
Tabel 6.4 Pengujian dengan Nilai Mr dan Cr.....	38
Tabel 6.5 Pengujian Berdasarkan Data Latih	40
Tabel 6.6 Hasil Pengujian Validasi.....	40



DAFTAR GAMBAR

Gambar 2.1 Pohon Keputusan	5
Gambar 2.2 Individu sebelum dilakukan <i>crossover</i>	6
Gambar 2.3 Individu sesudah dilakukan <i>crossover</i>	7
Gambar 2.4 Individu sebelum dilakukan mutasi	7
Gambar 2.5 Individu sesudah dilakukan mutasi	8
Gambar 3.1 Model perancangan	11
Gambar 4.1 Diagram Alir Metode <i>Genetic Programming</i>	13
Gambar 4.2 Diagram Alir Metode Inisialisasi	14
Gambar 4.3 Diagram Alir Metode <i>Full Tree</i>	15
Gambar 4.4 Diagram Alir Metode <i>Grow Tree</i>	16
Gambar 4.5 Diagram Alir Metode Mutasi	17
Gambar 4.6 Diagram Alir Metode Reproduksi	18
Gambar 4.7 Diagram Alir Metode <i>crossover</i>	19
Gambar 4.8 Contoh Individu	20
Gambar 6.1 Grafik Hasil Pengujian Jumlah Populasi	35
Gambar 6.2 Grafik Hasil Pengujian Nilai Bobot Perhitungan <i>Fitness</i>	36
Gambar 6.3 Grafik Hasil Pengujian Jumlah Iterasi	38
Gambar 6.4 Grafik Hasil Pengujian Nilai Mr dan Cr	39
Gambar 6.5 Grafik Jumlah <i>Node</i> Pohon	42
Gambar 6.6 Grafik Akurasi Terhadap Data Latih	43
Gambar 6.7 Grafik Perubahan Nilai <i>Fitness</i> Setiap Generasi	43
Gambar 6.8 Grafik Perubahan Nilai <i>Fitness</i> Generasi Awal	44
Gambar 6.9 <i>Crossover</i> pada <i>Node</i> Terminal	44

DAFTAR SOURCE CODE

Source Code 5.1 Inisialisasi	28
Source Code 5.2 Pembentukan Individu.....	28
Source Code 5.3 Pembentukan Pohon	29
Source Code 5.4 Perhitungan <i>Fitness</i>	30
Source Code 5.5 Iterasi	31
Source Code 5.6 Mutasi	32
Source Code 5.7 Reproduksi	32
Source Code 5.8 Crossover	33



DAFTAR LAMPIRAN

LAMPIRAN A DATA LATIH..... 48

LAMPIRAN B DATA UJI..... 57

LAMPIRAN C POHON KEPUTUSAN..... 62



BAB 1 PENDAHULUAN

1.1 Latar Belakang

Diabetes melitus merupakan penyakit yang disebabkan oleh pankreas tidak dapat menghasilkan insulin dalam jumlah yang dibutuhkan oleh tubuh atau ketika tubuh tidak dapat menggunakan insulin dengan efektif (WHO, 2018). Menurut WHO (World Health Organization), 8,5% orang dewasa menderita penyakit ini. Penderita diabetes semakin meningkat terutama di negara yang masyarakatnya memiliki penghasilan menengah ke bawah. Pada tahun 2016, sekitar 1,6 juta orang meninggal akibat penyakit ini. Diabetes berada di peringkat ketujuh sebagai penyakit yang paling mematikan pada tahun 2016. Penyakit ini juga dapat menyebabkan masalah kesehatan lain seperti kebutaan, gagal ginjal, serangan jantung, stroke, dan amputasi pada bagian kaki. Karena penyakit ini dapat mematikan, dibutuhkan metode untuk dapat menentukan penderita diabetes. Metode tersebut diharapkan dapat membantu penanganan penyakit secara dini. Metode yang diusulkan oleh penelitian ini adalah dengan *genetic programming*.

Penelitian dengan metode *genetic programming* yang telah dilakukan adalah penelitian yang dilakukan oleh Tür dan Güvenir mengenai induksi pohon keputusan dengan *genetic programming*. Pohon keputusan tersebut digunakan untuk melakukan klasifikasi afiliasi partai politik anggota kongres Amerika Serikat berdasarkan pilihan anggota tersebut mengenai kebijakan yang perlu diambil oleh kongres untuk masalah penting di Amerika Serikat (Tür & Güvenir, 1996).

Diperlukan metode untuk dapat membuat pohon keputusan berdasarkan data yang dimiliki. Salah satu metode yang dapat digunakan adalah metode *genetic programming*. Metode tersebut dapat digunakan karena *genetic programming* merupakan teknik *heuristic* yang sangat fleksibel sehingga dapat digunakan untuk berbagai jenis representasi permasalahan termasuk pohon keputusan (Espejo et al., 2010). Proses pembentukan pohon keputusan menggunakan algoritme *genetic programming* dilakukan seperti proses algoritme evolusi pada umumnya. Algoritme ini dimulai dengan melakukan pembentukan populasi berisi individu yang merupakan kandidat solusi permasalahan. Kemudian dilakukan proses pembentukan populasi baru dari populasi lama. Metode yang digunakan untuk pembentukan populasi baru adalah dengan cara *crossover* dan mutasi (Estrada-Gil et al., 2007). Dari populasi lama dan populasi baru akan dipilih individu mana saja yang akan digunakan untuk generasi berikutnya. Proses seleksi akan berdasarkan nilai *fitness* dari tiap individu. Nilai *fitness* suatu individu didapatkan dengan menghitung kecocokan suatu individu dalam menyelesaikan suatu permasalahan. Individu dengan nilai *fitness* yang baik akan berpeluang lebih besar untuk dipilih menjadi individu pada generasi berikutnya (Tür & Güvenir, 1996). Proses pembentukan populasi dan seleksi akan dilakukan berulang-ulang hingga batas iterasi telah dicapai. Hasil dari algoritme ini adalah pohon keputusan yang digunakan untuk menentukan

keputusan sesuai dengan permasalahan yang ada. Diharapkan pohon keputusan yang dihasilkan dengan metode *genetic programming* akan memiliki nilai akurasi yang memuaskan.

1.2 Rumusan Masalah

1. Bagaimana implementasi metode *genetic programming* untuk menghasilkan pohon keputusan yang dapat menentukan penderita diabetes melitus?
2. Apa saja parameter metode *genetic programming* yang dapat menghasilkan individu dengan nilai *fitness* yang maksimal?
3. Bagaimana tingkat akurasi saat menentukan penderita diabetes melitus menggunakan pohon keputusan yang dihasilkan oleh metode *genetic programming*?

1.3 Tujuan

1. Melakukan implementasi metode *genetic programming* untuk mendapatkan pohon keputusan yang dapat menentukan penderita diabetes melitus.
2. Mendapatkan parameter metode *genetic programming* yang dapat menghasilkan individu dengan nilai *fitness* yang maksimal.
3. Menguji tingkat akurasi saat menentukan penderita diabetes melitus menggunakan pohon keputusan yang dihasilkan oleh metode *genetic programming*.

1.4 Manfaat

Manfaat penelitian ini bagi penulis adalah untuk dapat menerapkan materi yang sudah didapat selama mengikuti perkuliahan terutama dalam bidang algoritme evolusi. Manfaat lain dari penelitian ini adalah dapat membantu dalam melakukan diagnosis terhadap penyakit diabetes melitus.

1.5 Batasan Masalah

Batasan masalah dari penelitian ini ditentukan sebagai berikut:

1. Hasil keluaran dari metode *genetic programming* adalah pohon keputusan yang menentukan apakah pasien menderita diabetes melitus atau tidak.
2. Data yang digunakan sebagai data latih dan data uji adalah data pasien wanita Indian Pima yang terdiri dari penderita diabetes dan yang bukan penderita diabetes.

1.6 Sistematika Pembahasan

Skripsi ini akan dibagi menjadi beberapa bagian:

BAB I PENDAHULUAN

Bab ini terdiri dari latar belakang yang mendasari penelitian yang akan dilakukan.

Dari latar belakang tersebut dilakukan identifikasi masalah sehingga didapatkan rumusan masalah yang perlu dipecahkan oleh penelitian ini. Kemudian bab ini juga berisi tujuan, batasan masalah, manfaat, dan sistematika pembahasan.

BAB II LANDASAN KEPUSTAKAAN

Bab ini berisi teori yang nantinya akan dijadikan dasar dalam melakukan penelitian. Bab ini juga membahas mengenai penelitian sebelumnya yang memiliki topik atau metode yang berkaitan dengan penelitian ini.

BAB III METODOLOGI PENELITIAN

Bab ini menjabarkan metode penelitian yang dipakai untuk penelitian: studi literatur, analisis kebutuhan, pengumpulan data, perancangan sistem, implementasi, dan analisis hasil penelitian.

BAB IV PERANCANGAN

Bab ini terdiri atas tahapan penyelesaian masalah dan perhitungan. Selain itu, bab ini mencakup perancangan pengujian.

BAB V IMPLEMENTASI

Bab ini membahas tentang implementasi dari rancangan pada bab sebelumnya. Perangkat keras dan lunak yang digunakan saat implementasi juga dijelaskan pada bab ini.

BAB VI PENGUJIAN DAN ANALISIS

Bab ini berisi pembahasan tentang pengujian yang dilakukan setelah implementasi metode yang digunakan. Bab ini juga terdapat pembahasan tentang analisis dari perhitungan akurasi terhadap keluaran yang dihasilkan.

BAB VII PENUTUP

Bab terakhir mengandung kesimpulan yang didapatkan setelah dilakukannya penelitian. Kemudian dari kesimpulan yang didapat, dihasilkan saran yang diharapkan dapat membuat penelitian berikutnya menjadi lebih baik.

BAB 2 LANDASAN KEPUSTAKAAN

2.1 Kajian Pustaka

Kajian pustaka penelitian ini akan membahas mengenai perbandingan penelitian sebelumnya dan penelitian ini yang akan diusulkan. Penelitian sebelumnya yang dibandingkan adalah penelitian dengan metode yang sama tetapi menggunakan objek penelitian yang berbeda. Perbandingan terhadap penelitian yang sudah ada bertujuan untuk dapat membandingkan kriteria dan objek yang menjadi bahan masukan, metode, dan keluaran yang dihasilkan.

1. Penelitian oleh Tür and Güvenir pada tahun 1996 mengenai induksi pohon keputusan menggunakan *genetic programming*. Pada penelitian, didefinisikan bahwa pohon keputusan terbaik adalah pohon keputusan yang memiliki akurasi maksimum dengan jumlah *node* yang minimum. Nilai akurasi dengan dataset partai anggota kongres Amerika Serikat yang dihasilkan adalah sebesar 97,3% (Tür & Güvenir, 1996).
2. Penelitian oleh Eggermont, Kok, dan Kosters pada 2003 membahas mengenai membatasi ruang pencarian dalam pembentukan pohon keputusan menggunakan *genetic programming*. Kesimpulan dari penelitian menunjukkan bahwa pembatasan ruang pencarian dalam pembentukan pohon keputusan dapat meningkatkan kinerja dari metode ini (Eggermont et al., 2003).
3. Penelitian yang dilakukan pada 2007 oleh Estrada-Gil, Fernández-López, Hernández-Lemus, Silva-Zolezzi, Hidalgo-Miranda, Jiménez-Sánchez, dan Vallejo-Clemente berisi tentang induksi pohon keputusan untuk menemukan interaksi antara faktor genetika untuk mendeteksi penyakit. Hasil yang didapatkan adalah pohon keputusan yang dihasilkan tidak mengalami *overfitting* menggunakan metode *genetic programming* (Estrada-Gil et al., 2007).

Penelitian oleh Espejo, Ventura, dan Herrera pada 2010 melakukan survei terhadap penelitian-penelitian yang telah dilakukan sebelumnya yang menggunakan metode *genetic programming*. Penelitian tersebut dilakukan untuk mendapatkan gambaran yang jelas mengenai keuntungan dan kekurangan yang dimiliki oleh metode ini sehingga penelitian berikutnya dapat menghasilkan hasil yang akurat (Espejo et al., 2010).

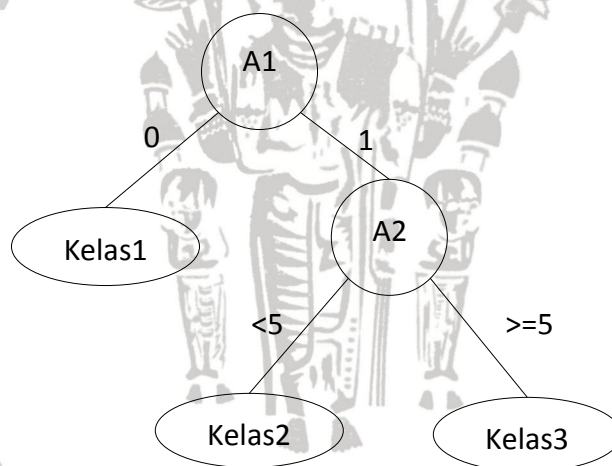
2.2 Diabetes Melitus

Diabetes adalah penyakit yang muncul jika tubuh tidak mampu menghasilkan insulin dalam jumlah yang cukup atau ketika tubuh tidak dapat menggunakan insulin secara efektif (WHO, 2018). Insulin adalah hormon yang diproduksi oleh tubuh untuk mengatur kadar gula. Diabetes akan menyebabkan kadar gula dalam darah lebih tinggi daripada kondisi normal. Kondisi tersebut akan memicu penyakit lain seperti meningkatnya risiko serangan jantung, stroke, amputasi pada bagian kaki, kebutaan dan gagal ginjal.

Pada penelitian ini ada beberapa variabel yang akan digunakan untuk menentukan apakah pasien menderita diabetes. Variabel pertama adalah berapa kali pasien mengandung. Variabel kedua adalah jumlah konsentrasi plasma glukosa pada bagian mulut. Variabel ketiga adalah tekanan darah. Variabel keempat adalah ketebalan kulit pada bagian lengan atas (trisepe) dalam ukuran millimeter. Variabel kelima adalah berapa jumlah insulin dalam tubuh. Variabel keenam adalah BMI (*Body Mass Index*) yang menghitung jumlah massa tubuh sesuai dengan tinggi pasien. Variabel ketujuh adalah nilai dari fungsi *Diabetes Pedigree* yang nilainya didasarkan pada riwayat diabetes pada keluarga pasien. Variabel terakhir menunjukkan usia pasien.

2.3 Pohon Keputusan

Solusi dari algoritme *genetic programming* pada penelitian ini akan menghasilkan solusi berupa pohon keputusan biner. Pohon keputusan merupakan struktur pohon yang tiap *node* memiliki pengujian terhadap atribut dan *leaf* yang merepresentasikan kelas dari data sesuai dengan atribut yang dimiliki (Estrada-Gil et al., 2007). Contoh dari pohon keputusan ditunjukkan oleh Gambar 2.1.



Gambar 2.1 Pohon Keputusan

Gambar 2.1 merupakan salah satu contoh dari pohon keputusan. Pada *node* A1 dilakukan pengujian dengan nilai tipe *boolean*. Jika bernilai *false* maka akan termasuk kategori kelas 1. Jika bernilai A1 bernilai benar, dilakukan pengujian terhadap nilai A2. Untuk nilai A2 kurang dari 5 akan termasuk kelas 2 dan untuk nilai A2 yang lebih dari sama dengan 5 akan termasuk kelas 3.

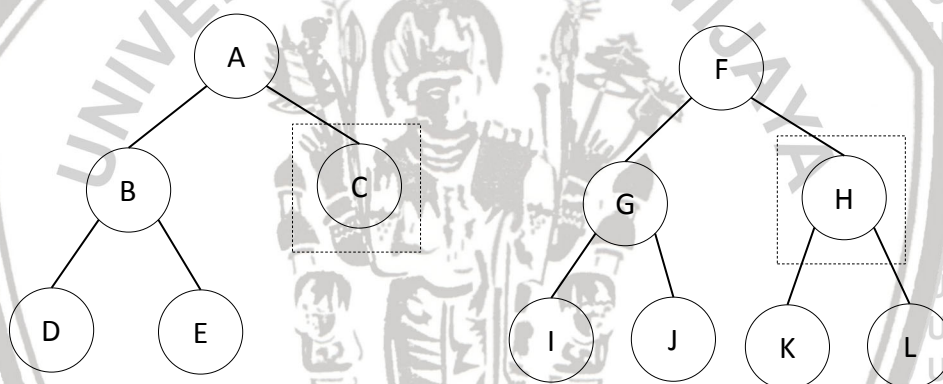
2.4 Genetic Programming

Genetic programming (GP) merupakan metode yang awalnya dikembangkan oleh John Koza dan James Rice dari Stanford University. Metode GP mirip dengan

metode algoritme genetika dimana perbedaannya terletak pada individu algoritme genetika berupa *string* sedangkan untuk GP individu direpresentasikan sebagai *parse tree* dari program (Tür & Güvenir, 1996). Karena memiliki representasi yang fleksibel, GP dapat diaplikasikan ke berbagai jenis masalah seperti klasifikasi, regresi, dan klustering (Espejo et al., 2010). Sama seperti algoritme genetika, GP memiliki proses yang sama yaitu pembentukan populasi awal, reproduksi individu baru, dan seleksi berdasarkan nilai *fitness* individu (Poli et al., 2008). Proses reproduksi dan seleksi akan dilakukan berulang kali hingga tercapai nilai iterasi maksimum.

2.4.1 Inisialisasi

Tahap inisialisasi merupakan tahap awal dimana terjadi pembentukan populasi. Pada populasi tersebut dibangkitkan sejumlah individu yang dibuat secara acak (Estrada-Gil et al., 2010). Jumlah individu dalam populasi ditentukan sesuai dengan kebutuhan.

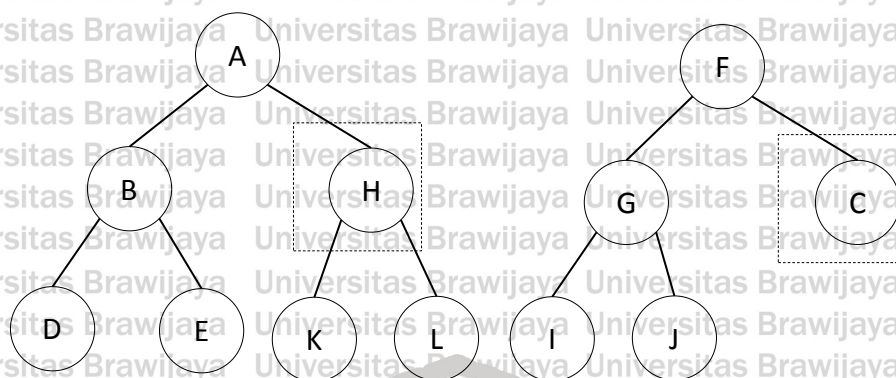


Gambar 2.2 Individu sebelum dilakukan crossover

2.4.2 Crossover

Tahap *crossover* dilakukan dengan memilih dua individu dari populasi yang ada. Individu tersebut dipilih dengan metode *tournament selection* (Poli et al., 2008). Metode tersebut akan memilih induk dengan cara memilih individu-individu secara acak dari populasi dan memilih individu dengan nilai *fitness* tertinggi sebagai induk. Dari dua induk terpilih tersebut akan ditentukan titik rekombinasi secara acak. Bagian pohon yang telah terpilih berdasarkan titik rekombinasi tersebut kemudian akan ditukar satu sama lain (Estrada-Gil et al., 2007). Individu baru akan dihasilkan dari proses penukaran tersebut. Proses *crossover* ditunjukkan oleh Gambar 2.2 dan Gambar 2.3. Titik *crossover* pada gambar tersebut ditunjukkan oleh kotak dengan garis putus-putus.

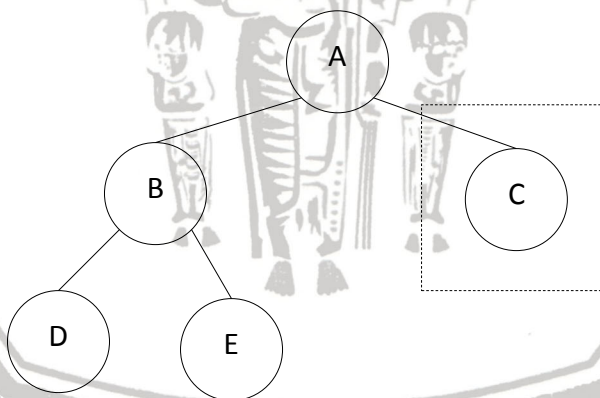




Gambar 2.3 Individu sesudah dilakukan crossover

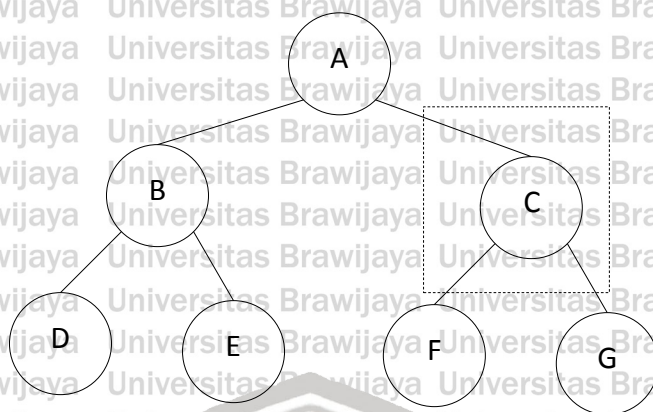
2.4.3 Mutasi

Pada proses mutasi, metode yang digunakan adalah *subtree mutation* yaitu dilakukan proses pemilihan bagian pohon secara acak dari individu terpilih (Poli et al., 2008). Bagian pohon dari individu tersebut kemudian akan diganti dengan bagian pohon baru yang dibangkitkan secara acak (Estrada-Gil et al., 2007). Proses mutasi ditunjukkan oleh Gambar 2.4 dan Gambar 2.5. Titik mutasi pada gambar tersebut ditunjukkan oleh kotak dengan garis putus-putus.



Gambar 2.4 Individu sebelum dilakukan mutasi





Gambar 2.5 Individu sesudah dilakukan mutasi

2.4.4 Seleksi

Dari individu yang terdapat pada populasi awal dan individu yang dihasilkan dari proses *crossover* dan mutasi, akan dilakukan proses seleksi untuk menentukan individu mana yang akan menjadi populasi awal untuk generasi berikutnya. Proses seleksi akan ditentukan oleh fungsi *fitness*. Fungsi *fitness* yang digunakan akan ditentukan oleh akurasi dan ukuran dari pohon keputusan (Tür & Güvenir, 1996).

Nilai *fitness* dapat dicari dengan Persamaan 2.1 (Tür & Güvenir, 1996)

$$fitness = \left(w \times \frac{1}{n+1} \right) + ((1 - w) \times akurasi) \quad (2.1)$$

Nilai *n* merupakan jumlah *node* pada pohon keputusan. Nilai *w* merupakan bobot yang bernilai antara 0 dan 1. Nilai *fitness* yang baik akan memaksimalkan akurasi dan meminimalkan ukuran pohon keputusan.

Setelah nilai *fitness* individu telah ditemukan, proses seleksi yang dilakukan adalah dengan cara probabilitas (Poli et al., 2008). Metode ini akan membuat individu yang memiliki nilai *fitness* yang tinggi akan berpeluang besar untuk masuk ke generasi berikutnya walaupun tidak terjamin akan pasti masuk ke generasi selanjutnya.



BAB 3 METODOLOGI PENELITIAN

3.1 Studi Literatur

Pada penelitian ini studi literatur merupakan tahapan untuk mempelajari literatur dari berbagai sumber mengenai *genetic programming*. Literatur tersebut didapatkan dari buku, jurnal, internet, dan dosen pembimbing. Literatur yang dipelajari mengenai:

- a. Penelitian sebelumnya yang membahas metode *genetic programming* atau penelitian yang memiliki obyek penelitian yang sama.
- b. Teori mengenai *genetic programming*.
- c. Teori mengenai pohon keputusan.

3.2 Analisis Kebutuhan

Analisis kebutuhan pada penelitian ini dilakukan untuk menentukan kebutuhan yang nantinya akan digunakan dalam penelitian ini. Kebutuhan pada implementasi program yang nantinya akan dikembangkan adalah:

1. Kebutuhan perangkat keras dengan spesifikasi minimal:
 - a. Intel Core i5
 - b. RAM 4 GB
 - c. Ruang pada harddisk yang kosong minimal 500 MB
2. Kebutuhan perangkat lunak:
 - a. Sistem operasi Windows 7
 - b. GNU Emacs
 - c. Steel Bank Common Lisp
3. Data yang dibutuhkan:
 - a. Data berupa data pasien penderita diabetes.

3.3 Pengumpulan Data

Pengumpulan data yang digunakan untuk melakukan proses *training* ataupun pengujian merupakan data pasien. Data yang diambil akan menjadi atribut yang nantinya akan direpresentasikan dalam bentuk pohon keputusan. Data yang digunakan adalah data pasien perempuan Indian Pima yang berjumlah 768 orang (Smith et al., 1988). Data tersebut terbagi menjadi 500 orang yang tidak menderita diabetes dan 268 orang yang menderita diabetes. Data yang didapatkan memiliki delapan fitur yang ditunjukkan pada tabel 3.1.

Tabel 3.1 Fitur pada Data yang Dikumpulkan

Fitur	Deskripsi
Pr	Berapa kali pasien mengandung
Gl	Jumlah konsentrasi plasma glukosa pada mulut
Bp	Tekanan darah
St	Ketebalan kulit pada bagian lengan atas (trisepe)
In	Jumlah insulin dalam tubuh
Bm	Jumlah massa tubuh sesuai dengan tinggi pasien
Dp	Nilai dari fungsi <i>Diabetes Pedigree</i> yang nilainya didasarkan pada riwayat diabetes pada keluarga pasien
Ag	Usia pasien

3.4 Model Perancangan

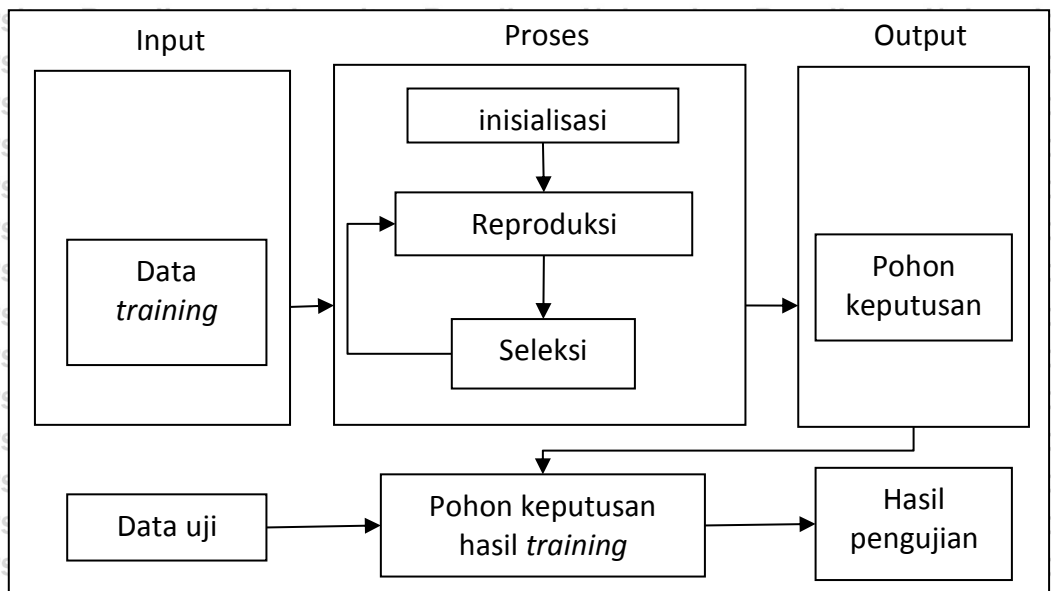
Tahapan perancangan sistem pada penelitian ini untuk memudahkan saat dilakukan proses implementasi dan pengujian. Perancangan sistem akan menghasilkan rancangan langkah kerja dari sistem yang sesuai dengan proses yang harus dilakukan saat melakukan pembangkitan pohon keputusan. Rancangan yang dihasilkan berupa model perancangan sistem yang akan digunakan pada penelitian ini. Diagram model perancangan ditunjukkan oleh Gambar 3.1.

Masukan yang diberikan adalah data *training*. Dari masukan tersebut kemudian dilakukan tahapan sesuai dengan metode *genetic programming*. Hasil keluaran adalah berupa pohon keputusan.

3.5 Implementasi

Tahapan implementasi merupakan tahapan yang dilakukan untuk membangun atau mengimplementasikan sistem sesuai dengan perancangan yang telah dibuat. Selain itu, implementasi yang dilakukan pada tahapan ini adalah implementasi terhadap informasi yang didapat dari tahapan studi literatur. Tahap yang dilakukan pada implementasi:

- Implementasi algoritma *genetic programming* akan ditulis dengan bahasa pemrograman Common Lisp.
- Implementasi sistem ini nantinya akan menghasilkan output berupa pohon keputusan yang memiliki nilai *fitness* paling tinggi.



Gambar 3.1 Model perancangan

3.6 Pengujian

Tahapan pengujian yang nantinya akan dilakukan pada penelitian adalah pengujian akurasi. Pengujian akurasi merupakan tahapan yang dilakukan sebagai pengujian terhadap pohon keputusan yang dihasilkan. Pengujian dilakukan dengan cara membandingkan antara hasil klasifikasi menggunakan pohon keputusan dan data uji yang sebenarnya. Pengujian akurasi dilakukan dengan persamaan 3.1.

$$akurasi = \frac{jumlah\ data\ akurat}{jumlah\ seluruh\ data\ uji} \times 100\% \quad (3.1)$$

Akurasi tersebut didapatkan dengan membandingkan antara berapa jumlah data pasien yang penentuannya tepat menggunakan pohon keputusan dengan jumlah data pasien secara keseluruhan. Dengan cara ini, jumlah akurasi pohon keputusan yang dihasilkan dengan metode *genetic programming* dapat dihasilkan.

3.7 Analisis

Tahapan analisis yang dilakukan ketika penelitian ini merupakan tahapan untuk melakukan analisis terhadap hasil yang didapatkan saat pengujian. Dari hasil tersebut dapat dilakukan analisis mengenai tingkat akurasi metode yang digunakan dan parameter yang berpengaruh terhadap hasil. Semakin tinggi nilai akurasi dari hasil pengujian maka akan didapatkan bahwa metode *genetic programming* menghasilkan nilai yang baik untuk mendapatkan pohon keputusan. Hasil analisis ini kemudian akan digunakan untuk membuat kesimpulan dari penelitian ini.

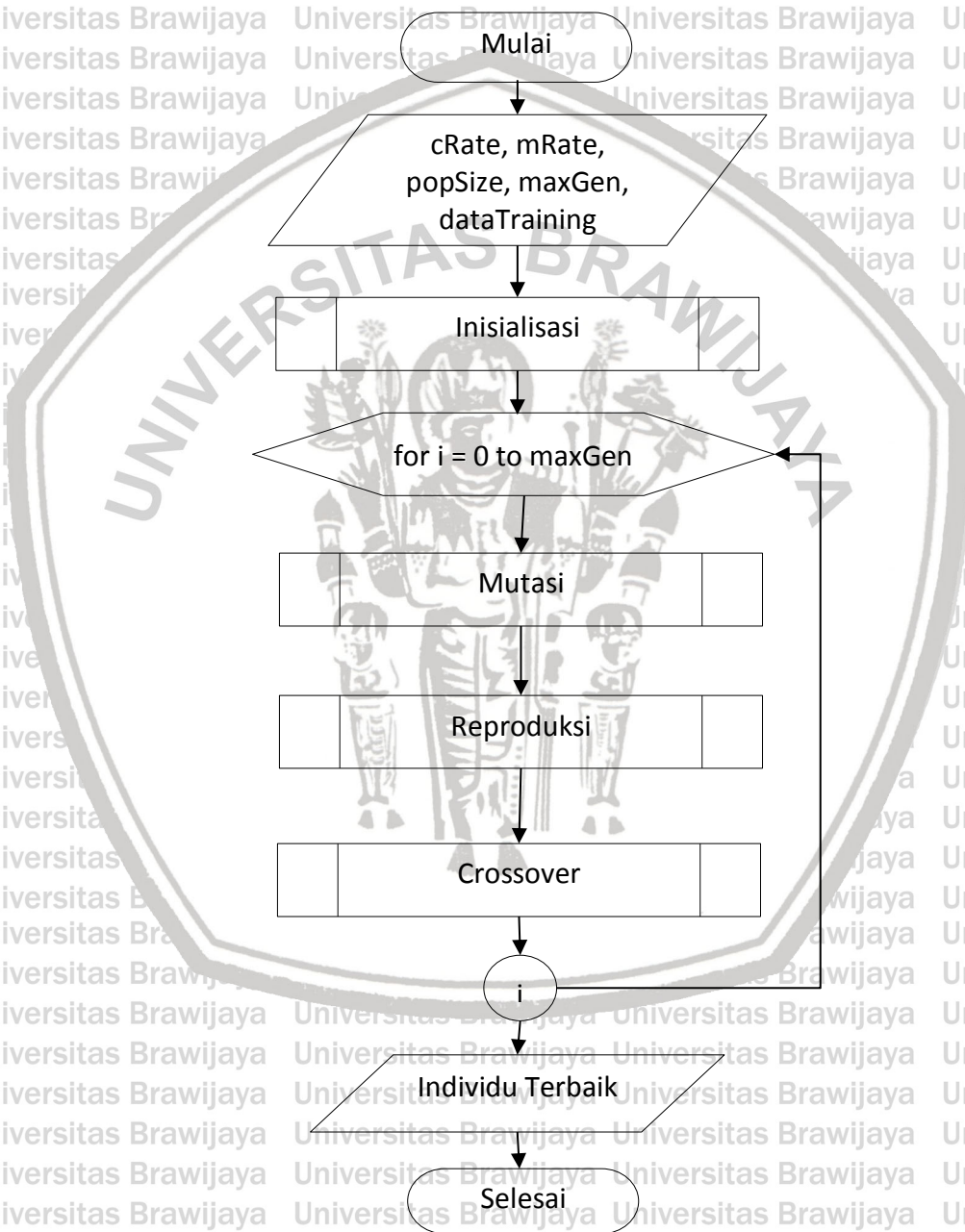
3.8 Kesimpulan

Tahapan kesimpulan pada penelitian ini dilakukan ketika tahapan perancangan, implementasi, dan pengujian metode yang digunakan telah selesai dilakukan. Kesimpulan akan didapatkan dari analisis berdasarkan metode penelitian yang digunakan. Setelah pembuatan kesimpulan dilakukan penulisan saran. Saran bertujuan agar kesalahan yang terjadi dari hasil atau proses penelitian ini dapat diperbaiki. Selain itu, penulisan saran juga bertujuan agar penelitian yang telah dikerjakan ini dapat dikembangkan menjadi lebih baik.



BAB 4 PERANCANGAN

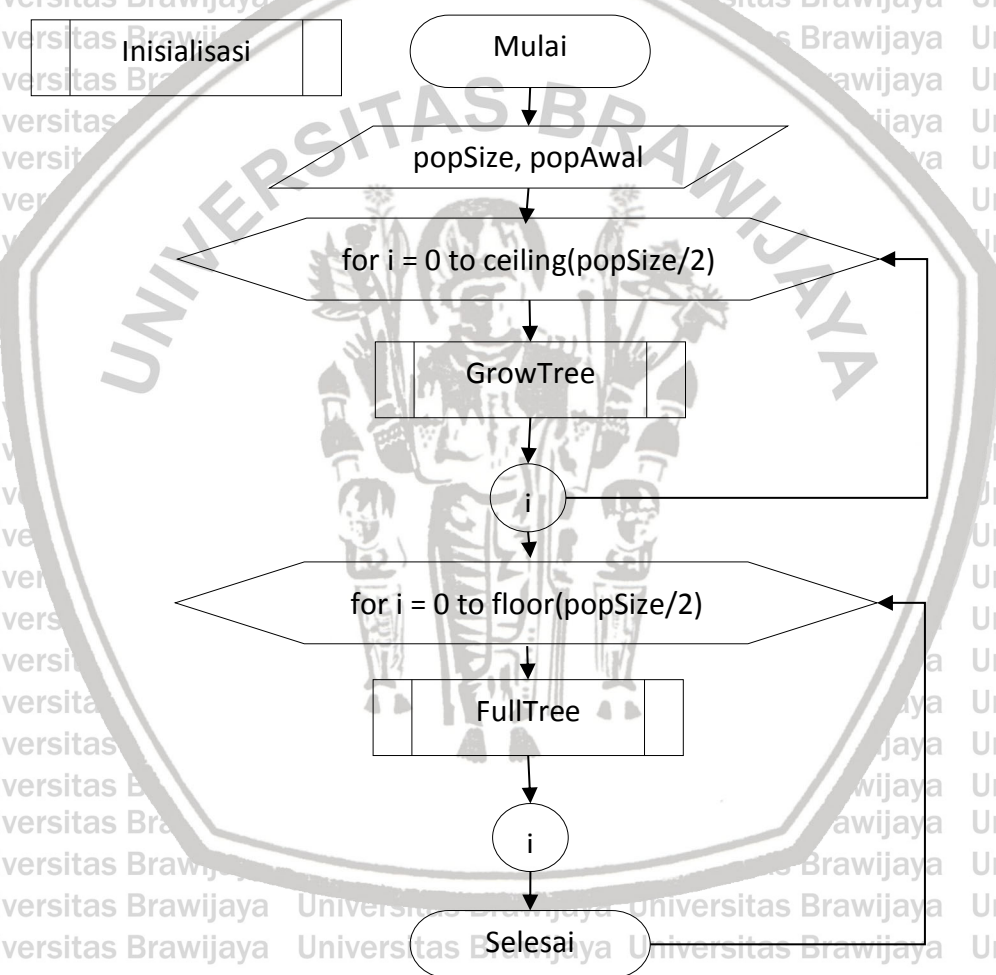
Bab ini berisi penjelasan tentang formulasi permasalahan, tahapan penyelesaian masalah secara sederhana, perancangan antarmuka, dan perancangan pengujian. Pada Gambar 4.1 menunjukkan diagram alir algoritme *genetic programming*.



Gambar 4.1 Diagram Alir Metode *Genetic Programming*

Pada metode *genetic programming* ditentukan terlebih dahulu nilai parameter yang diperlukan. Parameter yang digunakan adalah *cRate* yang menentukan jumlah individu hasil *crossover*, *mRate* yang menentukan individu hasil mutasi, jumlah populasi, jumlah iterasi maksimal, dan data latih. Tahapan pertama yang dilakukan adalah inialisasi yaitu pembentukan populasi awal sebanyak jumlah populasi. Kemudian dilakukan proses pembentukan individu baru dengan proses mutasi, reproduksi, dan *crossover*. Proses tersebut dilakukan sebanyak jumlah iterasi yang telah ditentukan. Individu terbaik yang didapatkan adalah berupa pohon keputusan yang akan digunakan untuk menentukan penderita diabetes.

4.1 Tahapan Penyelesaian Masalah

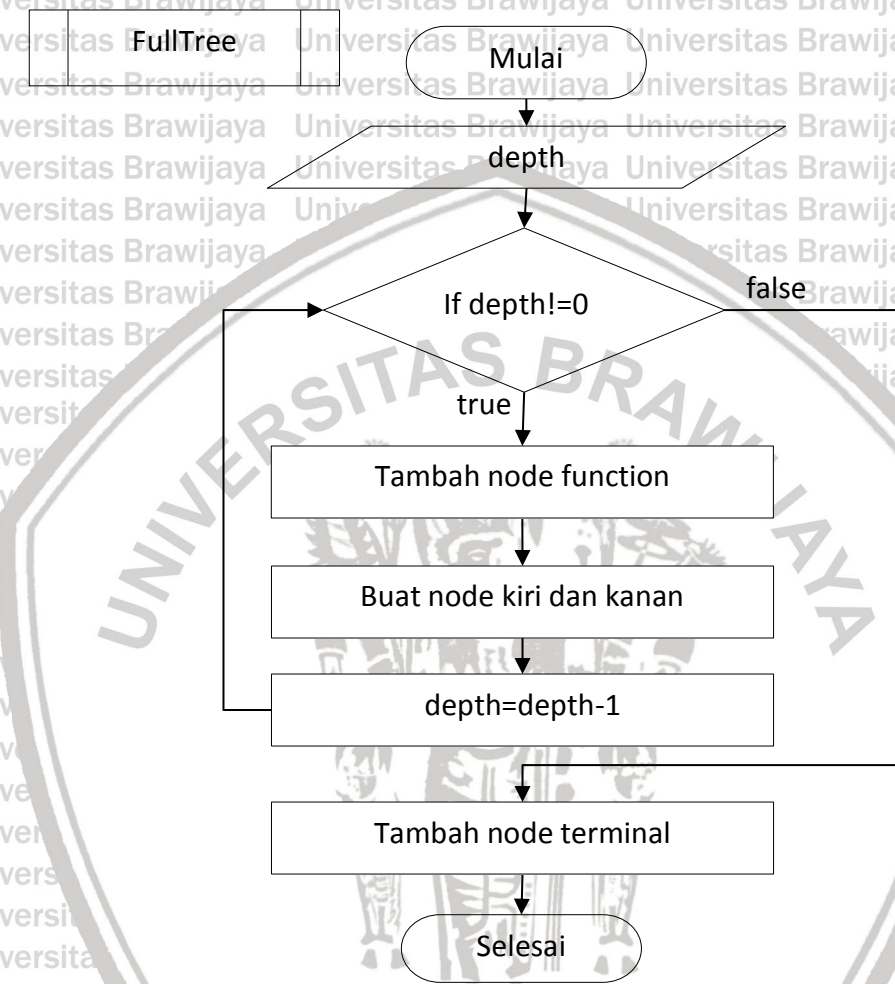


Gambar 4.2 Diagram Alir Metode Inialisasi

4.1.1 Inialisasi

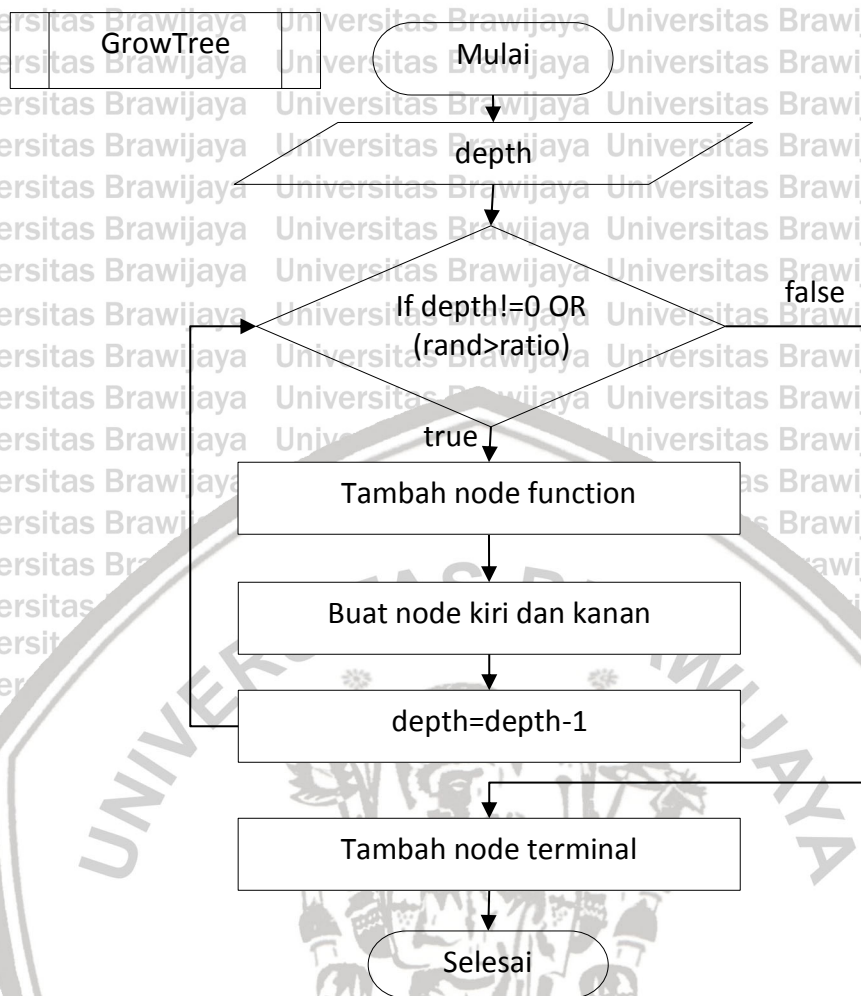
Gambar 4.2 menunjukkan diagram alir proses inialisasi populasi awal. Tahapan ini dilakukan dengan membuat individu sebanyak ukuran populasi

dengan metode *ramped half-and-half*. Metode *ramped half-and-half* akan membentuk individu dengan metode *full* dan *grow* dengan perbandingan jumlah 1:1. Setiap individu yang telah dibentuk dengan metode *full* atau *grow* akan dimasukkan kedalam populasi awal



Gambar 4.3 Diagram Alir Metode *Full Tree*

Metode *full* yang ditunjukkan oleh gambar 4.3 membuat individu dengan tiap *node* diisi dengan anggota dari himpunan fungsi hingga kedalaman maksimal tercapai. Langkah pertama yang dilakukan adalah menentukan nilai kedalaman untuk pohon individu. *Node* akan diisi dengan fungsi jika nilai *depth* belum maksimal. Setelah *node* diisi dengan fungsi, bagian kiri dan kanan dari *node* tersebut akan diisi dengan *node* baru sesuai dengan metode *full* secara rekursif. Jika kedalaman *node* sudah maksimal, *node* tersebut diisi dengan *terminal*. Setelah seluruh cabang pohon diisi dengan penuh sesuai dengan kedalamannya, individu baru telah terbentuk.



Gambar 4.4 Diagram Alir Metode *Grow Tree*

Pada metode *grow* yang ditunjukkan oleh gambar 4.4, tiap *node* dapat diisi oleh anggota dari himpunan fungsi dan himpunan *terminal*. Setelah menentukan kedalaman maksimal dari pohon, *node* baru akan diisi dengan fungsi atau *terminal*. Jika kedalaman maksimal belum tercapai, *node* tersebut akan diisi secara acak berdasarkan perbandingan rasio antara jumlah anggota himpunan *terminal* dengan jumlah anggota himpunan *terminal* dan fungsi. Jika *node* tersebut diisi dengan fungsi, bagian kiri dan kanan *node* akan diisi dengan metode *grow* secara rekursif. Jika *node* diisi dengan anggota *terminal*, *node* tersebut dianggap sudah mencapai kedalaman maksimal. Ketika nilai kedalaman maksimal telah tercapai, *node* akan diisi dengan *terminal*. Setelah pohon selesai dibuat, individu yang telah terbentuk akan menjadi anggota populasi awal.

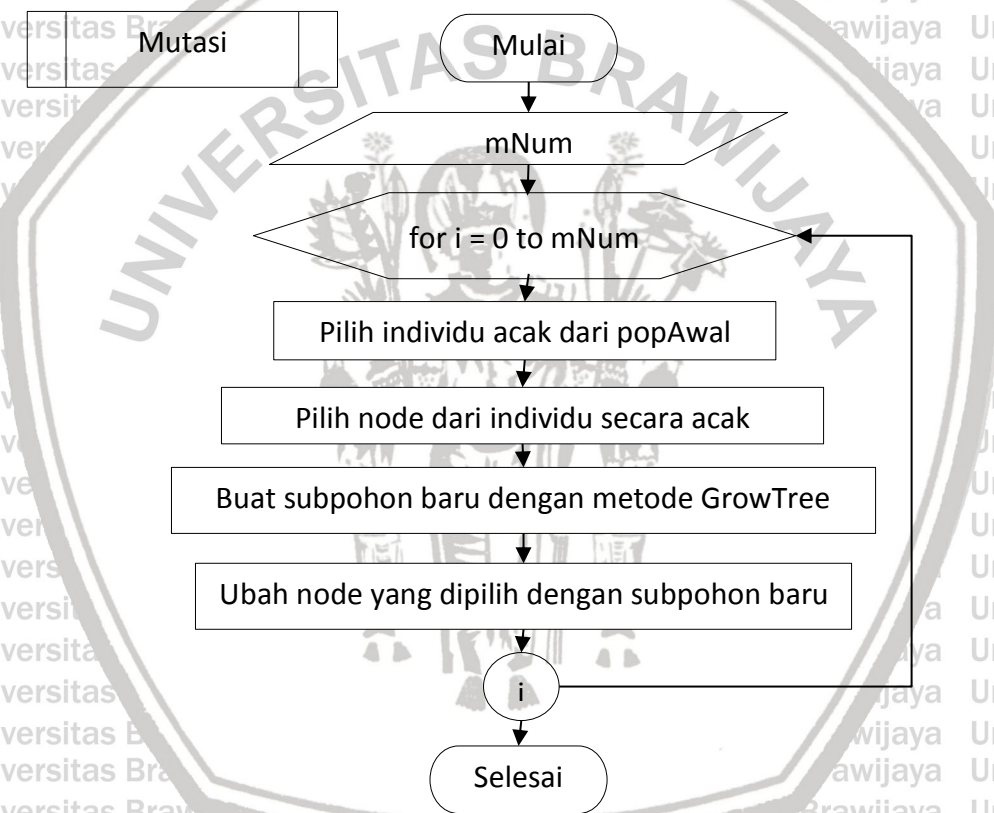
4.1.2 Mutasi

Gambar 4.5 menunjukkan diagram alir proses mutasi yang dilakukan pada induk terpilih. Nilai *mNum* menentukan berapa kali proses mutasi dilakukan. Tahapan ini dilakukan dengan melakukan perubahan pada titik mutasi dengan

menanamkan subpohon baru kepada pohon induk. Dari induk mutasi akan dipilih *node* secara acak. *Node* tersebut akan menjadi *node* induk. Kemudian dilakukan proses *grow tree* untuk membuat subpohon baru. *Node* yang dipilih secara acak itu kemudian akan dihapus dan diganti dengan subpohon baru yang telah dibentuk. Individu baru hasil mutasi selesai terbentuk setelah perubahan *node* lama dengan subpohon baru.

4.1.3 Reproduksi

Gambar 4.6 menunjukkan diagram alir proses reproduksi. Pada tahap reproduksi dilakukan pemilihan individu dari populasi lama berdasarkan nilai *fitness*. Individu yang terpilih merupakan individu dengan *fitness* terbesar pada populasi lama. Jumlah individu yang dipilih akan sesuai dengan nilai *rNum*.

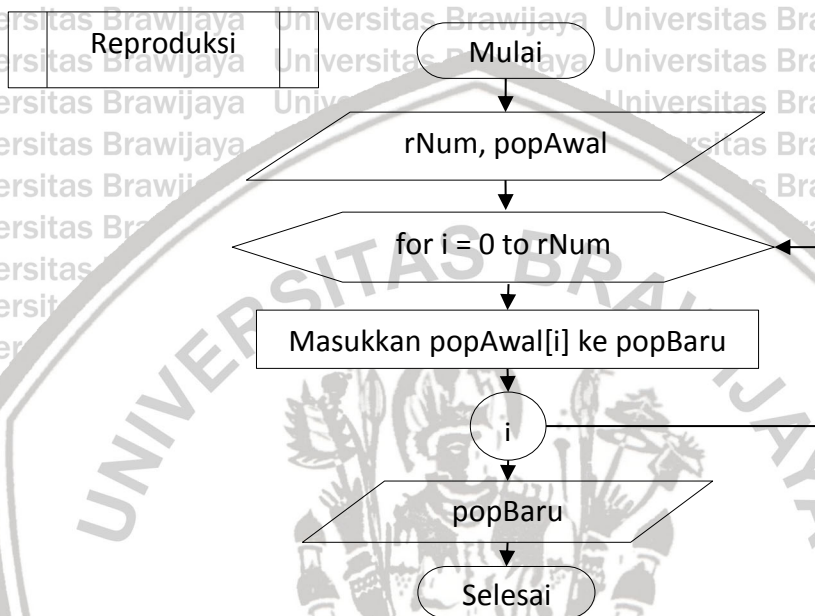


Gambar 4.5 Diagram Alir Metode Mutasi

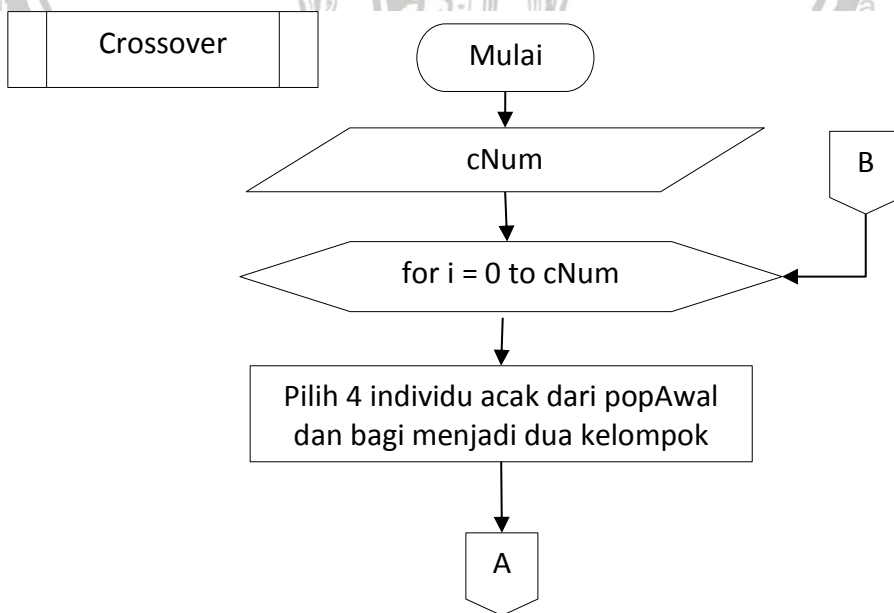
4.1.4 Crossover

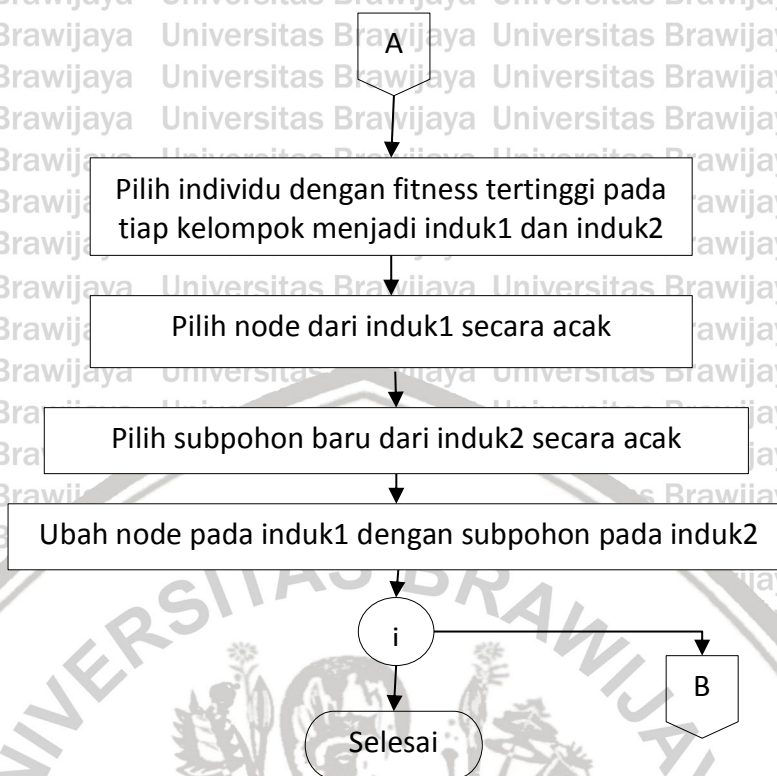
Gambar 4.7 menunjukkan diagram alir *crossover*. Proses *crossover* dilakukan dengan metode *binary tournament*. *Crossover* dilakukan dengan membuat individu atau individu baru sesuai dengan nilai *cNum*. Pertama-tama dilakukan pemilihan empat individu secara acak. Empat inidividu kemudian dibagi menjadi dua kelompok. Dari tiap-tiap kelompok dipilih individu dengan nilai *fitness* tertinggi. Kedua individu tersebut akan menjadi induk pada proses *crossover*.

Operasi *crossover* dilakukan pada dua individu induk dengan cara menukarkan masing-masing subpohonnya. Pada induk pertama dipilih *node* secara acak. *Node* tersebut kemudian akan dihapus. Proses berikutnya adalah memilih *node* pada induk kedua secara acak. *Node* pada individu kedua yang telah terpilih tersebut akan menjadi subpohon yang menggantikan *node* pada individu pertama yang telah dihapus. Individu baru telah selesai dibentuk dan diletakkan pada populasi baru.



Gambar 4.6 Diagram Alir Metode Reproduksi





Gambar 4.7 Diagram Alir Metode *crossover*

4.2 Perhitungan Manual

4.2.1 Skenario Perhitungan Manual

Diketahui populasi awal berisi 4 individu dengan kedalaman maksimal 2-3 saat proses inialisasi. Nilai *mutation rate* = 1%, *reproduction rate* = 9%, dan *crossover rate* = 90%. Data *training* yang digunakan pada skenario ini berjumlah 4 data dengan perbandingan antara penderita diabetes dan bukan penderita diabetes sejumlah 1:1.

4.2.2 Representasi Individu

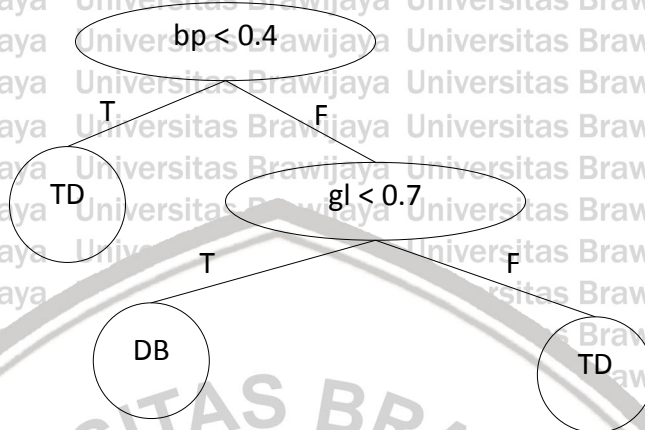
Pada skenario ini pohon akan direpresentasikan dengan *symbolic expression* (*sexps*). Contoh representasi pohon adalah sebagai berikut.

$(((< \text{bp } 0,4) \text{ TD } ((< \text{gl } 0,7) \text{ DB } \text{TD}))$

Representasi tersebut sama dengan individu pada gambar 4.8.

Notasi *sexps* pada representasi individu didefinisikan memiliki dua anggota yaitu atom dan *sexps*. Atom pada ekspresi merupakan anggota dari himpunan *terminal* yaitu DB (menderita diabetes) dan TD (tidak menderita diabetes). *Sexps* didefinisikan sebagai *list* dengan *syntax* yang ditutup dengan tanda kurung. *List* pada representasi memiliki tiga anggota. Anggota pertama menunjukkan fungsi

dari individu yang menggunakan notasi *prefix*. Anggota kedua merupakan hasil ketika evaluasi fungsi menghasilkan nilai benar. Sedangkan anggota ketiga adalah hasil yang didapatkan jika evaluasi fungsi menghasilkan nilai salah. Anggota kedua dan ketiga dapat diisi dengan atom atau *sexps*.



Gambar 4.8 Contoh Individu

4.2.3 Inisialisasi Populasi Awal

Proses inisialisasi dilakukan sebanyak empat kali sesuai dengan jumlah populasi awal. Inisialisasi populasi awal dilakukan secara *ramped-half-half* dengan perbandingan individu *full* dan *grow* adalah 1:1. Nilai konstan pada *node* fungsi akan diberi nilai antara 0 hingga 1. Berikut hasil inisialisasi populasi awal.

- P1: ((< bp 0,4) TD ((< gl 0,7) DB TD))
- P2: ((< gl 0,5) DB ((< dp 0,5) TD ((< bp 0,8) TD DB)))
- P3: ((< pr 0,5) ((< ag 0,4) TD DB) ((< bm 0,2) DB TD))
- P4: ((< st 0,1) ((< dp 0,7) TD DB) ((< in 0,7) TD DB))

Individu P1 dan P2 dibentuk dengan metode *grow* dimana P1 memiliki kedalaman maksimal 2 dan P2 memiliki kedalaman maksimal 3. Individu P3 dan P4 memiliki kedalaman 2 dan dibentuk dengan metode *full*.

4.2.4 Mutasi

Individu yang akan dilakukan proses mutasi dipilih secara acak dari populasi awal. Jumlah individu yang dihasilkan dari proses mutasi didapatkan dari jumlah populasi awal dan nilai *mutation rate*. Perhitungan jumlah individu dilakukan dengan cara sebagai berikut.

$$M = [4 \times 0,01] = [0,04] = 1$$

Dari populasi awal dipilih satu individu secara acak. Pada skenario ini, individu yang terpilih adalah P1. Pada individu tersebut, dipilih satu *node* secara acak sebagai titik mutasi. *Node* yang menjadi titik mutasi pada P1 akan digarisbawahi pada *sexps* berikut.



P1: ((< bp 0,4) TD ((< gl 0,7) DB TD))

Setelah *node* dipilih, *subtree* baru perlu dibentuk dengan metode *grow*.

Subtree baru yang terbentuk adalah:

S: ((< st 0,9) ((< in 0,9) DB TD) DB)

Subtree yang telah terbentuk akan menggantikan *node* pada individu P1.

Individu baru yang dihasilkan:

M: ((< bp 0,4) ((< st 0,9) ((< in 0,9) DB TD) DB) ((< gl 0,7) DB TD))

4.2.5 Reproduksi

Jumlah individu yang dihasilkan dari proses reproduksi didapat dari jumlah populasi awal dan nilai *reproduction rate*. Jumlah individu yang dihasilkan didapat dengan cara:

$$R = [4 \times 0,09] = [0,36] = 1$$

Sesuai dengan nilai R, satu individu yang dipilih adalah individu dengan nilai *fitness* tertinggi pada populasi awal. Nilai *fitness* dihitung dengan persamaan 2.1.

Data *training* yang digunakan untuk menghitung nilai akurasi didapat dari tabel 4.1. Nilai *w* pada skenario ini ditentukan sebesar 0,2.

Tabel 4.1 Data Training Perhitungan Manual

No	Pr	gl	bp	st	in	bm	dp	ag	Diabetes?
1	1	89	66	23	94	28.1	0.167	21	Tidak
2	3	126	88	41	235	39.3	0.704	27	Tidak
3	7	160	54	32	175	30.5	0.588	39	Ya
4	10	125	70	26	115	31.1	0.205	41	Ya

Perhitungan akurasi menggunakan P1 pada data pertama adalah dengan melakukan evaluasi pada fungsi pertama yaitu (< bp 0.4). Nilai bp pada data pertama dilakukan proses normalisasi:

$$bp = \frac{66 - 54}{88 - 54} = 0,35$$

Dari *node* fungsi dievaluasi bahwa 0,35 < 0,4 bernilai benar. *Node* kemudian dievaluasi adalah anggota *list* kedua yang merupakan atom yaitu TD. Hasil dari individu adalah bukan penderita diabetes dan cocok dengan nilai pada data pertama. Individu P1 akurat jika dibandingkan dengan data pertama. Hasil individu dengan data *training* ditunjukkan pada tabel 4.2.

Dari hasil evaluasi tersebut, nilai akurasi tiap individu dapat dihitung dengan cara sebagai berikut.

$$akurasi P1 = \frac{2}{4} = 0,5$$



$$\text{akurasi } P2 = \frac{0}{4} = 0$$

$$\text{akurasi } P3 = \frac{2}{4} = 0,5$$

$$\text{akurasi } P4 = \frac{1}{4} = 0,25$$

Tabel 4.2 Hasil Evaluasi Individu dengan Data Training

Data \ Individu	P1	P2	P3	P4
1	TD	DB	TD	TD
2	TD	DB	TD	DB
3	TD	TD	TD	TD
4	TD	TD	TD	TD

Nilai *fitness* dapat diketahui setelah nilai akurasi didapatkan. Perhitungan nilai *fitness* masing-masing individu:

$$\text{fitness } P1 = \left(0,2 \times \frac{1}{5+1}\right) + ((1 - 0,2) \times 0,5) = 0,033 + 0,4 = 0,433$$

$$\text{fitness } P2 = \left(0,2 \times \frac{1}{7+1}\right) + ((1 - 0,2) \times 0) = 0,025 + 0 = 0,025$$

$$\text{fitness } P3 = \left(0,2 \times \frac{1}{7+1}\right) + ((1 - 0,2) \times 0,5) = 0,025 + 0,4 = 0,425$$

$$\text{fitness } P4 = \left(0,2 \times \frac{1}{7+1}\right) + ((1 - 0,2) \times 0,25) = 0,025 + 0,2 = 0,225$$

Dari perhitungan nilai *fitness*, individu yang dipilih dalam proses reproduksi adalah individu P1.

R: ((< bp 0,4) TD ((< gl 0,7) DB TD))

4.2.6 Crossover

Jumlah individu yang dihasilkan dari proses *crossover* berasal dari jumlah individu yang dihasilkan oleh proses sebelumnya. Proses reproduksi dan mutasi masing-masing menghasilkan satu individu baru. Dengan jumlah populasi awal empat, individu yang dihasilkan oleh *crossover* adalah dua individu. Induk akan dipilih dengan metode *binary tournament*.

Kandidat induk pertama: P1 dan P3 (*fitness* P1>P3)

Kandidat induk kedua: P2 dan P4 (*fitness* P4>P2)

Induk *crossover*: P1 dan P4

Pada P1 akan dipilih *node* secara acak yang akan digantikan oleh *node* pada P4 yang juga dipilih secara acak. *Node* yang dipilih dan hasil *crossover*:

P1: ((\leq bp 0,4) TD ((\leq gl 0,7) DB TD))

P4: ((\leq st 0,1) ((\leq dp 0,7) TD DB) (\leq in 0,7) TD DB))

Hasil (C1): ((\leq bp 0,4) ((\leq in 0,7) TD DB) ((\leq gl 0,7) DB TD))

Kemudian proses yang sama juga dilakukan untuk mendapat individu hasil *crossover* kedua.

Kandidat induk pertama: P2 dan P3 (*fitness* P3>P2)

Kandidat induk kedua: P1 dan P4 (*fitness* P1>P4)

Induk *crossover*: P2 dan P1

P2: ((\leq gl 0,5) DB ((\leq dp 0,5) TD ((\leq bp 0,8) TD DB))

P1: ((\leq bp 0,4) TD (\leq gl 0,7) DB TD))

Hasil (C2): ((\leq gl 0,5) DB ((\leq dp 0,5) TD ((\leq bp 0,8) ((\leq gl 0,7) DB TD) DB))

Populasi baru didapatkan dari individu-individu yang dihasilkan oleh proses mutasi, reproduksi, dan *crossover*. Populasi baru yang dihasilkan:

M: ((\leq bp 0,4) ((\leq st 0,9) ((\leq in 0,9) DB TD) DB) ((\leq gl 0,7) DB TD))

R: ((\leq bp 0,4) TD ((\leq gl 0,7) DB TD))

C1: ((\leq bp 0,4) ((\leq in 0,7) TD DB) ((\leq gl 0,7) DB TD))

C2: ((\leq gl 0,5) DB ((\leq dp 0,5) TD ((\leq bp 0,8) ((\leq gl 0,7) DB TD) DB))

4.3 Perancangan Pengujian

Pengujian pada penelitian ini dilakukan dengan beberapa skenario pengujian. Masing-masing skenario pengujian tersebut digunakan untuk mendapat solusi penyelesaian masalah yang optimal. Pengujian pada setiap skenario dilakukan sebanyak sepuluh kali. Dari pengujian tersebut, nilai rata-rata *fitness* akan digunakan sebagai acuan untuk mendapatkan individu yang paling optimal.

4.3.1 Perancangan Pengujian Berdasarkan Jumlah Populasi

Pengujian pada jumlah populasi dilakukan untuk mengetahui pengaruh jumlah populasi terhadap nilai *fitness* yang didapat. Jumlah populasi yang digunakan memiliki kelipatan ratusan, dari 100 hingga 500. Pengujian yang dilakukan ditunjukkan pada Tabel 4.3.

Tabel 4.3 Rancangan Pengujian dengan Jumlah Populasi

Jumlah Populasi	Nilai <i>fitness</i> pengujian ke-i										Rata-rata nilai <i>fitness</i>
	1	2	3	4	5	6	7	8	9	10	
100											
200											
300											
400											
500											
600											
700											
800											
900											
1000											
5000											
10000											

4.3.2 Perancangan Pengujian Berdasarkan Nilai Bobot pada *Fitness*

Pengujian ini dilakukan untuk mengetahui pengaruh nilai bobot pada perhitungan *fitness* terhadap nilai *fitness* yang didapat. Nilai bobot yang digunakan merupakan bilangan riil antara 0 dan 1. Pengujian yang dilakukan ditunjukkan pada Tabel 4.4.

Tabel 4.4 Rancangan Pengujian dengan Nilai Bobot

Nilai bobot	Nilai <i>fitness</i> pengujian ke-i										Rata-rata nilai <i>fitness</i>
	1	2	3	4	5	6	7	8	9	10	
0											
0,25											
0,5											
0,75											

4.3.3 Perancangan Pengujian Berdasarkan Jumlah Iterasi

Pengujian pada jumlah iterasi dilakukan untuk mengetahui pengaruh terhadap nilai *fitness* dari setiap percobaan. Jumlah iterasi yang digunakan



memiliki nilai dari 100 hingga 1000 dengan kelipatan ratusan dan nilai iterasi 5000 dan 10000. Pengujian yang dilakukan ditunjukkan pada tabel 4.5.

Tabel 4.5 Rancangan Pengujian dengan Jumlah Iterasi

Jumlah Iterasi	Nilai <i>fitness</i> pengujian ke-i										Rata-rata nilai <i>fitness</i>
	1	2	3	4	5	6	7	8	9	10	
100											
200											
300											
400											
500											
600											
700											
800											
900											
1000											
5000											
10000											

4.3.4 Perancangan Pengujian Berdasarkan Nilai Mr dan Cr

Pengujian pada nilai *mutation rate* dan *crossover rate* dilakukan untuk mengetahui pengaruh nilai tersebut terhadap nilai *fitness* dari setiap percobaan. Pengujian yang dilakukan ditunjukkan pada Tabel 4.6.

Tabel 4.6 Rancangan Pengujian dengan nilai Mr dan Cr

Nilai Mr; Cr	Nilai <i>fitness</i> pengujian ke-i										Rata-rata nilai <i>fitness</i>
	1	2	3	4	5	6	7	8	9	10	
0,1;0,8											
0,2;0,7											
0,3;0,6											
0,4;0,5											
0,5;0,4											
0,6;0,3											



0,7;0,2										
0,8;0,1										

4.3.5 Perancangan Pengujian Berdasarkan Data Latih

Pengujian berdasarkan data latih dilakukan untuk mengetahui apakah data latih yang seimbang memiliki pengaruh terhadap nilai *fitness* dari setiap percobaan. Data latih yang digunakan merupakan data latih yang memiliki perbandingan kelas yang sama dan berbeda. Data latih yang tidak seimbang memiliki perbandingan antara penderita diabetes dan bukan penderita diabetes sebesar 1:2. Pengujian yang dilakukan ditunjukkan pada Tabel 4.7.

Tabel 4.7 Rancangan Pengujian Berdasarkan Data Latih

Data Latih	Nilai <i>fitness</i> pengujian ke-i										Rata-rata nilai <i>fitness</i>	
	1	2	3	4	5	6	7	8	9	10		
Diabetes: Tidak Diabetes												
178:178												
178:356												

BAB 5 IMPLEMENTASI

5.1 Spesifikasi Sistem

Bagian ini akan menunjukkan sistem yang digunakan pada penelitian ini. Spesifikasi yang digunakan meliputi spesifikasi perangkat keras dan spesifikasi perangkat lunak.

5.1.1 Spesifikasi Perangkat Keras

Spesifikasi perangkat keras pada penelitian ini ditunjukkan pada Tabel 5.1.

Tabel 5.1 Spesifikasi Perangkat Keras

Komponen	Spesifikasi
Processor	Intel® Core™ i5 2450M
RAM	4 GB
Storage	500 GB

5.1.2 Spesifikasi Perangkat Lunak

Spesifikasi perangkat lunak yang digunakan untuk proses implementasi pada penelitian ini ditunjukkan pada Tabel 5.2.

Tabel 5.2 Spesifikasi Perangkat Lunak

Komponen	Spesifikasi
Sistem Operasi	Windows 7 Ultimate 64-bit SP1
Bahasa Pemrograman	Common Lisp
IDE	Portacle

5.2 Implementasi Sistem

Implementasi dilakukan sesuai dengan perancangan pada bab 4. Bahasa pemrograman yang digunakan adalah bahasa Common Lisp. Hasil implementasi akan dijelaskan pada subbab berikut.

5.2.1 Inisialisasi

Inisialisasi dilakukan dengan membuat individu sesuai dengan jumlah populasi. Perhitungan nilai *fitness* dilakukan setiap kali individu dibuat. Populasi

hasil inialisasi kemudian akan diurutkan berdasarkan nilai *fitness* dari yang terbesar hingga yang terkecil. Implementasi inialisasi ditunjukkan oleh Source Code 5.1.

No.	Kode Sumber
1	(defun initialize (n)
2	"Return descending sorted population by fitness with size of n
3	with half-and-ramped method"
4	(let ((n-full (floor (/ n 2)))
5	(n-grow (ceiling (/ n 2)))
6	(new-population (make-empty-population)))
7	(loop repeat n-grow do
8	(push (new-grow-program) new-population))
9	(loop repeat n-full do
10	(push (new-full-program) new-population))
11	(stable-sort new-population #'> :key #'program-fitness))

Source Code 5.1 Inialisasi

Keterangan:

Baris 4-5 : menentukan jumlah individu yang dibuat dengan metode *full* dan *grow*.

Baris 7-8 : membuat individu dengan metode *grow*.

Baris 9-10 : membuat individu dengan metode *full*.

Baris 11 : mengurutkan populasi yang sudah dibuat.

Implementasi pembentukan individu dilakukan dengan membentuk pohon baru, menghitung jumlah *node*, dan nilai *fitness*. Pembentukan individu ditunjukkan oleh Source Code 5.2.

No.	Kode Sumber
1	(defun new-program (tree-fn &optional (low 4) (high 8))
2	"Generate new random program with tree crated by tree-fn"
3	(let ((tree (funcall tree-fn (random-in-range low high))))
4	(create-program tree))
5	
6	(defun new-full-program ()
7	"Generate new random program with full tree"
8	(new-program #'make-full-tree))
9	
10	(defun new-grow-program ()
11	"Generate new random program with grow tree"
12	(new-program #'make-grow-tree))
13	
14	(defun create-program (tree)
15	"Return new program from tree"
16	(make-program :tree tree
17	:fitness (calculate-fitness tree)
18	:vertices (number-of-node tree))

Source Code 5.2 Pembentukan Individu



Keterangan:

Baris 3 : membuat pohon baru sesuai dengan kedalaman acak antara nilai minimal dan maksimal.

Baris 8 : membuat individu dengan metode *full*.

Baris 12 : membuat individu dengan metode *grow*.

Baris 16-18 : membuat individu baru dengan tipe data *struct*.

Baris 17 : menghitung nilai *fitness* dari pohon.

Baris 18 : menghitung jumlah *node* yang dimiliki pohon.

Implementasi pembentukan pohon baru dilakukan dengan metode *full* dan *grow*. Pembentukan pohon baru ditunjukkan oleh Source Code 5.3.

No.	Kode Sumber
1	(defun make-full-tree (depth)
2	"Make new tree using full method with assigned depth"
3	(if (zerop depth)
4	(pick-terminal)
5	(make-tree (pick-function)
6	(make-full-tree (1- depth))
7	(make-full-tree (1- depth))))))
8	
9	(defun make-grow-tree (depth &optional (head-p t))
10	"Make new tree using grow method with assigned depth"
11	;;always pick function if node is head to avoid producing tree
12	with only one node
13	(if (or (zerop depth)
14	(and (not head-p) (terminate-tree-p)))
15	(pick-terminal)
16	(make-tree (pick-function)
17	(make-grow-tree (1- depth) nil)
18	(make-grow-tree (1- depth) nil))))))
19	
20	(defun make-tree (node left right)
21	"Make new tree as list"
22	(list node left right))

Source Code 5.3 Pembentukan Pohon

Keterangan:

Baris 1 : membuat pohon baru dengan metode *full*.

Baris 4 : membuat *node* terminal jika nilai kedalaman maksimal tercapai.

Baris 5 : membuat *node* fungsi.

Baris 6-7 : membuat *node* kiri dan kanan secara rekursif.

Baris 9 : membuat pohon baru dengan metode *grow*.

Baris 13-15 : membuat *node* terminal jika kedalaman maksimal tercapai atau pembentukan pohon dihentikan secara acak.

Baris 22 : membentuk pohon menggunakan *list*

Perhitungan nilai *fitness* ditunjukkan pada Source Code 5.4.

No.	Kode Sumber
1	(defun calculate-fitness (tree)
2	"Calculate fitness of tree"
3	(let ((node (number-of-node tree))
4	(accuracy (accuracy tree *data-training-imbalanced*))
5	(weight 0.0))
6	(+ (* weight (/ (1+ node)))
7	(* (- 1 weight)
8	accuracy)))
9)
10	(defun accuracy (tree data)
11	"Calculate percentage of tree accuracy with data"
12	(let ((correct-count 0)
13	(total-data (length data)))
14	(loop for d in data do
15	(when (equal (outcome d)
16	(parse-tree tree d))
17	(incf correct-count)))
18	(/ correct-count total-data)))
19)
20	(defun parse-tree (tree datum)
21	"Parse tree with datum and return the terminal node"
22	(if (atom tree)
23	tree
24	(if (tree-predicate (head-node tree) datum)
25	(parse-tree (left-node tree) datum)
26	(parse-tree (right-node tree) datum)))

Source Code 5.4 Perhitungan *Fitness*

Keterangan:

Baris 3 : mendapatkan jumlah *node* pada pohon.

Baris 4 : menghitung akurasi pohon berdasarkan data *training*.

Baris 6-8 : menghitung nilai *fitness* berdasarkan akurasi dan nilai bobot.

Baris 14-17 : menghitung jumlah data *training* yang sesuai dengan hasil pohon.

Baris 18 : menghitung akurasi berdasarkan nilai yang dihasilkan pohon.

Baris 22 : membentuk pohon menggunakan *list*.

Baris 18 : menghitung akurasi berdasarkan nilai yang dihasilkan pohon.

Baris 20 : mendapatkan nilai yang dihasilkan pohon sesuai dengan datum pada data *training*.

Baris 22-23 : mengembalikan *node* terminal atau nilai yang dihasilkan pohon.

Baris 24 : melakukan evaluasi pada *node* fungsi.

Baris 25 : melakukan evaluasi pada *node* kiri jika hasil evaluasi *node* fungsi bernilai benar.



Baris 26 : melakukan evaluasi pada *node* kanan jika hasil evaluasi *node* fungsi bernilai salah.

5.2.2 Iterasi

Proses iterasi dilakukan dengan membentuk populasi baru dengan proses mutasi, reproduksi, dan *crossover*. Implementasi iterasi ditunjukkan pada Source Code 5.5.

No.	Kode Sumber
1	(defun iterate (population n-mutate n-reproduce n-crossover
2	iteration)
3	"Return new population after iteration"
4	(if (zerop iteration)
5	population
6	(let ((new-population (make-empty-population)))
7	(setf new-population (append (mutate-programs n-mutate
8	population)
9	(reproduce-programs n-reproduce population)
10	(crossover-programs n-crossover population))
11	(setf new-population (stable-sort new-population #'>
12	:key #'program-fitness))
13	(iterate new-population n-mutate n-reproduce
14	n-crossover (1- iteration))))

Source Code 5.5 Iterasi

Keterangan:

Baris 4-5 : mengembalikan populasi hasil iterasi setelah proses iterasi selesai.

Baris 6 : membuat populasi baru berupa *list* kosong.

Baris 7 : melakukan proses mutasi sebanyak nilai *n-mutate*.

Baris 9 : melakukan proses reproduksi.

Baris 10 : melakukan proses *crossover* sebanyak nilai *n-crossover*.

Baris 11 : mengurutkan individu baru hasil proses mutasi, reproduksi, dan *crossover* berdasarkan nilai *fitness*.

Baris 13 : meneruskan proses iterasi secara rekursif.

5.2.3 Mutasi

Mutasi dilakukan dengan mengganti *node* pada pohon induk dengan *node* pohon baru. Implementasi mutasi ditunjukkan pada Source Code 5.6.



No.	Kode Sumber
1	(defun mutate (program)
2	"Generate new program by mutate old program non-destructively"
3	(let ((new-node (make-grow-tree (random-in-range 1 2)))
4	(old-tree (program-tree program))
5	(mutation-point (recombination-point program)))
6	(create-program (replace-node-in-tree old-tree
7	mutation-point
8	new-node))))

Source Code 5.6 Mutasi

Keterangan:

Baris 3 : membuat *node* pohon baru dengan metode *grow*.

Baris 4 : mendapatkan pohon dari induk mutasi.

Baris 5 : mendapatkan posisi *node* mutasi dari pohon induk.

Baris 6-8 : mengubah *node* pada pohon lama dengan *node* baru yang telah terbentuk dan membuat individu baru dari pohon hasil proses mutasi

5.2.4 Reproduksi

Reproduksi dilakukan dengan memilih individu dengan *fitness* terbaik sesuai dengan jumlah *reproduction rate*. Implementasi reproduksi ditunjukkan pada Source Code 5.7.

No.	Kode Sumber
1	(defun reproduce-programs (n sorted-population)
2	"Return list of n reproduced program from population"
3	(subseq sorted-population 0 n))

Source Code 5.7 Reproduksi

Keterangan:

Baris 3 : mendapatkan individu pada *sorted list* dari *index 0* hingga *index n*

5.2.5 Crossover

Crossover dilakukan dengan mengganti *node* pada pohon induk 1 dengan *subpohon* dari pohon 2. Implementasi *crossover* ditunjukkan pada Source Code 5.8.

No.	Kode Sumber
1	(defun crossover (program1 program2)
2	"Generate new program by replacing random program1 node with
3	random program2 node"
4	(let ((old-tree (program-tree program1))



```

5 (crossover-point (recombination-point program1))
6 (new-node (pick-random-node program2)))
7 (create-program (replace-node-in-tree old-tree
8 crossover-point
9 new-node)))

```

Source Code 5.8 Crossover

Keterangan:

- Baris 4 : mendapatkan pohon dari induk mutasi.
- Baris 5 : mendapatkan posisi *node* crossover dari pohon induk 1.
- Baris 6 : mendapatkan *subtree* dari pohon induk 2.
- Baris 7-9 : membuat pohon baru dengan mengubah *node* pada pohon induk 1 dengan *subtree* dari pohon induk 2 dan membuat individu baru dari pohon baru.



BAB 6 PENGUJIAN DAN ANALISIS

Bagian ini membahas hasil pengujian dan analisis dari penelitian ini. Pengujian yang dilakukan berdasar pada jumlah populasi, nilai bobot pada perhitungan *fitness*, jumlah iterasi, dan data latih.

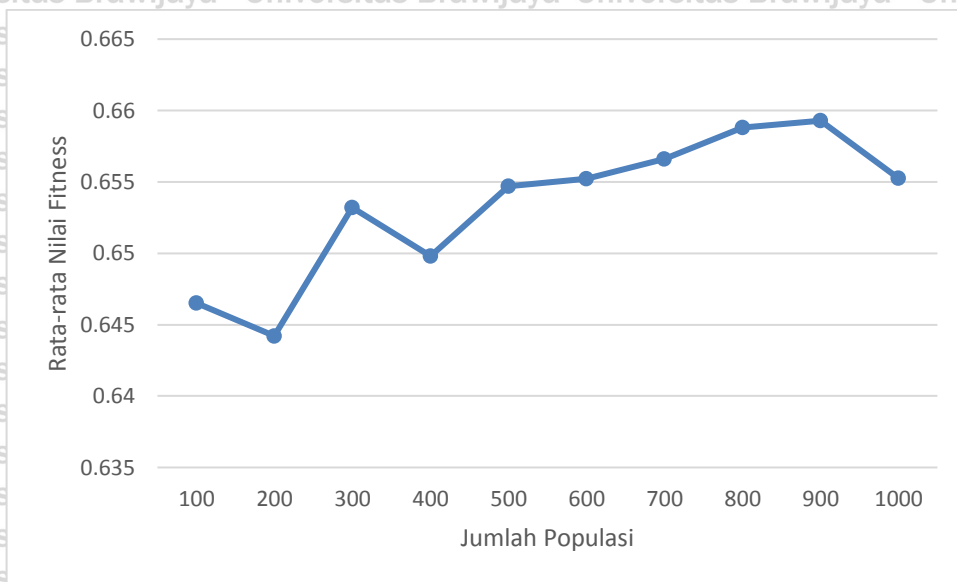
6.1 Pengujian Berdasarkan Jumlah Populasi

Pengujian pertama yang dilakukan adalah pengujian berdasarkan jumlah populasi. Pengujian ini dilakukan untuk mengetahui pengaruh jumlah populasi terhadap nilai *fitness* dari individu yang dihasilkan. Jumlah populasi yang digunakan untuk pengujian memiliki nilai kelipatan ratusan yang dimulai dari 100 hingga 500. Pengujian untuk tiap nilai populasi dilakukan sebanyak sepuluh kali. Nilai *fitness* rata-rata dari setiap jumlah populasi akan dibandingkan. Hasil pengujian ditunjukkan pada tabel 6.1.

Tabel 6.1 Pengujian dengan Jumlah Populasi

Jumlah Populasi	Nilai <i>fitness</i> pengujian ke-i										Rata-rata nilai <i>fitness</i>
	1	2	3	4	5	6	7	8	9	10	
100	0.639	0.639	0.651	0.664	0.639	0.648	0.651	0.628	0.643	0.664	0.6465
200	0.649	0.634	0.663	0.648	0.648	0.630	0.646	0.651	0.627	0.648	0.6442
300	0.663	0.649	0.663	0.655	0.633	0.664	0.664	0.642	0.649	0.651	0.6532
400	0.646	0.651	0.645	0.649	0.651	0.648	0.664	0.649	0.648	0.648	0.6498
500	0.661	0.649	0.645	0.664	0.649	0.648	0.664	0.654	0.649	0.664	0.6547
600	0.664	0.646	0.648	0.663	0.655	0.655	0.651	0.660	0.661	0.649	0.6552
700	0.648	0.664	0.651	0.651	0.661	0.649	0.660	0.664	0.663	0.655	0.6566
800	0.666	0.664	0.648	0.646	0.664	0.661	0.664	0.648	0.664	0.663	0.6588
900	0.666	0.652	0.655	0.654	0.664	0.646	0.664	0.666	0.666	0.660	0.6593
1000	0.654	0.664	0.648	0.663	0.651	0.655	0.652	0.660	0.651	0.655	0.6552

Perubahan rata-rata nilai *fitness* pada pengujian jumlah populasi ditunjukkan pada Gambar 6.1. Hasil pengujian pada Tabel 6.1 menunjukkan pengaruh jumlah populasi terhadap nilai *fitness* akhir yang dihasilkan. Nilai *fitness* rata-rata mencapai nilai tertinggi yaitu 0,6593 saat populasi berjumlah 900.



Gambar 6.1 Grafik Hasil Pengujian Jumlah Populasi

Berdasarkan Gambar 6.1, semakin meningkatnya jumlah populasi membuat nilai *fitness* semakin meningkat. Walaupun pada jumlah populasi antara 100 hingga 400, nilai *fitness* mengalami naik turun. Nilai *fitness* saat populasi berjumlah 100 lebih besar dari pada nilai *fitness* saat populasi berjumlah 200. Begitu pula ketika populasi berjumlah 300, dimana nilai *fitness* lebih tinggi daripada saat populasi berjumlah 400. Setelah itu nilai *fitness* meningkat hingga puncaknya pada saat populasi berjumlah 900. Nilai *fitness* tertinggi yang didapatkan sebesar 0,6593. Peningkatan nilai *fitness* tersebut disebabkan oleh ruang pencarian solusi yang semakin luas ketika jumlah populasi ditambah.

6.2 Pengujian Berdasarkan Nilai Bobot pada Perhitungan *Fitness*

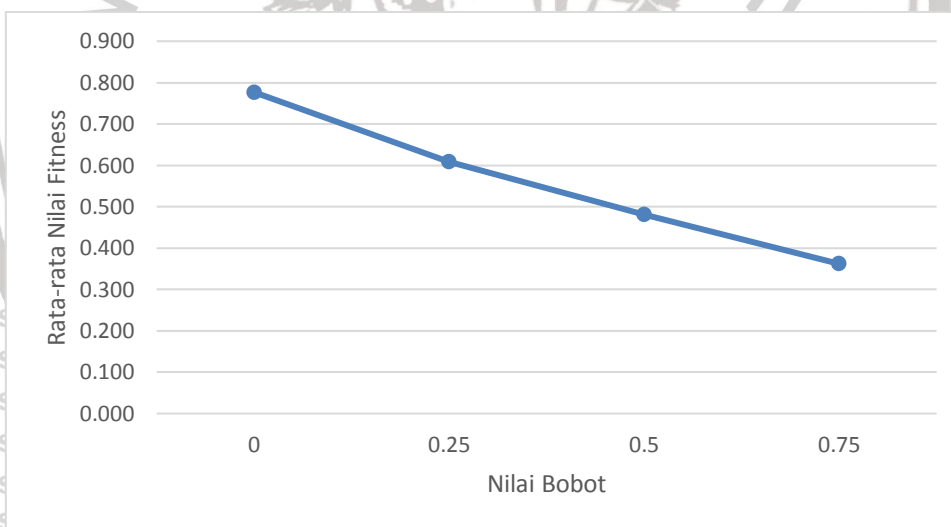
Pengujian pada bagian ini berdasarkan pada nilai bobot yang digunakan saat perhitungan nilai *fitness* yang ditunjukkan oleh Persamaan 6.1. Pengujian ini dilakukan untuk mengetahui pengaruh perubahan nilai bobot saat perhitungan *fitness*. Nilai bobot yang digunakan pada pengujian ini adalah dimulai dari nilai 0 hingga 0,75 dengan peningkatan sebesar 0,25. Jumlah populasi pada pengujian ini sebesar 400. Pengujian dilakukan sepuluh kali dan nilai *fitness* rata-rata dari tiap bobot akan dibandingkan. Hasil pengujian ditunjukkan oleh Tabel 6.2.

$$fitness = \left(w \times \frac{1}{n+1} \right) + ((1 - w) \times akurasi) \tag{2.1}$$

Tabel 6.2 Pengujian dengan Nilai Bobot

Nilai bobot	Nilai <i>fitness</i> pengujian ke-i										Rata-rata nilai <i>fitness</i>
	1	2	3	4	5	6	7	8	9	10	
0	0.785	0.777	0.773	0.781	0.790	0.798	0.785	0.792	0.787	0.779	0.785
0,25	0.638	0.626	0.626	0.618	0.637	0.620	0.618	0.618	0.626	0.626	0.625
0,5	0.496	0.461	0.471	0.461	0.509	0.460	0.490	0.462	0.467	0.509	0.479
0,75	0.379	0.355	0.362	0.375	0.362	0.356	0.375	0.374	0.355	0.356	0.365

Perubahan rata-rata nilai *fitness* pada pengujian berdasarkan nilai bobot pada perhitungan *fitness* ditunjukkan pada Gambar 6.2. Hasil pengujian pada Tabel 6.2 menunjukkan pengaruh nilai bobot pada perhitungan *fitness* terhadap rata-rata nilai *fitness* yang dimiliki oleh individu yang didapatkan. Gambar 6.2 menunjukkan grafik perubahan nilai *fitness* berdasarkan Tabel 6.2. Nilai *fitness* semakin kecil ketika nilai bobot saat perhitungan *fitness* bertambah. Nilai *fitness* rata-rata mencapai nilai tertinggi yaitu 0,785 saat bobot memiliki nilai 0.



Gambar 6.2 Grafik Hasil Pengujian Nilai Bobot Perhitungan *Fitness*

Nilai bobot pada perhitungan nilai *fitness* dapat memberi perubahan pada nilai *fitness* dari individu yang dihasilkan. Gambar 6.2 menunjukkan semakin meningkatnya nilai bobot, nilai *fitness* semakin turun. Pada perhitungan nilai *fitness*, nilai bobot menentukan perbandingan pengaruh antara nilai akurasi terhadap data latih dan jumlah *node* pada pohon terhadap besarnya *fitness*. Semakin besar nilai bobot pengaruh jumlah *node* pohon terhadap nilai *fitness* semakin besar. Pengaruh akurasi terhadap data latih terhadap nilai *fitness* semakin kecil.



Ketika nilai bobot bernilai 0,5 yang menunjukkan pengaruh jumlah *node* dan nilai akurasi yang sama, nilai *fitness* rendah. Hal itu disebabkan oleh pengaruh jumlah *node* yang kecil pada perhitungan *fitness* pada persamaan 6.1. Nilai *node* terkecil yang mungkin adalah satu. Pada persamaan 6.1, jumlah tersebut akan menghasilkan nilai hanya sebesar 0,25. Sedangkan nilai akurasi terhadap data latih dapat memiliki nilai lebih dari 50%. Pada persamaan 6.1, nilai akurasi dapat menghasilkan nilai lebih dari 0,25 ketika akurasi terhadap data latih memiliki nilai lebih dari 50%.

6.3 Pengujian Berdasarkan Jumlah Iterasi

Pengujian berdasarkan jumlah iterasi dilakukan untuk mengetahui pengaruh jumlah iterasi terhadap nilai *fitness* individu yang dihasilkan. Jumlah iterasi yang digunakan merupakan nilai antara 100 hingga 1000 yang memiliki kelipatan 100 ditambah dengan iterasi dengan jumlah 5000 dan 10000. Pada pengujian ini jumlah populasi yang digunakan adalah 400 dan nilai bobot pada perhitungan *fitness* sebesar 0,2. Hasil pengujian terdapat pada Tabel 6.3.

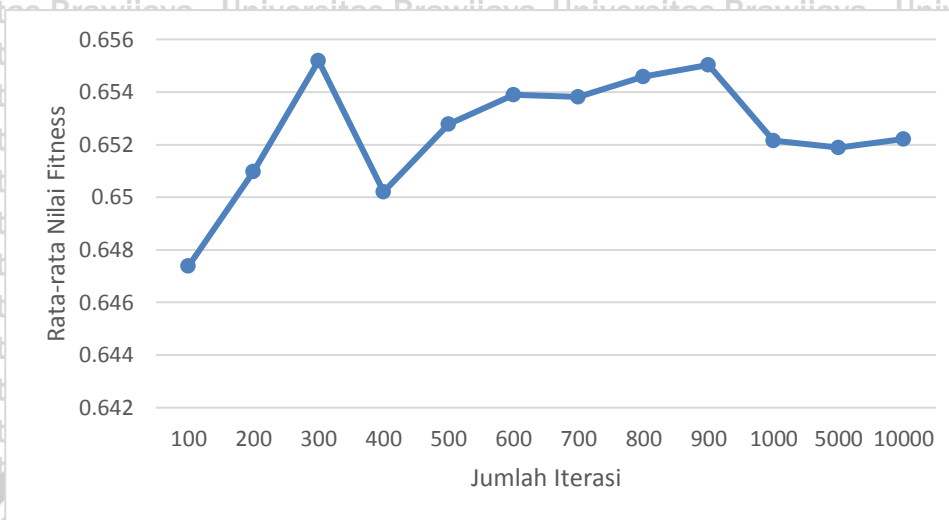
Tabel 6.3 Pengujian dengan Jumlah Iterasi

Jumlah Iterasi	Nilai <i>fitness</i> pengujian ke-i										Rata-rata nilai <i>fitness</i>
	1	2	3	4	5	6	7	8	9	10	
100	0.661	0.645	0.651	0.648	0.649	0.652	0.651	0.633	0.639	0.646	0.6474
200	0.655	0.651	0.634	0.646	0.648	0.649	0.651	0.646	0.666	0.664	0.6510
300	0.663	0.649	0.663	0.661	0.663	0.646	0.648	0.664	0.648	0.648	0.6552
400	0.651	0.649	0.648	0.651	0.648	0.646	0.664	0.648	0.648	0.651	0.6502
500	0.664	0.648	0.639	0.648	0.655	0.648	0.654	0.664	0.660	0.649	0.6528
600	0.651	0.648	0.648	0.646	0.655	0.664	0.654	0.664	0.661	0.648	0.6539
700	0.649	0.648	0.655	0.661	0.649	0.652	0.664	0.651	0.663	0.646	0.6538
800	0.646	0.655	0.649	0.649	0.651	0.655	0.664	0.666	0.655	0.655	0.6546
900	0.655	0.661	0.646	0.649	0.646	0.651	0.649	0.664	0.664	0.664	0.6550
1000	0.664	0.651	0.655	0.649	0.651	0.649	0.649	0.663	0.643	0.648	0.6522
5000	0.649	0.649	0.649	0.649	0.655	0.664	0.655	0.649	0.651	0.648	0.6519
10000	0.648	0.663	0.646	0.655	0.651	0.649	0.651	0.649	0.661	0.649	0.6522

Perubahan rata-rata nilai *fitness* pada pengujian berdasarkan jumlah iterasi ditunjukkan pada Gambar 6.3. Grafik pada Gambar 6.3 menunjukkan perubahan nilai *fitness* berdasarkan perubahan jumlah iterasi. Nilai *fitness* rata-rata mencapai nilai tertinggi yaitu 0,6552 saat iterasi berjumlah 300. Perbedaan nilai *fitness* tertinggi dengan nilai *fitness* yang lain tidak besar. Pencarian solusi pada



genetic programming pada umumnya lebih optimal dilakukan pada generasi awal (Poli et al., 2008). Hal tersebut dapat menyebabkan jumlah iterasi yang besar tidak memberikan pengaruh terhadap nilai *fitness* yang dihasilkan.



Gambar 6.3 Grafik Hasil Pengujian Jumlah Iterasi

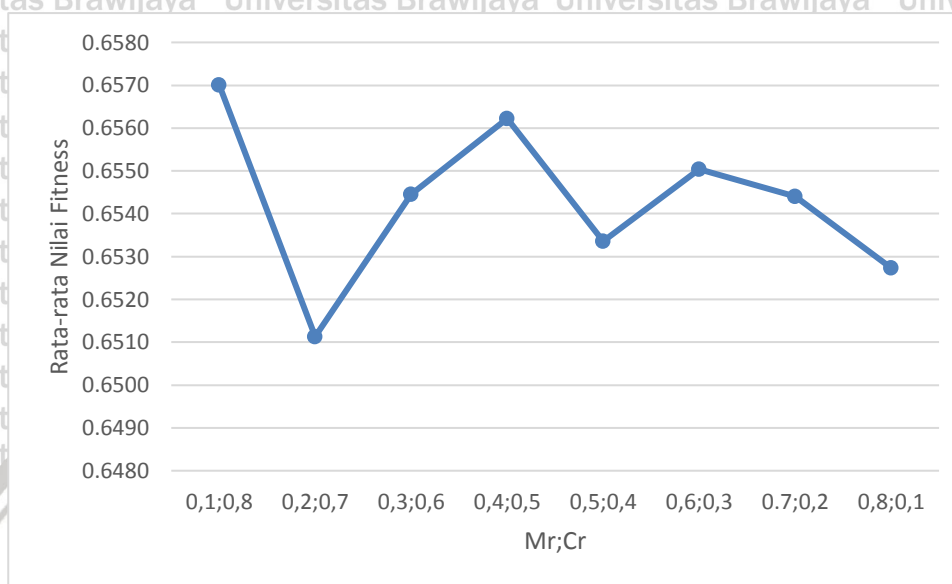
6.4 Pengujian Berdasarkan Nilai Mr dan Cr

Pengujian berdasarkan nilai *mutation rate* dan *crossover rate* dilakukan untuk mengetahui pengaruh parameter tersebut terhadap nilai *fitness* individu yang dihasilkan. Nilai yang digunakan adalah nilai antara 0,1 hingga 0,8 dengan kelipatan 0,1. Pada pengujian ini jumlah populasi yang digunakan adalah 400, jumlah iterasi yang dilakukan sejumlah 300 dan nilai bobot pada perhitungan *fitness* sebesar 0,2. Hasil pengujian terdapat pada Tabel 6.4.

Tabel 6.4 Pengujian dengan Nilai Mr dan Cr

Nilai Mr; Cr	Nilai <i>fitness</i> pengujian ke-i										Rata-rata nilai <i>fitness</i>
	1	2	3	4	5	6	7	8	9	10	
0,1;0,8	0.663	0.655	0.648	0.651	0.664	0.666	0.649	0.651	0.664	0.660	0.6570
0,2;0,7	0.648	0.651	0.648	0.649	0.655	0.648	0.650	0.663	0.649	0.651	0.6511
0,3;0,6	0.663	0.648	0.666	0.666	0.648	0.648	0.649	0.655	0.649	0.654	0.6544
0,4;0,5	0.663	0.649	0.651	0.664	0.664	0.655	0.651	0.651	0.648	0.666	0.6562
0,5;0,4	0.663	0.649	0.651	0.654	0.648	0.649	0.652	0.664	0.649	0.655	0.6534
0,6;0,3	0.663	0.649	0.663	0.660	0.649	0.661	0.655	0.649	0.649	0.652	0.6550
0,7;0,2	0.664	0.649	0.649	0.649	0.660	0.660	0.651	0.649	0.649	0.664	0.6544
0,8;0,1	0.648	0.663	0.649	0.648	0.655	0.664	0.649	0.655	0.649	0.648	0.6527

Perubahan rata-rata nilai *fitness* pada pengujian berdasarkan nilai *mutation rate* dan *crossover rate* ditunjukkan pada Gambar 6.4.



Gambar 6.4 Grafik Hasil Pengujian Nilai Mr dan Cr

Perubahan pada nilai *mutation rate* dan *crossover rate* dapat menyebabkan perubahan pada nilai *fitness*. Pada Gambar 6.4, nilai *fitness* tertinggi dapat dicapai saat *mutation rate* bernilai 0,1 dan *crossover rate* bernilai 0,8. Proses iterasi dengan nilai *fitness* terbaik dihasilkan dengan pembuatan individu baru dengan metode *crossover* yang lebih banyak dari pada metode mutasi. Pada umumnya metode *genetic programming* memiliki probabilitas *crossover rate* yang jauh lebih tinggi dibandingkan dengan *mutation rate* (Poli et al., 2008).

6.5 Pengujian Berdasarkan Data Latih

Pengujian ini dilakukan untuk mengetahui pengaruh data latih terhadap nilai *fitness* dari individu yang dihasilkan. Pengujian pertama dilakukan dengan data latih yang memiliki perbandingan yang sama antara jumlah penderita diabetes dan yang tidak. Sedangkan pengujian kedua menggunakan data latih dengan perbandingan antara penderita diabetes dengan yang tidak menderita diabetes sebesar 1:2. Jumlah populasi bernilai 400 dan nilai bobot yang digunakan berjumlah 0,2. Iterasi yang dilakukan sejumlah 50 kali. Hasil pengujian ditunjukkan pada Tabel 6.5.

Tabel 6.5 Pengujian Berdasarkan Data Latih

Data Latih	Nilai <i>fitness</i> pengujian ke-i										Rata-rata nilai <i>fitness</i>
	1	2	3	4	5	6	7	8	9	10	
178:178	0.772	0.761	0.778	0.770	0.764	0.781	0.784	0.792	0.801	0.789	0.779
178:356	0.788	0.770	0.787	0.796	0.781	0.777	0.777	0.787	0.792	0.783	0.784

Pada Tabel 6.5, nilai *fitness* tertinggi diraih saat data latih yang digunakan memiliki perbandingan yang berbeda. Meskipun terdapat perbedaan, perbedaan nilai *fitness* yang dihasilkan tidak terlalu besar.

6.6 Pengujian Validasi

Pengujian ini dilakukan menggunakan nilai parameter yang menghasilkan rata-rata *fitness* tertinggi dari pengujian sebelumnya. Jumlah populasi yang digunakan sebanyak 900. Nilai bobot yang digunakan adalah 0. Iterasi yang dilakukan sejumlah 300 kali. *Mutation rate* yang digunakan bernilai 0,1 dan nilai *crossover rate* sebesar 0,9. Hasil pengujian ditunjukkan pada Tabel 6.6. Pengujian dilakukan sebanyak sepuluh kali. Nilai *fitness* tertinggi yang dihasilkan sebesar 0,84457. Rata-rata dari keseluruhan nilai *fitness* hasil pengujian adalah 0,82566.

Tabel 6.6 Hasil Pengujian Validasi

Pengujian ke-	Nilai <i>Fitness</i>
1	0.83708
2	0.82210
3	0.81273
4	0.82584
5	0.79213
6	0.82772
7	0.83146
8	0.83708
9	0.84457
10	0.82584
Nilai rata-rata <i>fitness</i>	0.82566

Pohon keputusan dari individu dengan *fitness* tertinggi yang didapatkan dengan metode *genetic programming* kemudian digunakan untuk melakukan



penentuan penyakit diabetes terhadap data uji. Data uji tersebut berupa data pasien yang berjumlah 180 orang, dengan 90 penderita diabetes dan 90 sisanya bukan penderita diabetes. Pohon keputusan yang digunakan adalah sebagai berikut.

```
(( (GL 0.6365285) (( (IN 0.9347515) (( (DP 0.874076) (( (ST 0.074047446)
(( (GL 0.7842332) (( (DP 0.19683552) (( (GL 0.63926816)
(( (BM 0.32779098) (( (DP 0.21806455) (( (BP 0.75144804)
(( (PR 0.77879035) (( (ST 0.48243833) TD ...
DB TD)) DB)) TD))))
```

Pohon keputusan tersebut memiliki *node* sebanyak 773. Nilai akurasi akan dihitung dengan membandingkan antara jumlah pasien yang dengan tepat ditentukan oleh pohon keputusan dan jumlah data pasien keseluruhan. Nilai akurasi yang didapat:

$$\text{akurasi} = \frac{119}{180} \times 100\% = 66,11\%$$

6.7 Analisis Hasil Pengujian

Perubahan nilai parameter dapat memberi pengaruh terhadap nilai *fitness* dari individu yang dihasilkan. Berdasarkan Gambar 6.1, semakin meningkatnya jumlah populasi belum tentu membuat nilai *fitness* semakin meningkat. Nilai *fitness* saat populasi berjumlah 200 lebih besar dari pada nilai *fitness* saat populasi berjumlah 300. Nilai *fitness* tertinggi diraih saat populasi memiliki nilai 900.

Pada Gambar 6.3, nilai *fitness* meningkat saat parameter jumlah iterasi meningkat dari nilai 100 hingga 300. Nilai *fitness* tertinggi tercapai saat jumlah iterasi bernilai 300. Nilai *fitness* turun saat jumlah iterasi bernilai 400 dan meningkat kembali seiring bertambahnya jumlah iterasi. Peningkatan tersebut terjadi hingga jumlah populasi bernilai 900. Semakin banyak jumlah populasi ruang pencarian solusi akan semakin luas. Hal tersebut dapat memberi pengaruh terhadap nilai *fitness*.

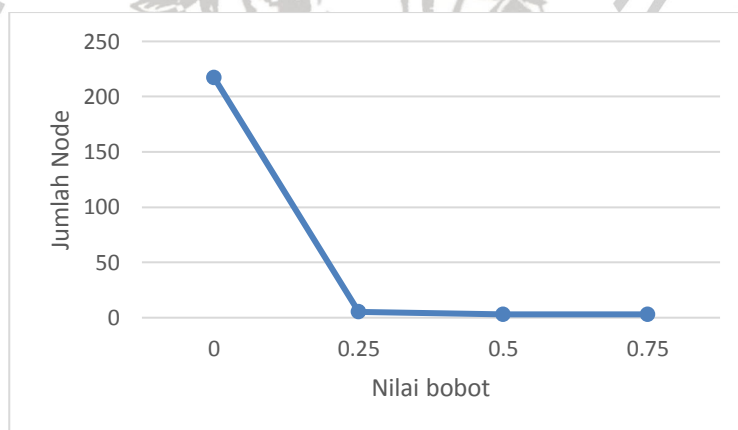
Perubahan pada nilai *mutation rate* dan *crossover rate* dapat menyebabkan perubahan pada nilai *fitness*. Pada Gambar 6.4, nilai *fitness* tertinggi dapat dicapai saat *mutation rate* bernilai 0,1 dan *crossover rate* bernilai 0,8. Proses iterasi dengan nilai *fitness* terbaik dihasilkan dengan pembuatan individu baru dengan metode *crossover* yang lebih banyak dari pada metode mutasi.

Nilai *fitness* mengalami perubahan saat data latih yang digunakan memiliki perbedaan. Meskipun perubahan tersebut tidak terlalu besar. Perbedaan

tersebut ditunjukkan pada Tabel 6.5. Data latihan yang tidak seimbang menghasilkan nilai *fitness* yang lebih tinggi dibandingkan nilai *fitness* yang dihasilkan dengan data latihan yang seimbang.

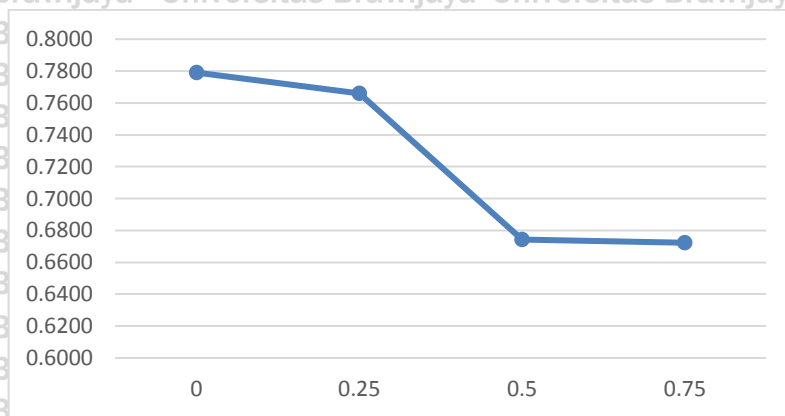
Nilai bobot pada perhitungan nilai *fitness* dapat memberi perubahan pada nilai *fitness* dari individu yang dihasilkan. Gambar 6.2 menunjukkan semakin meningkatnya nilai bobot, nilai *fitness* semakin turun. Pada perhitungan nilai *fitness*, nilai bobot menentukan perbandingan pengaruh antara nilai akurasi terhadap data latihan dan jumlah *node* pada pohon terhadap besarnya *fitness*. Semakin besar nilai bobot pengaruh jumlah *node* pohon terhadap nilai *fitness* semakin besar. Pengaruh akurasi terhadap data latihan terhadap nilai *fitness* semakin kecil.

Gambar 6.5 menunjukkan pengaruh nilai bobot saat perhitungan *fitness* terhadap jumlah *node* pohon. Jumlah *node* menjadi sangat banyak ketika bobot memiliki nilai 0. Hal tersebut dikarenakan jumlah *node* pada individu sama sekali tidak memiliki pengaruh pada saat perhitungan nilai *fitness*. Operasi *crossover* dan mutasi akan menyebabkan perubahan terhadap individu lama. Individu baru yang dihasilkan akan berpotensi memiliki jumlah *node* yang semakin bertambah seiring dengan proses iterasi.



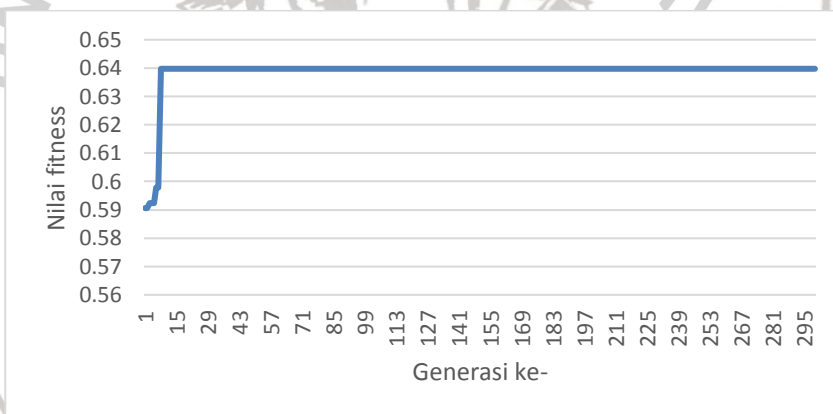
Gambar 6.5 Grafik Jumlah *Node* Pohon

Perbedaan jumlah *node* saat nilai bobot bernilai 0 dan 0,25 sangat besar. Sedangkan pada Gambar 6.6 perbedaan nilai akurasi terhadap data latihan saat bobot bernilai 0 dan 0,25 tidak besar. Nilai akurasi dapat dikorbankan untuk mendapatkan individu dengan jumlah *node* yang sedikit. Hal tersebut dapat membantu jika kondisi memori sistem terlalu sedikit atau waktu yang dibutuhkan menghitung nilai *fitness* terlalu lama. Perhitungan *fitness* merupakan tahapan yang dapat memakan waktu lama terutama jika individu memiliki *node* yang banyak (Poli et al., 2008).



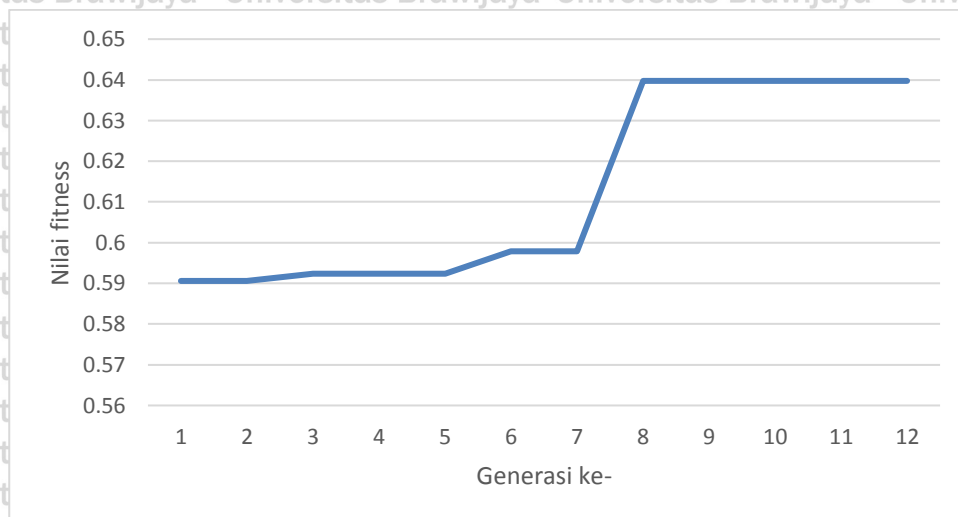
Gambar 6.6 Grafik Akurasi Terhadap Data Latih

Dengan parameter yang dapat menghasilkan nilai *fitness* terbaik saat pengujian, Individu baru dibentuk. Individu baru tersebut memiliki nilai *fitness* 0,6397472, dengan akurasi terhadap data latih yang tidak seimbang sebesar 0,769. Jumlah *node* yang dihasilkan sebesar 3 karena nilai bobot yang digunakan adalah 0,25.



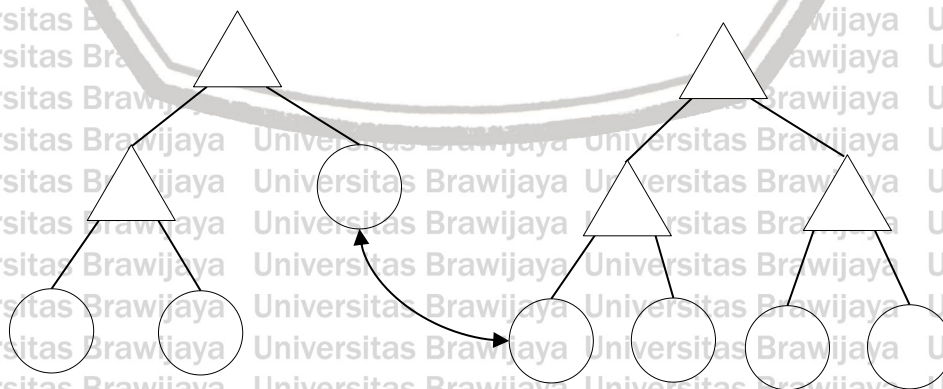
Gambar 6.7 Grafik Perubahan Nilai *Fitness* Setiap Generasi

Gambar 6.7 menunjukkan perubahan nilai *fitness* di setiap iterasi. Nilai *fitness* mencapai puncak pada saat iterasi ke-8 yang ditunjukkan pada Gambar 6.8. Hal tersebut disebabkan karena metode *genetic programming* menghasilkan pencarian solusi yang maksimal saat generasi awal (Poli et al., 2008). Oleh karena itu, metode ini menggunakan jumlah populasi yang besar. Jumlah populasi untuk pengujian yang digunakan sebesar 900.



Gambar 6.8 Grafik Perubahan Nilai *Fitness* Generasi Awal

Hal lain yang dapat menyebabkan nilai *fitness* mencapai maksimal ketika generasi masih berada pada tahapan awal adalah kurangnya variasi pada individu hasil *crossover*. Salah satu cara untuk mencegah hal tersebut adalah dengan memilih *node* fungsi sebagai *crossover point* (Poli et al., 2008). Jika *node* terminal terpilih pada kedua induk, individu baru yang dihasilkan akan tidak terlalu berbeda dengan induknya. Hal tersebut akan sering terjadi karena kemungkinan *node* terminal terpilih saat proses *crossover* lebih besar daripada terpilihnya *node* fungsi. Jumlah *node* terminal pada pohon selalu lebih banyak daripada jumlah *node* fungsi seperti yang ditunjukkan oleh Gambar 6.9. Pada gambar tersebut, *node* fungsi ditunjukkan oleh segitiga dan *node* terminal ditunjukkan oleh lingkaran. Tanda panah menunjukkan pertukaran antara *node* terminal pada kedua pohon yang menyebabkan kurangnya variasi saat proses *crossover*.



Gambar 6.9 *Crossover* pada *Node Terminal*

Individu yang dihasilkan dengan parameter terbaik sesuai dengan pengujian sebelumnya digunakan untuk menghitung akurasi berdasarkan data uji yang berbeda dengan data latih yang digunakan. Individu terbaik tersebut memiliki nilai *fitness* sebesar 0,84457 sesuai dengan Tabel 6.6.

Individu terbaik tersebut kemudian digunakan untuk melakukan penentuan penderita diabetes. Data uji yang digunakan adalah 180 data pasien dengan perbandingan antara penderita diabetes dan yang bukan 1:1. Akurasi penentuan penderita diabetes melitus yang didapatkan adalah sebesar 66,11 %.



BAB 7 PENUTUP

Bagian ini memuat kesimpulan dan saran terhadap penelitian yang telah dilakukan.

7.1 Kesimpulan

Kesimpulan dari penelitian ini adalah sebagai berikut:

1. Metode *genetic programming* dapat digunakan untuk menghasilkan pohon keputusan untuk menentukan penderita diabetes melitus. Pada implementasi *genetic programming*, individu direpresentasikan dengan menggunakan pohon keputusan. Pohon keputusan tersebutlah yang nantinya akan digunakan untuk menentukan penderita diabetes.
2. Parameter *genetic programming* yang dapat menghasilkan individu dengan nilai maksimal adalah jumlah populasi bernilai 900, nilai bobot perhitungan *fitness* 0, jumlah iterasi maksimal 300, nilai *mutation rate* 0,1, nilai *crossover rate* 0,8, dan perbandingan jumlah penderita diabetes dan bukan penderita diabetes pada data latih bernilai 178:356. Parameter tersebut digunakan untuk menghasilkan pohon keputusan yang digunakan untuk menentukan penderita diabetes melitus.
3. Pohon keputusan yang dihasilkan dengan metode *genetic programming* digunakan untuk melakukan penentuan penderita diabetes pada 180 data uji pasien. Nilai akurasi didapatkan dengan membandingkan antara hasil penentuan yang didapatkan dari pohon keputusan dan kondisi pasien sebenarnya. Nilai akurasi yang dihasilkan adalah sebesar 66,11%.

7.2 Saran

Beberapa teknik dapat digunakan untuk menghasilkan nilai akurasi yang lebih tinggi pada penelitian berikutnya. Cara tersebut antara lain dengan menggunakan metode pembuatan populasi awal yang lebih beragam, metode untuk mengendalikan jumlah *node*, dan pemilihan metode *crossover* yang dapat membuat hasil individu yang lebih beragam.

DAFTAR REFERENSI

Eggermont, J., Kok, J.N. and Kusters, W.A., 2003. Genetic programming for data classification: Refining the search space. Proceedings of the 15th Belgium/Netherlands Conference on Artificial Intelligence (BNAIC'03) hal 123-130.

Espejo, P.G., Ventura, S. and Herrera, F., 2010. A survey on the application of genetic programming to classification. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), Vol. 40, No. 2, hal 121-144.

Estrada-Gil, J.K., Fernández-López, J.C., Hernández-Lemus, E., Silva-Zolezzi, I., Hidalgo-Miranda, A., Jiménez-Sánchez, G. and Vallejo-Clemente, E.E., 2007. GPDTI: A Genetic Programming Decision Tree Induction method to find epistatic effects in common complex diseases. Bioinformatics, Vol. 23, No. 13, hal i167-i174.

Poli, R., Langdon, W. B., and McPhee, N. F., 2008. A field guide to genetic programming. [online] Lulu Enterprises, UK Ltd. Tersedia di: <http://www.gp-field-guide.org.uk> [Diakses 11 September 2019]

Smith, J.W., Everhart, J.E., Dickson, W.C., Knowler, W.C., & Johannes, R.S., 1988. Using the ADAP learning algorithm to forecast the onset of diabetes mellitus. Proceedings of the Symposium on Computer Applications and Medical Care hal 261-265.

Tür, G. and Güvenir, H., 1996. Decision tree induction using genetic programming. Proceedings of the Fifth Turkish Symposium on Artificial Intelligence and Neural Networks.

WHO (World Health Organization), 2018. Diabetes. [online] World Health Organization. Tersedia di: <https://www.who.int/news-room/factsheets/detail/diabetes> [Diakses 10 September 2019]

LAMPIRAN A DATA LATIH

Jumlah Kehamilan	Glukosa	Tekanan Darah	Ketebalan Kulit	Insulin	Indeks Massa Tubuh	Diabetes Pedigree	Usia	Keluaran
0.588	0.200	0.592	0.174	0.020	0.309	0.079	0.283	0
0.059	0.555	0.367	0.174	0.188	0.213	0.262	0.000	0
0.118	0.258	0.265	0.174	0.075	0.249	0.380	0.000	0
0.471	0.490	0.551	0.174	0.020	0.139	0.141	0.717	0
0.000	0.613	0.388	0.109	0.236	0.080	0.055	0.000	0
0.529	0.303	0.449	0.174	0.020	0.123	0.052	0.617	0
0.118	0.303	0.388	0.174	0.020	0.186	0.191	0.017	0
0.176	0.355	0.306	0.130	0.087	0.151	0.032	0.050	0
0.412	0.523	0.633	0.174	0.020	0.397	0.096	0.500	0
0.765	0.206	0.367	0.174	0.020	0.299	0.044	0.333	0
0.353	0.548	0.673	0.000	0.375	0.029	0.215	0.650	0
0.118	0.155	0.469	0.272	0.063	0.139	0.047	0.067	0
0.176	0.516	0.571	0.283	0.139	0.307	0.097	0.083	0
0.353	0.452	0.490	0.174	0.020	0.282	0.047	0.083	0
0.176	0.523	0.347	0.174	0.020	0.274	0.031	0.050	0
0.176	0.277	0.367	0.120	0.020	0.074	0.156	0.000	0
0.059	0.342	0.408	0.130	0.082	0.000	0.094	0.000	0
0.176	0.465	0.510	0.087	0.109	0.166	0.012	0.050	0
0.000	0.471	0.429	0.261	0.209	0.258	0.177	0.017	0
0.000	0.432	0.418	0.174	0.020	0.131	0.249	0.167	0
0.118	0.503	0.367	0.120	0.111	0.237	0.273	0.017	0
0.000	0.406	0.531	0.174	0.020	0.554	0.260	0.050	0
0.059	0.271	0.429	0.489	0.061	0.472	0.358	0.133	0
0.353	0.303	0.490	0.174	0.020	0.237	0.181	0.167	0
0.059	0.213	0.327	0.250	0.050	0.309	0.501	0.050	0
0.000	0.394	0.673	0.174	0.020	0.233	0.051	0.417	0
0.000	0.084	0.367	0.174	0.020	0.072	0.281	0.767	0
0.000	0.535	0.571	0.326	0.236	0.370	0.310	0.033	0
0.235	0.258	0.673	0.174	0.050	0.436	0.035	0.067	0
0.059	0.284	0.551	0.239	0.075	0.282	0.123	0.133	0
0.235	0.561	0.449	0.152	0.183	0.305	0.035	0.117	0
0.059	0.774	0.592	0.391	0.064	0.299	0.112	0.483	0
0.235	0.935	0.878	0.261	0.020	0.211	0.257	0.267	0
0.059	0.465	0.469	0.228	0.020	0.188	0.054	0.000	0
0.176	0.258	0.449	0.250	0.111	0.280	0.219	0.067	0
0.353	0.452	0.653	0.174	0.020	0.196	0.072	0.750	0
0.059	0.284	0.388	0.185	0.036	0.239	0.147	0.033	0
0.059	0.258	0.408	0.174	0.121	0.382	0.168	0.117	0



0.412	0.516	0.469	0.283	0.242	0.149	0.035	0.267	0
0.059	0.342	0.469	0.359	0.020	0.407	0.060	0.150	0
0.471	0.426	0.531	0.174	0.020	0.196	0.068	0.617	0
0.647	0.381	0.449	0.359	0.020	0.573	0.020	0.350	0
0.647	0.265	0.510	0.174	0.020	0.243	0.095	0.233	0
0.059	0.277	0.449	0.293	0.076	0.397	0.138	0.050	0
0.353	0.355	0.367	0.130	0.048	0.178	0.179	0.183	0
0.000	0.303	0.571	0.174	0.020	0.290	0.223	0.100	0
0.118	0.329	0.306	0.076	0.089	0.162	0.286	0.017	0
0.059	0.355	0.490	0.250	0.005	0.417	0.143	0.000	0
0.353	0.310	0.388	0.272	0.135	0.282	0.003	0.417	0
0.235	0.710	0.490	0.239	0.135	0.268	0.111	0.267	0
0.176	0.219	0.469	0.174	0.020	0.292	0.082	0.300	0
0.118	0.555	0.735	0.174	0.020	0.090	0.081	0.000	0
0.176	0.432	0.347	0.261	0.036	0.231	0.150	0.017	0
0.118	0.348	0.367	0.109	0.127	0.337	0.051	0.017	0
0.059	0.639	0.633	0.250	0.380	0.243	0.348	0.033	0
0.059	0.484	0.204	0.435	0.059	0.354	0.086	0.067	0
0.353	0.413	0.204	0.141	0.139	0.119	0.314	0.233	0
0.588	0.574	0.449	0.174	0.020	0.180	0.071	0.250	0
0.353	0.419	0.367	0.217	0.020	0.139	0.055	0.100	0
0.706	0.497	0.551	0.109	0.020	0.170	0.077	0.683	0
0.471	0.361	0.531	0.174	0.020	0.419	0.048	0.350	0
0.059	0.316	0.327	0.043	0.020	0.088	0.145	0.017	0
0.353	0.381	0.429	0.174	0.020	0.125	0.073	0.133	0
0.000	0.187	0.490	0.174	0.020	0.059	0.113	0.067	0
0.118	0.439	0.551	0.467	0.151	0.434	0.041	0.050	0
0.118	0.245	0.286	0.163	0.121	0.211	0.692	0.067	0
0.353	0.510	0.490	0.413	0.260	0.315	0.280	0.217	0
0.000	0.148	0.531	0.174	0.020	0.554	0.050	0.417	0
0.059	0.290	0.000	0.130	0.013	0.196	0.205	0.000	0
0.059	0.419	0.143	0.120	0.127	0.100	0.140	0.083	0
0.059	0.413	0.653	0.130	0.020	0.182	0.137	0.050	0
0.353	0.335	0.490	0.174	0.020	0.112	0.048	0.117	0
0.059	0.516	0.510	0.315	0.020	0.196	0.009	0.150	0
0.059	0.516	0.367	0.272	0.020	0.360	0.186	0.000	0
0.059	0.310	0.388	0.196	0.032	0.027	0.173	0.067	0
0.000	0.697	0.592	0.348	0.310	0.476	0.082	0.100	0
0.059	0.432	0.388	0.065	0.202	0.119	0.026	0.033	0
0.176	0.400	0.306	0.152	0.173	0.260	0.091	0.050	0
0.353	0.394	0.571	0.228	0.020	0.292	0.342	0.083	0
0.176	0.400	0.490	0.174	0.020	0.155	0.055	0.100	0
0.353	0.471	0.735	0.174	0.020	0.215	0.034	0.150	0



0.118	0.155	0.388	0.065	0.001	0.039	0.076	0.033	0
0.118	0.439	0.633	0.380	0.175	0.413	0.072	0.117	0
0.118	0.310	0.531	0.141	0.020	0.123	0.692	0.117	0
0.353	0.897	0.714	0.174	0.020	0.462	0.591	0.400	0
0.000	0.323	0.469	0.217	0.121	0.517	0.115	0.000	0
0.118	0.413	0.408	0.174	0.020	0.258	0.034	0.000	0
0.235	0.297	0.653	0.435	0.048	0.399	0.121	0.133	0
0.000	0.523	0.449	0.174	0.020	0.133	0.055	0.000	0
0.000	0.568	0.551	0.174	0.020	0.290	0.135	0.000	0
0.294	0.542	0.571	0.174	0.020	0.335	0.028	0.400	0
0.235	0.323	0.418	0.163	0.020	0.133	0.030	0.000	0
0.000	0.374	0.551	0.359	0.091	0.333	0.068	0.050	0
0.118	0.432	0.367	0.174	0.020	0.164	0.113	0.033	0
0.059	0.542	0.592	0.109	0.203	0.190	0.016	0.017	0
0.588	0.310	0.388	0.174	0.020	0.157	0.038	0.167	0
0.294	0.387	0.510	0.174	0.020	0.217	0.032	0.450	0
0.118	0.323	0.531	0.120	0.063	0.274	0.244	0.033	0
0.059	0.361	0.510	0.054	0.038	0.027	0.030	0.117	0
0.000	0.374	0.633	0.109	0.109	0.227	0.263	0.100	0
0.235	0.542	0.469	0.174	0.020	0.329	0.096	0.050	0
0.235	0.297	0.490	0.174	0.020	0.200	0.227	0.167	0
0.176	0.381	0.490	0.250	0.166	0.192	0.278	0.100	0
0.118	0.729	0.510	0.304	0.512	0.434	0.024	0.150	0
0.000	0.406	0.367	0.196	0.020	0.168	0.023	0.033	0
0.059	0.303	0.306	0.196	0.103	0.143	0.067	0.033	0
0.059	0.471	0.367	0.174	0.111	0.319	0.166	0.100	0
0.294	0.510	0.510	0.359	0.076	0.325	0.082	0.117	0
0.118	0.490	0.306	0.174	0.020	0.176	0.161	0.100	0
0.059	0.400	0.469	0.228	0.145	0.327	0.027	0.017	0
0.118	0.368	0.347	0.304	0.091	0.074	0.033	0.017	0
0.059	0.490	0.571	0.446	0.224	0.423	0.463	0.333	0
0.647	0.535	0.837	0.174	0.020	0.425	0.048	0.500	0
0.588	0.761	0.612	0.174	0.020	0.194	0.044	0.550	0
0.059	0.439	0.571	0.413	0.142	0.339	0.059	0.050	0
0.353	0.348	0.347	0.283	0.212	0.323	0.150	0.367	0
0.529	0.710	0.551	0.250	0.103	0.260	0.037	0.400	0
0.353	0.781	0.449	0.207	0.185	0.315	0.236	0.467	0
0.059	0.355	0.347	0.033	0.020	0.147	0.202	0.000	0
0.588	0.155	0.837	0.174	0.042	0.354	0.088	0.433	0
0.176	0.510	0.776	0.304	0.272	0.800	0.342	0.017	0
0.471	0.303	0.592	0.174	0.020	0.356	0.217	0.783	0
0.000	0.316	0.367	0.174	0.020	0.350	0.079	0.067	0
0.118	0.368	0.347	0.109	0.302	0.123	0.229	0.033	0



0.118	0.077	0.327	0.228	0.037	0.123	0.108	0.017	0.000
0.000	0.329	0.408	0.348	0.109	0.540	0.123	0.017	0.000
0.294	0.594	0.592	0.174	0.020	0.282	0.240	0.800	0.000
0.118	0.548	0.510	0.207	0.230	0.307	0.219	0.067	0.000
0.176	0.555	0.408	0.174	0.020	0.100	0.101	0.017	0.000
0.059	0.406	0.265	0.130	0.020	0.207	0.044	0.133	0.000
0.059	0.619	0.510	0.207	0.200	0.121	0.320	0.033	0.000
0.471	0.406	0.571	0.174	0.020	0.131	0.332	0.217	0.000
0.118	0.497	0.469	0.272	0.097	0.427	0.345	0.033	0.000
0.118	0.297	0.367	0.174	0.020	0.108	0.048	0.067	0.000
0.000	0.355	0.490	0.174	0.020	0.139	0.075	0.017	0.000
0.235	0.535	0.653	0.043	0.169	0.333	0.222	0.117	0.000
0.235	0.477	0.469	0.174	0.020	0.538	0.353	0.083	0.000
0.118	0.503	0.531	0.217	0.224	0.362	0.173	0.083	0.000
0.118	0.548	0.490	0.174	0.020	0.415	0.096	0.333	0.000
0.235	0.426	0.531	0.141	0.103	0.209	0.017	0.100	0.000
0.353	0.232	0.571	0.315	0.020	0.442	0.042	0.117	0.000
0.118	0.535	0.224	0.152	0.386	0.331	0.042	0.017	0.000
0.176	0.735	0.408	0.065	0.448	0.266	0.093	0.050	0.000
0.294	0.529	0.551	0.217	0.010	0.233	0.154	0.317	0.000
0.000	0.581	0.347	0.141	0.333	0.168	0.117	0.000	0.000
0.176	0.374	0.510	0.174	0.020	0.231	0.018	0.183	0.000
0.588	0.323	0.490	0.120	0.020	0.100	0.221	0.583	0.000
0.059	0.413	0.367	0.424	0.197	0.354	0.144	0.050	0.000
0.235	0.252	0.633	0.130	0.020	0.227	0.102	0.217	0.000
0.059	0.452	0.429	0.315	0.224	0.407	0.090	0.000	0.000
0.294	0.471	0.633	0.250	0.109	0.427	0.074	0.350	0.000
0.059	0.432	0.714	0.174	0.020	0.299	0.080	0.400	0.000
0.235	0.439	0.551	0.359	0.020	0.434	0.067	0.283	0.000
0.059	0.465	0.551	0.239	0.200	0.366	0.178	0.067	0.000
0.000	0.626	0.612	0.207	0.020	0.290	0.152	0.017	0.000
0.118	0.845	0.653	0.174	0.020	0.096	0.106	0.017	0.000
0.118	0.310	0.286	0.174	0.020	0.243	0.027	0.017	0.000
0.118	0.400	0.327	0.217	0.181	0.221	0.149	0.017	0.000
0.118	0.394	0.520	0.174	0.020	0.104	0.206	0.533	0.000
0.235	0.329	0.367	0.272	0.020	0.352	0.088	0.117	0.000
0.000	0.529	0.633	0.217	0.127	0.188	0.187	0.000	0.000
0.471	0.135	0.490	0.174	0.020	0.282	0.223	0.350	0.000
0.118	0.355	0.367	0.109	0.175	0.376	0.160	0.000	0.000
0.176	0.374	0.204	0.141	0.096	0.258	0.137	0.083	0.000
0.059	0.419	0.347	0.120	0.123	0.211	0.060	0.017	0.000
0.765	0.703	0.653	0.326	0.151	0.458	0.468	0.300	0.000
0.706	0.361	0.612	0.283	0.109	0.241	0.175	0.417	0.000



0.059	0.239	0.510	0.370	0.052	0.575	0.435	0.183	0
0.059	0.497	0.551	0.348	0.072	0.425	0.078	0.117	0
0.176	0.413	0.388	0.185	0.020	0.160	0.062	0.067	0
0.412	0.600	0.673	0.370	0.020	0.282	0.134	0.300	0
0.059	0.400	0.531	0.174	0.020	0.395	0.051	0.083	0
0.118	0.284	0.347	0.207	0.002	0.209	0.294	0.017	0
0.529	0.290	0.388	0.174	0.020	0.088	0.027	0.200	0
0.588	0.368	0.531	0.446	0.200	0.301	0.040	0.700	0
0.118	0.503	0.469	0.217	0.020	0.380	0.112	0.100	0
0.294	0.497	0.490	0.174	0.118	0.164	0.071	0.150	0
0.059	0.316	0.469	0.261	0.020	0.249	0.101	0.033	0
0.412	0.884	0.612	0.152	0.214	0.362	0.217	0.500	1
0.000	0.871	0.673	0.217	0.020	0.530	0.260	0.033	1
0.529	0.774	0.612	0.152	0.020	0.258	0.322	0.183	1
0.176	0.613	0.306	0.174	0.020	0.151	0.138	0.017	1
0.353	0.484	0.265	0.163	0.195	0.182	0.529	0.200	1
0.529	0.903	0.622	0.087	0.020	0.241	0.485	0.467	1
0.706	0.310	0.388	0.000	0.293	0.192	0.362	0.383	1
0.059	0.445	0.408	0.304	0.020	0.315	0.199	0.000	1
0.647	0.716	0.531	0.228	0.163	0.309	0.544	0.500	1
0.176	0.626	0.490	0.174	0.020	0.241	0.292	0.100	1
0.235	0.510	0.388	0.174	0.020	0.282	0.063	0.233	1
0.000	0.606	0.490	0.174	0.020	0.370	0.365	0.067	1
0.118	0.658	0.490	0.174	0.020	0.190	0.069	0.117	1
0.588	0.368	0.633	0.326	0.020	0.560	0.452	0.283	1
0.412	0.400	0.367	0.185	0.020	0.170	0.093	0.133	1
0.000	0.658	0.469	0.174	0.020	0.403	0.109	0.117	1
0.412	0.755	0.633	0.174	0.020	0.249	0.037	0.433	1
0.118	0.413	0.571	0.174	0.020	0.180	0.077	0.517	1
0.059	0.484	0.633	0.348	0.248	0.560	0.312	0.133	1
0.000	0.406	0.388	0.250	0.072	0.376	0.290	0.067	1
0.118	0.542	0.551	0.326	0.202	0.513	0.489	0.167	1
0.059	0.542	0.245	0.413	0.216	0.456	0.228	0.050	1
0.118	0.658	0.469	0.337	0.416	0.200	0.111	0.133	1
0.824	0.361	0.551	0.196	0.204	0.376	0.143	0.417	1
0.000	0.794	0.490	0.174	0.020	0.288	0.325	0.150	1
0.118	0.645	0.347	0.283	0.145	0.274	0.147	0.067	1
0.294	0.458	0.755	0.174	0.020	0.710	0.056	0.117	1
0.588	0.755	0.449	0.174	0.142	0.149	0.106	0.433	1
0.000	0.542	0.449	0.130	0.200	0.252	0.561	0.067	1
0.118	0.516	0.449	0.228	0.230	0.301	0.340	0.150	1
0.118	0.716	0.510	0.109	0.099	0.172	0.152	0.100	1
0.412	0.419	0.571	0.261	0.020	0.362	0.448	0.367	1



0.176	0.890	0.510	0.174	0.020	0.252	0.114	0.133	1
0.353	0.968	0.551	0.174	0.020	0.108	0.022	0.633	1
0.176	0.439	0.510	0.250	0.020	0.274	0.051	0.067	1
0.000	0.516	0.469	0.141	0.020	0.188	0.075	0.250	1
0.765	0.697	0.673	0.283	0.018	0.176	0.279	0.367	1
0.059	0.503	0.408	0.272	0.171	0.346	0.262	0.150	1
0.118	0.374	0.633	0.315	0.127	0.558	0.021	0.033	1
0.059	0.877	0.490	0.174	0.020	0.513	0.087	0.333	1
0.294	0.458	0.531	0.174	0.020	0.266	0.113	0.383	1
0.529	0.697	0.551	0.293	0.189	0.327	0.348	0.200	1
0.412	0.865	0.612	0.174	0.020	0.444	0.108	0.333	1
0.294	0.471	0.571	0.272	0.020	0.466	0.114	0.267	1
0.529	0.781	0.653	0.174	0.020	0.249	0.096	0.467	1
0.059	0.523	0.265	0.359	0.184	0.309	0.377	0.117	1
0.765	0.548	0.490	0.250	0.020	0.444	0.210	0.383	1
0.059	0.981	0.531	0.315	0.282	0.374	0.340	0.133	1
0.294	0.935	0.408	0.283	0.374	0.266	0.216	0.133	1
0.235	0.658	0.551	0.174	0.020	0.415	0.189	0.767	1
0.353	0.516	0.490	0.174	0.020	0.192	0.124	0.133	1
0.059	0.574	0.796	0.228	0.151	0.299	0.067	0.400	1
0.176	0.832	0.592	0.446	0.542	0.413	0.879	0.067	1
0.706	0.619	0.592	0.391	0.374	0.429	0.192	0.617	1
0.235	0.723	0.520	0.174	0.020	0.616	0.068	0.183	1
0.294	0.465	0.510	0.239	0.020	0.288	0.249	0.233	1
0.471	0.394	0.776	0.315	0.020	0.513	0.069	0.400	1
0.294	0.645	0.592	0.207	0.326	0.282	0.160	0.617	1
0.294	0.787	0.531	0.174	0.020	0.562	0.112	0.100	1
0.235	0.735	0.551	0.174	0.020	0.301	0.310	0.167	1
0.000	0.561	0.429	0.359	0.020	0.329	0.050	0.017	1
0.176	0.961	0.469	0.261	0.020	0.342	0.070	0.067	1
0.235	0.329	0.408	0.174	0.020	0.282	0.035	0.167	1
0.294	0.594	0.612	0.370	0.089	0.344	0.089	0.233	1
0.294	0.800	0.408	0.174	0.020	0.301	0.024	0.333	1
0.235	0.458	0.490	0.174	0.020	0.219	0.127	0.417	1
0.471	0.987	0.510	0.174	0.020	0.157	0.475	0.300	1
0.059	0.826	0.449	0.457	0.679	0.495	0.266	0.117	1
0.000	0.606	0.367	0.304	0.184	0.335	0.195	0.000	1
0.176	0.832	0.612	0.283	0.553	0.358	0.077	0.017	1
0.235	0.645	0.592	0.272	0.020	0.415	0.203	0.267	1
0.176	0.548	0.408	0.239	0.121	0.168	0.060	0.117	1
0.471	0.690	0.551	0.272	0.236	0.505	0.187	0.250	1
0.235	0.903	0.551	0.348	0.316	0.384	0.079	0.167	1
0.059	0.884	0.408	0.250	0.200	0.325	0.107	0.283	1



0.059	0.329	0.592	0.196	0.200	0.344	0.066	0.367	1
0.000	0.626	0.490	0.174	0.020	0.495	0.054	0.133	1
0.000	0.935	0.816	0.196	0.020	0.329	0.152	0.333	1
0.471	0.413	0.469	0.174	0.020	0.252	0.374	0.200	1
0.235	0.471	0.388	0.054	0.020	0.235	0.129	0.150	1
0.000	0.877	0.551	0.609	0.000	0.843	1.000	0.067	1
0.000	0.387	0.408	0.326	0.060	0.315	0.184	0.017	1
0.118	0.581	0.469	0.174	0.020	0.219	0.198	0.033	1
0.824	0.845	0.388	0.250	0.020	0.315	0.057	0.283	1
0.588	0.671	0.612	0.446	0.268	0.397	0.394	0.500	1
0.471	0.490	0.490	0.174	0.020	0.241	0.045	0.283	1
0.118	0.394	0.571	0.413	0.213	0.317	0.270	0.133	1
0.176	0.735	0.469	0.250	0.377	0.354	0.114	0.233	1
0.000	0.652	0.490	0.174	0.020	0.532	0.236	0.167	1
0.000	0.587	0.449	0.380	0.284	0.493	0.123	0.050	1
0.235	0.523	0.469	0.120	0.130	0.219	0.455	0.400	1
0.412	0.974	0.469	0.283	0.157	0.141	0.036	0.567	1
0.353	0.471	0.449	0.370	0.020	0.425	0.277	0.333	1
0.000	0.877	0.673	0.207	0.091	0.374	0.101	0.233	1
0.706	0.258	0.490	0.261	0.020	0.235	0.094	0.417	1
0.176	0.768	0.469	0.120	0.109	0.274	0.081	0.117	1
0.529	0.652	0.653	0.293	0.181	0.247	0.296	0.533	1
0.529	0.555	0.469	0.174	0.020	0.327	0.245	0.400	1
0.235	0.568	0.490	0.174	0.020	0.301	0.096	0.033	1
0.176	0.548	0.694	0.457	0.169	0.372	0.380	0.183	1
0.471	0.361	0.510	0.359	0.242	0.434	0.249	0.367	1
0.176	0.542	0.490	0.196	0.212	0.290	0.201	0.100	1
0.588	0.297	0.622	0.272	0.020	0.342	0.319	0.583	1
0.471	0.916	0.673	0.304	0.254	0.333	0.147	0.267	1
0.294	0.923	0.531	0.217	0.232	0.519	0.408	0.533	1
0.353	0.523	0.531	0.174	0.020	0.319	0.018	0.550	1
0.000	0.994	0.429	0.272	0.313	0.472	0.181	0.117	1
0.000	0.497	0.429	0.250	0.181	0.329	0.053	0.200	1
0.118	0.477	0.571	0.174	0.020	0.505	0.263	0.000	1
0.118	0.987	0.469	1.000	0.020	0.337	0.212	0.683	1
0.000	0.690	0.673	0.424	0.020	0.489	0.125	0.000	1
0.471	0.516	0.531	0.185	0.704	0.215	0.260	0.517	1
0.471	0.639	0.429	0.174	0.020	0.342	0.022	0.333	1
0.176	0.852	0.633	0.217	0.171	0.309	0.459	0.517	1
0.647	0.432	0.612	0.359	0.020	0.585	0.362	0.400	1
0.176	0.568	0.571	0.174	0.020	0.331	0.138	0.383	1
0.000	0.929	0.592	0.076	0.206	0.282	0.258	0.017	1
0.059	0.832	0.510	0.174	0.020	0.380	0.004	0.283	1



0.412	0.684	0.551	0.239	0.135	0.348	0.262	0.550	1
0.235	0.897	0.490	0.174	0.020	0.209	0.057	0.250	1
0.059	0.884	0.551	0.380	0.335	0.446	0.504	0.017	1
0.176	0.839	0.347	0.163	0.216	0.301	0.220	0.250	1
0.412	0.800	0.653	0.380	0.369	0.409	0.303	0.317	1
0.647	0.606	0.510	0.207	0.156	0.366	0.205	0.483	1
0.529	0.439	0.592	0.185	0.020	0.204	0.514	0.483	1
0.000	0.484	0.490	0.174	0.020	0.290	0.027	0.050	1
0.412	0.452	0.408	0.174	0.020	0.188	0.279	0.217	1
0.765	0.387	0.490	0.174	0.020	0.266	0.165	0.283	1
0.412	0.342	0.531	0.272	0.093	0.464	0.339	0.183	1
0.353	0.665	0.571	0.174	0.020	0.231	0.043	0.483	1
0.059	0.794	0.510	0.109	0.156	0.106	0.158	0.200	1
0.000	0.871	0.265	0.315	0.174	0.401	0.161	0.017	1
0.647	0.594	0.612	0.304	0.139	0.207	0.078	0.350	1
0.118	0.716	0.286	0.217	0.632	0.419	0.069	0.067	1
0.176	0.232	0.592	0.261	0.067	0.327	0.518	0.100	1
0.059	1.000	0.531	0.391	0.020	0.505	0.562	0.017	1
0.471	0.794	0.837	0.424	0.261	0.397	0.037	0.367	1
0.529	0.652	0.571	0.424	0.139	0.403	0.239	0.317	1
0.353	0.458	0.367	0.348	0.020	0.317	0.071	0.317	1
0.235	0.652	0.592	0.120	0.020	0.292	0.067	0.817	1
0.588	0.432	0.469	0.217	0.020	0.190	0.027	0.317	1
0.353	0.974	0.469	0.174	0.020	0.260	0.107	0.167	1
0.529	0.723	0.633	0.174	0.020	0.135	0.065	0.533	1
0.176	0.497	0.286	0.174	0.020	0.364	0.021	0.067	1
0.000	0.761	0.531	0.315	0.020	0.642	0.122	0.083	1
0.235	0.523	0.571	0.174	0.020	0.288	0.196	0.100	1
0.059	0.645	0.592	0.424	0.200	0.571	0.110	0.417	1
0.765	0.735	0.918	0.174	0.020	0.493	0.076	0.383	1
0.412	0.548	0.449	0.457	0.133	0.415	0.154	0.367	1
0.412	0.632	0.673	0.185	0.560	0.249	0.021	0.367	1
0.176	0.806	0.510	0.130	0.133	0.239	0.081	0.167	1
0.353	0.523	0.551	0.261	0.020	0.192	0.208	0.467	1
0.059	0.800	0.653	0.239	0.020	0.344	0.353	0.517	1
0.588	0.458	0.490	0.174	0.020	0.282	0.078	0.150	1
0.529	0.774	0.551	0.174	0.020	0.299	0.030	0.400	1
0.118	0.316	0.408	0.272	0.175	0.405	0.254	0.033	1
0.588	0.548	0.388	0.315	0.020	0.470	0.155	0.283	1
0.412	0.923	0.265	0.283	0.454	0.321	0.319	0.217	1
0.176	0.832	0.551	0.348	0.206	0.319	0.381	0.167	1
0.294	0.342	0.531	0.217	0.020	0.356	0.128	0.517	1
0.059	0.677	0.449	0.239	0.136	0.227	0.116	0.350	1



0.176	0.555	0.551	0.174	0.078	0.209	0.105	0.217	1
0.471	0.490	0.633	0.174	0.020	0.209	0.077	0.017	1
0.118	0.839	0.653	0.326	0.127	0.538	0.243	0.050	1
0.059	0.374	0.510	0.174	0.020	0.436	0.092	0.350	1
0.647	0.490	0.571	0.326	0.163	0.493	0.302	0.450	1
0.529	0.619	0.714	0.174	0.020	0.297	0.280	0.400	1
0.059	0.665	0.714	0.370	0.020	0.636	0.120	0.100	1
0.176	0.923	0.469	0.163	0.224	0.372	0.141	0.250	1
0.353	0.761	0.388	0.174	0.020	0.125	0.043	0.483	1
0.235	0.594	0.469	0.174	0.020	0.266	0.471	0.017	1
0.000	0.884	0.653	0.402	0.596	0.513	0.061	0.083	1
0.471	0.710	0.551	0.272	0.020	0.290	0.156	0.400	1
0.059	0.542	0.653	0.348	0.115	0.374	0.418	0.267	1
0.000	0.510	0.490	0.174	0.020	0.370	0.077	0.517	1
0.353	0.942	0.694	0.174	0.020	0.354	0.085	0.750	1
0.529	0.813	0.510	0.261	0.020	0.528	0.139	0.367	1
0.059	0.529	0.367	0.174	0.020	0.243	0.116	0.433	1



LAMPIRAN B DATA UJI

Jumlah Kehamilan	Glukosa	Tekanan Darah	Ketebalan Kulit	Insulin	Indeks Massa Tubuh	Diabetes Pedigree	Usia	Keluaran
0.059	0.265	0.429	0.239	0.020	0.172	0.117	0.167	0
0.059	0.290	0.429	0.174	0.096	0.202	0.038	0.000	0
0.294	0.465	0.510	0.174	0.020	0.151	0.053	0.150	0
0.588	0.458	0.490	0.174	0.020	0.350	0.024	0.133	0
0.235	0.426	0.694	0.174	0.020	0.397	0.048	0.150	0
0.588	0.613	0.571	0.174	0.020	0.182	0.582	0.600	0
0.059	0.381	0.061	0.337	0.083	0.513	0.045	0.200	0
0.176	0.529	0.653	0.370	0.266	0.431	0.267	0.100	0
0.471	0.355	0.612	0.174	0.020	0.352	0.132	0.483	0
0.059	0.342	0.429	0.087	0.151	0.102	0.175	0.017	0
0.765	0.652	0.592	0.130	0.115	0.082	0.071	0.600	0
0.294	0.471	0.694	0.174	0.020	0.325	0.111	0.283	0
0.294	0.419	0.520	0.207	0.020	0.364	0.200	0.650	0
0.176	0.284	0.347	0.043	0.048	0.135	0.081	0.017	0
0.353	0.310	0.694	0.174	0.020	0.035	0.047	0.117	0
0.588	0.503	0.551	0.261	0.020	0.192	0.185	0.400	0
0.235	0.381	0.367	0.283	0.214	0.119	0.379	0.200	0
0.647	0.606	0.531	0.174	0.020	0.307	0.146	0.233	0
0.176	0.877	0.408	0.196	0.067	0.323	0.082	0.083	0
0.412	0.574	0.612	0.174	0.020	0.450	0.264	0.267	0
0.412	0.400	0.694	0.120	0.020	0.092	0.067	0.450	0
0.412	0.742	0.408	0.174	0.020	0.188	0.092	0.317	0
0.059	0.658	0.327	0.174	0.020	0.235	0.208	0.133	0
0.118	0.174	0.469	0.217	0.020	0.200	0.217	0.017	0
0.412	0.394	0.490	0.174	0.020	0.282	0.097	0.050	0
0.059	0.381	0.571	0.043	0.082	0.025	0.176	0.017	0
0.059	0.368	0.265	0.087	0.026	0.123	0.191	0.083	0
0.294	0.284	0.429	0.152	0.011	0.127	0.113	0.150	0
0.412	0.684	0.429	0.380	0.394	0.337	0.273	0.350	0
0.059	0.187	0.265	0.033	0.020	0.098	0.073	0.000	0
0.000	0.361	0.653	0.576	0.115	0.585	0.377	0.167	0
0.000	0.658	0.592	0.174	0.020	0.456	0.727	0.383	0
0.000	0.394	0.408	0.370	0.154	0.476	0.041	0.017	0



0.118	0.258	0.490	0.174	0.020	0.282	0.096	0.000	0
0.294	0.000	0.388	0.174	0.020	0.139	0.217	0.250	0
0.118	0.626	0.347	0.293	0.137	0.147	0.265	0.050	0
0.294	0.355	0.510	0.217	0.020	0.221	0.053	0.183	0
0.118	0.419	0.694	0.174	0.020	0.501	0.327	0.550	0
0.059	0.329	0.429	0.065	0.029	0.029	0.109	0.067	0
0.235	0.658	0.622	0.217	0.103	0.219	0.047	0.100	0
0.294	0.613	0.408	0.304	0.151	0.213	0.142	0.083	0
0.235	0.548	0.633	0.141	0.308	0.346	0.065	0.033	0
0.059	0.226	0.520	0.250	0.020	0.282	0.136	0.017	0
0.059	0.471	0.245	0.141	0.020	0.133	0.026	0.017	0
0.412	0.116	0.551	0.174	0.020	0.294	0.134	0.333	0
0.294	0.329	0.490	0.283	0.020	0.399	0.125	0.100	0
0.118	0.439	0.429	0.163	0.020	0.139	0.098	0.050	0
0.176	0.445	0.204	0.065	0.020	0.086	0.026	0.017	0
0.118	0.194	0.490	0.174	0.020	0.282	0.010	0.017	0
0.412	0.252	0.551	0.207	0.069	0.227	0.294	0.250	0
0.000	0.368	0.418	0.228	0.020	0.131	0.068	0.017	0
0.118	0.426	0.510	0.239	0.133	0.290	0.265	0.100	0
0.765	0.400	0.490	0.511	0.020	0.376	0.043	0.400	0
0.118	0.361	0.449	0.196	0.069	0.415	0.105	0.083	0
0.059	0.406	0.449	0.130	0.020	0.170	0.037	0.050	0
0.059	0.232	0.316	0.174	0.020	0.018	0.077	0.000	0
0.235	0.510	0.571	0.087	0.195	0.282	0.156	0.217	0
0.412	0.239	0.551	0.359	0.041	0.583	0.078	0.350	0
0.118	0.632	0.592	0.120	0.060	0.133	0.292	0.000	0
0.353	0.645	0.490	0.217	0.257	0.321	0.076	0.317	0
0.118	0.310	0.388	0.228	0.020	0.274	0.022	0.050	0
0.059	0.174	0.245	0.120	0.075	0.045	0.105	0.017	0
0.353	0.316	0.265	0.250	0.060	0.215	0.119	0.033	0
0.059	0.690	0.367	0.174	0.020	0.162	0.043	0.017	0
0.000	0.523	0.735	0.174	0.020	0.088	0.079	0.000	0
0.059	0.239	0.490	0.120	0.031	0.172	0.088	0.050	0
0.118	0.265	0.418	0.174	0.020	0.438	0.364	0.100	0
0.059	0.529	0.327	0.239	0.166	0.215	0.309	0.000	0
0.059	0.335	1.000	0.174	0.020	0.086	0.055	0.100	0
0.235	0.645	0.347	0.228	0.151	0.231	0.089	0.267	0
0.176	0.252	0.347	0.261	0.005	0.329	0.110	0.067	0



0.059	0.290	0.531	0.293	0.028	0.266	0.049	0.033	0
0.235	0.206	0.388	0.174	0.020	0.323	0.134	0.067	0
0.294	0.219	0.245	0.174	0.020	0.317	0.246	0.067	0
0.235	0.342	0.367	0.174	0.020	0.204	0.156	0.017	0
0.235	0.355	0.531	0.087	0.044	0.102	0.062	0.000	0
0.353	0.432	0.408	0.348	0.020	0.327	0.078	0.050	0
0.118	0.406	0.510	0.250	0.103	0.315	0.139	0.033	0
0.294	0.568	0.571	0.174	0.020	0.176	0.046	0.800	0
0.176	0.490	0.469	0.250	0.145	0.505	0.160	0.150	0
0.059	0.477	0.347	0.315	0.096	0.309	0.078	0.033	0
0.471	0.258	0.510	0.261	0.020	0.411	0.162	0.300	0
0.118	0.335	0.449	0.065	0.042	0.059	0.243	0.083	0
0.118	0.523	0.367	0.141	0.151	0.319	0.004	0.167	0
0.000	0.361	0.469	0.207	0.043	0.258	0.222	0.000	0
0.000	0.316	0.367	0.196	0.094	0.215	0.194	0.017	0
0.000	0.548	0.571	0.174	0.020	0.266	0.267	0.133	0
0.294	0.394	0.490	0.239	0.374	0.382	0.035	0.117	0
0.176	0.542	0.551	0.174	0.020	0.059	0.081	0.567	0
0.294	0.400	0.592	0.250	0.020	0.436	0.089	0.283	0
0.353	0.671	0.490	0.304	0.020	0.315	0.234	0.483	1
0.471	0.897	0.408	0.174	0.020	0.104	0.254	0.183	1
0.000	0.600	0.163	0.304	0.185	0.509	0.944	0.200	1
0.176	0.219	0.265	0.272	0.089	0.262	0.073	0.083	1
0.118	0.987	0.469	0.413	0.636	0.252	0.034	0.533	1
0.471	0.523	0.735	0.174	0.020	0.282	0.066	0.550	1
0.588	0.800	0.510	0.174	0.020	0.405	0.196	0.217	1
0.059	0.935	0.367	0.174	1.000	0.243	0.137	0.633	1
0.294	0.787	0.490	0.130	0.194	0.155	0.217	0.500	1
0.412	0.361	0.490	0.174	0.020	0.241	0.173	0.183	1
0.000	0.477	0.612	0.435	0.260	0.564	0.202	0.167	1
0.412	0.406	0.510	0.174	0.020	0.233	0.075	0.167	1
0.059	0.458	0.469	0.250	0.099	0.335	0.193	0.183	1
0.412	0.981	0.673	0.174	0.020	0.442	0.159	0.333	1
0.529	0.484	0.571	0.304	0.020	0.221	0.079	0.133	1
0.647	0.639	0.714	0.283	0.159	0.376	0.075	0.500	1
0.588	0.523	0.469	0.207	0.121	0.264	0.054	0.333	1
0.412	0.665	0.531	0.174	0.020	0.434	0.076	0.367	1
0.176	0.735	0.531	0.315	0.278	0.274	0.330	0.117	1



0.529	0.374	0.531	0.326	0.020	0.301	0.251	0.417	1
0.118	0.297	0.449	0.380	0.020	0.409	0.181	0.100	1
0.235	0.432	0.490	0.435	0.232	0.387	0.560	0.583	1
0.529	0.819	0.878	0.185	0.272	0.556	0.275	0.550	1
0.000	0.877	0.429	0.348	0.020	0.487	0.775	0.067	1
0.412	0.381	0.429	0.272	0.020	0.427	0.114	0.167	1
0.471	0.852	0.673	0.293	0.344	0.317	0.166	0.617	1
0.412	0.923	0.449	0.348	0.349	0.399	0.075	0.333	1
0.471	0.574	0.490	0.174	0.020	0.301	0.082	0.300	1
0.412	0.452	0.429	0.174	0.020	0.299	0.077	0.350	1
0.000	0.419	0.653	0.250	0.020	0.292	0.332	0.283	1
0.118	0.361	0.429	0.141	0.091	0.301	0.337	0.117	1
0.765	0.529	0.673	0.174	0.020	0.515	0.216	0.350	1
0.000	0.561	0.490	0.174	0.020	0.511	0.082	0.083	1
0.294	0.600	0.857	0.174	0.020	0.626	0.064	0.267	1
0.882	0.594	0.469	0.272	0.115	0.387	0.032	0.367	1
0.235	0.581	0.490	0.174	0.020	0.115	0.085	0.650	1
0.059	0.503	0.673	0.478	0.248	0.644	0.105	0.167	1
0.059	0.768	0.490	0.174	0.020	0.425	0.488	0.200	1
0.000	0.329	0.622	0.196	0.026	0.393	0.072	0.050	1
0.176	0.819	0.490	0.283	0.145	0.309	0.052	0.050	1
0.471	0.716	0.388	0.207	0.578	0.323	0.199	0.417	1
0.412	0.748	0.306	0.272	0.194	0.252	0.218	0.300	1
0.235	0.658	0.694	0.174	0.020	0.266	0.197	0.667	1
0.294	0.516	0.510	0.174	0.020	0.323	0.061	0.283	1
0.000	0.761	0.531	0.533	0.103	0.716	0.291	0.067	1
0.000	0.445	0.531	0.174	0.020	0.309	0.085	0.033	1
0.059	0.284	0.061	0.380	0.102	0.753	0.178	0.083	1
0.059	0.471	0.653	0.185	0.157	0.333	0.139	0.317	1
0.000	0.394	0.612	0.174	0.020	0.198	0.283	0.683	1
0.235	0.832	0.469	0.076	0.185	0.235	0.121	0.200	1
0.529	0.503	0.327	0.174	0.020	0.309	0.442	0.200	1
0.176	0.813	0.408	0.326	0.254	0.333	0.119	0.150	1
0.588	0.413	0.429	0.174	0.020	0.290	0.083	0.350	1
0.529	0.723	0.633	0.228	0.169	0.329	0.474	0.350	1
0.471	0.929	0.551	0.174	0.020	0.607	0.025	0.367	1
0.412	0.697	0.653	0.402	0.020	0.650	0.111	0.250	1
1.000	0.768	0.490	0.370	0.120	0.464	0.316	0.433	1



0.000	0.561	0.653	0.174	0.020	0.274	0.284	0.183	1
0.353	0.387	0.510	0.120	0.171	0.239	0.275	0.333	1
0.353	0.374	0.592	0.174	0.020	0.258	0.044	0.250	1
0.353	0.581	0.469	0.174	0.139	0.352	0.198	0.133	1
0.471	0.871	0.490	0.380	0.139	0.297	0.274	0.250	1
0.000	0.548	0.878	0.424	0.139	1.000	0.103	0.083	1
0.294	0.555	0.592	0.174	0.020	0.427	0.375	0.267	1
0.412	0.968	0.449	0.228	0.020	0.362	0.285	0.333	1
0.471	0.884	0.449	0.315	0.578	0.243	0.229	0.650	1
0.059	0.542	0.755	0.370	0.053	0.282	0.531	0.200	1
0.471	0.419	0.531	0.348	0.120	0.198	0.240	0.167	1
0.294	0.613	0.571	0.304	0.175	0.274	0.121	0.067	1
0.412	0.742	0.429	0.174	0.020	0.249	0.130	0.250	1
0.647	0.587	0.490	0.174	0.020	0.697	0.213	0.317	1
0.294	0.735	0.612	0.370	0.236	0.434	0.135	0.133	1
0.176	0.406	0.388	0.065	0.041	0.096	0.256	0.033	1
0.235	0.419	0.408	0.402	0.102	0.339	0.353	0.083	1
0.235	0.671	0.367	0.217	0.365	0.260	0.031	0.133	1
0.471	0.981	0.531	0.239	0.320	0.395	0.225	0.600	1
0.294	0.761	0.816	0.174	0.020	0.399	0.031	0.517	1
0.412	0.903	0.612	0.283	0.020	0.354	0.118	0.333	1
0.000	0.619	0.418	0.207	0.139	0.499	0.151	0.050	1
0.529	0.439	0.592	0.272	0.194	0.327	0.078	0.250	1
0.706	0.690	0.469	0.359	0.309	0.483	0.284	0.283	1
0.294	0.419	0.388	0.370	0.138	0.360	0.186	0.067	1
0.294	0.265	0.510	0.163	0.020	0.221	0.489	0.183	1
0.294	0.439	0.429	0.174	0.020	0.401	0.078	0.333	1
0.000	0.858	0.367	0.239	0.558	0.335	0.424	0.000	1
0.118	0.735	0.673	0.174	0.020	0.274	0.310	0.750	1
0.176	0.761	0.286	0.337	0.020	0.389	0.245	0.050	1
0.235	0.632	0.633	0.174	0.020	0.528	0.242	0.017	1
0.353	0.581	0.571	0.326	0.428	0.573	0.068	0.417	1
0.235	0.819	0.490	0.174	0.020	0.519	0.171	0.083	1



LAMPIRAN C POHON KEPUTUSAN

Berikut adalah pohon keputusan dalam notasi *s-expression* yang dihasilkan saat pengujian validasi pada Subbab 6.6.

(((< GL 0.6365285) ((< IN 0.9347515) ((< DP 0.874076) ((< ST 0.074047446) ((< GL 0.7842332) ((< DP 0.19683552) ((< GL 0.63926816) ((< BM 0.32779098) ((< DP 0.21806455) ((< BP 0.75144804) ((< PR 0.77879035) ((< ST 0.48243833) TD ((< ST 0.49998248) ((< PR 0.33959663) DB DB) ((< AG 0.11952233) TD DB))) DB) DB) ((< BP 0.89267313) ((< BP 0.68955433) TD TD) TD)) DB) ((< BM 0.5891324) ((< ST 0.4587903) DB TD) TD)) ((< AG 0.26613235) TD ((< BM 0.32779098) ((< DP 0.21806455) ((< GL 0.4133004) ((< GL 0.63926816) TD DB) ((< DP 0.40439844) TD DB)) DB) DB))) DB) ((< GL 0.7842332) ((< DP 0.19683552) ((< GL 0.63926816) TD ((< GL 0.36106253) TD ((< AG 0.9509016) ((< ST 0.5382546) TD TD) ((< DP 0.10168934) ((< BP 0.08690882) TD ((< IN 0.8183197) DB DB) TD)))) ((< AG 0.26613235) ((< DP 0.21806455) ((< GL 0.4133004) TD DB) ((< DP 0.874076) ((< ST 0.074047446) ((< GL 0.7842332) ((< GL 0.63926816) TD ((< BM 0.5891324) ((< GL 0.36106253) TD TD) ((< ST 0.5382546) TD TD))) DB) ((< GL 0.7842332) ((< GL 0.4133004) ((< GL 0.63926816) TD ((< BM 0.5891324) ((< GL 0.36106253) TD TD) ((< ST 0.5382546) TD TD))) ((< DP 0.40439844) ((< BM 0.32779098) ((< AG 0.10482597) ((< IN 0.074620605) DB TD) ((< DP 0.10168934) ((< BP 0.27567077) TD ((< GL 0.36106253) DB ((< AG 0.9509016) ((< ST 0.5382546) TD TD) ((< DP 0.10168934) ((< BP 0.08690882) TD DB) TD)))) ((< IN 0.3305062) ((< ST 0.3880787) ((< BP 0.5962409) DB TD) ((< PR 0.87880456) TD DB))





((< DP 0.10168934) ((< BP 0.08690882) TD DB) DB))
 ((< BM 0.5891324) ((< ST 0.4587903) DB TD) TD))
 ((< AG 0.7729366) ((< DP 0.57384586) DB TD) DB))
 DB)) DB)) ((< AG 0.26613235)
 ((< DP 0.21806455) ((< GL 0.4133004) ((< PR 0.8104538) TD DB) DB)
 ((< DP 0.19683552) ((< GL 0.63926816)
 ((< BM 0.32779098) ((< DP 0.21806455) ((< GL 0.4133004) TD
 ((< ST 0.05649054) TD ((< BP 0.86402583) TD TD))) DB) DB)
 ((< BM 0.32779098) ((< DP 0.21806455) ((< DP 0.21806455) TD DB) DB)
 DB))
 ((< AG 0.26613235) ((< DP 0.21806455) ((< GL 0.4133004) TD DB) TD)
 ((< BM 0.32779098) DB
 ((< DP 0.21806455) ((< GL 0.4133004) TD ((< BP 0.5962409) DB TD))
 ((< BP 0.89267313) DB DB)))))
 ((< BM 0.32779098) ((< DP 0.21806455) ((< GL 0.4133004) TD
 ((< BP 0.5962409) ((< IN 0.3968358)
 ((< IN 0.58919275) ((< AG 0.53647673) DB TD)
 ((< BP 0.08690882) TD TD)) ((< DP 0.874076)
 ((< AG 0.26613235) TD
 ((< BM 0.32779098) ((< BM 0.81149054) DB TD)
 ((< BM 0.32779098) TD ((< IN 0.6059842) TD
 ((< BP 0.5962409) ((< IN 0.3968358) ((< IN 0.58919275)
 ((< AG 0.53647673) DB ((< IN 0.8183197) DB DB))
 ((< BP 0.08690882) TD TD)) ((< DP 0.874076)
 ((< AG 0.26613235) TD
 ((< BM 0.32779098) ((< BM 0.81149054) DB TD)
 ((< BM 0.32779098) TD
 ((< IN 0.6059842) TD ((< BP 0.86402583) TD TD)))))
 ((< IN 0.8183197)
 ((< AG 0.5498977) ((< ST 0.089791775) TD TD)
 ((< DP 0.2315172) TD DB)) DB)) TD)))))
 ((< IN 0.8183197) ((< AG 0.5498977) ((< ST 0.089791775) TD TD)



(((< DP 0.2315172) TD DB)) DB))) TD)))
 ((< BP 0.89267313) ((< BM 0.32513213)
 ((< BP 0.68955433) ((< DP 0.5493666) ((< DP 0.39809692) TD DB) TD)
 DB) ((< GL 0.4133004) TD ((< DP 0.874076)
 ((< AG 0.26613235) TD
 ((< BM 0.32779098) ((< DP 0.21806455) TD DB) DB))
 ((< BM 0.32779098) DB DB)))
 ((< AG 0.26613235) ((< IN 0.715909) DB DB)
 ((< BM 0.32779098) ((< DP 0.21806455) TD DB)
 ((< BM 0.32779098) ((< DP 0.21806455)
 ((< GL 0.4133004) ((< ST 0.074047446) TD ((< GL 0.7842332)
 ((< DP 0.19683552) ((< GL 0.63926816) ((< BM 0.32779098)
 ((< DP 0.21806455) ((< BP 0.75144804)
 ((< PR 0.77879035) ((< ST 0.48243833) TD DB)
 ((< GL 0.4133004) TD ((< BP 0.27567077)
 ((< BP 0.45723403) TD
 ((< PR 0.77879035) ((< ST 0.48243833) TD DB)
 ((< GL 0.4133004) TD ((< BP 0.27567077) TD
 ((< GL 0.36106253) DB ((< AG 0.9509016)
 ((< IN 0.6059842) TD ((< DP 0.21806455)
 ((< GL 0.4133004) TD ((< ST 0.05649054) TD
 ((< BP 0.86402583) TD TD))) DB)) TD))))))
 ((< GL 0.36106253) DB
 ((< AG 0.9509016) ((< IN 0.6059842) TD TD)
 ((< ST 0.15426922) DB TD)))))) DB)
 ((< BP 0.89267313) TD TD)) DB)
 ((< BM 0.5891324) ((< ST 0.4587903) DB TD)
 ((< ST 0.5382546) ((< ST 0.09613919) TD
 ((< BM 0.32513213) ((< BP 0.68955433) TD DB)
 ((< IN 0.3968358) ((< IN 0.58919275)
 ((< PR 0.50937426) ((< IN 0.074620605) DB TD) DB)
 ((< DP 0.874076) ((< AG 0.26613235) TD



((< BM 0.32779098) TD ((< BM 0.32779098)
 ((< DP 0.21806455)
 ((< GL 0.4133004) TD ((< ST 0.4587903) DB TD)) DB) DB)))
 ((< IN 0.8183197) DB DB)))
 ((< DP 0.0028336048) ((< GL 0.3227681) DB TD)
 ((< PR 0.8104538) TD DB))))))
 ((< BP 0.9612719) ((< AG 0.46263874)
 ((< GL 0.36106253) ((< BP 0.89267313) DB TD) TD)
 ((< ST 0.09613919) ((< BP 0.4327854)
 ((< AG 0.26613235)
 ((< DP 0.21806455) ((< GL 0.4133004) TD DB)
 ((< DP 0.21806455) ((< GL 0.4133004) TD
 ((< ST 0.05649054) TD ((< BP 0.86402583) TD TD))) DB))
 ((< BM 0.32779098) ((< DP 0.21806455)
 ((< GL 0.4133004) TD
 ((< BP 0.15979922) ((< IN 0.16256142) TD DB)
 ((< BM 0.09794402) DB TD)))
 ((< BP 0.38068795) ((< BM 0.81149054) DB TD)
 ((< BP 0.49345243) TD DB))) DB))
 ((< BM 0.32779098) ((< DP 0.21806455)
 ((< GL 0.4133004) TD ((< BP 0.5962409) ((< IN 0.3968358)
 ((< IN 0.58919275) ((< AG 0.53647673) DB TD)
 ((< BP 0.08690882) ((< ST 0.4587903) DB TD) DB))
 ((< DP 0.874076) ((< AG 0.26613235) TD
 ((< BM 0.32779098) TD ((< BM 0.32779098)
 ((< DP 0.21806455) ((< GL 0.4133004) TD DB) DB) DB))
 DB)) TD)) ((< BP 0.89267313)
 ((< BM 0.32513213) ((< BP 0.68955433) TD DB)
 ((< GL 0.4133004) TD ((< DP 0.874076)
 ((< AG 0.26613235) TD
 ((< BM 0.32779098) ((< DP 0.21806455) TD DB) DB)) TD)))
 TD)) ((< DP 0.21806455)



((< GL 0.4133004) TD ((< AG 0.26613235) TD
 ((< BM 0.32779098) TD ((< BM 0.32779098) DB DB))))
 ((< BP 0.89267313) ((< BM 0.32779098) DB
 ((< BM 0.32779098)
 ((< DP 0.21806455) ((< GL 0.4133004) TD DB) DB) DB))
 DB)))) TD)) ((< DP 0.31640816) ((< BP 0.75144804)
 ((< PR 0.77879035) ((< ST 0.48243833) TD DB)
 ((< GL 0.4133004) TD
 ((< BP 0.27567077) ((< GL 0.4133004) TD DB)
 ((< GL 0.36106253) DB
 ((< AG 0.9509016) ((< ST 0.15426922) DB TD) TD))))
 DB) TD)))) ((< AG 0.26613235) ((< ST 0.15426922) DB
 ((< IN 0.3968358) ((< IN 0.58919275) ((< PR 0.50937426)
 ((< IN 0.074620605) ((< BM 0.32779098) ((< DP 0.21806455)
 ((< IN 0.3305062)
 ((< ST 0.3880787) DB ((< PR 0.87880456) TD TD)) TD) DB) DB)
 TD) ((< AG 0.3522389) DB DB)) ((< DP 0.874076)
 ((< AG 0.26613235) TD ((< BM 0.32779098) TD
 ((< BM 0.32779098) ((< DP 0.21806455)
 ((< GL 0.4133004) TD ((< ST 0.4587903) DB TD)) DB) DB)))
 ((< IN 0.8183197) DB DB)))
 ((< DP 0.0028336048) ((< GL 0.3227681) DB TD)
 ((< PR 0.8104538) TD DB)))) ((< BM 0.32779098)
 ((< DP 0.874076) ((< AG 0.26613235) TD
 ((< BM 0.32779098) TD ((< BM 0.32779098)
 ((< DP 0.21806455)
 ((< GL 0.4133004) TD ((< ST 0.4587903) DB TD)) DB) DB)))
 ((< IN 0.8183197) DB DB)) DB))) DB))
 ((< ST 0.05649054) TD TD)) DB) DB))))
 ((< DP 0.21806455) ((< GL 0.4133004) TD DB) ((< BP 0.89267313)
 ((< IN 0.3305062) ((< ST 0.3880787) DB ((< PR 0.87880456) TD TD))
 TD) DB)))))) DB) DB) ((< AG 0.46263874) ((< GL 0.36106253) DB



((< AG 0.9509016) ((< ST 0.5382546) TD TD))
 ((< DP 0.10168934) ((< AG 0.80725443) TD DB) TD))) DB))
 ((< AG 0.6838579) ((< ST 0.074047446) TD ((< GL 0.7842332)
 ((< DP 0.19683552) ((< GL 0.63926816) ((< BM 0.32779098)
 ((< DP 0.21806455)
 ((< BP 0.75144804) ((< PR 0.77879035) ((< ST 0.05649054) TD TD) TD)
 DB) ((< BP 0.89267313) TD TD)) DB)
 ((< BM 0.5891324) ((< ST 0.4587903) DB TD)
 ((< ST 0.5382546) ((< ST 0.09613919) TD
 ((< BM 0.32513213) ((< BP 0.68955433) TD DB) ((< IN 0.3968358)
 ((< IN 0.58919275) ((< PR 0.50937426) ((< IN 0.074620605) DB TD)
 ((< AG 0.3522389) DB DB))
 ((< DP 0.874076) ((< AG 0.26613235) TD ((< DP 0.21806455) TD DB))
 ((< IN 0.8183197) DB DB)))
 ((< DP 0.0028336048) ((< GL 0.3227681) DB TD) TD)))
 ((< BP 0.9612719) TD ((< DP 0.31640816)
 ((< BP 0.75144804) ((< PR 0.77879035) ((< ST 0.48243833) TD DB)
 ((< GL 0.4133004) TD ((< BP 0.27567077) ((< GL 0.4133004) TD DB)
 ((< GL 0.36106253) DB ((< IN 0.8183197) DB DB)))))) DB) TD))))
 ((< AG 0.26613235) ((< ST 0.15426922) DB ((< IN 0.3968358)
 ((< IN 0.58919275) ((< PR 0.50937426) ((< IN 0.074620605)
 ((< BM 0.32779098)
 ((< DP 0.21806455) ((< GL 0.4133004) TD ((< ST 0.4587903) DB TD))
 DB) ((< IN 0.7014077) ((< ST 0.31199372) DB TD)
 ((< BP 0.9676484) DB TD))) TD) ((< AG 0.3522389) DB DB))
 ((< DP 0.874076) ((< AG 0.26613235) TD ((< BM 0.32779098) TD
 ((< BM 0.32779098) ((< DP 0.21806455)
 ((< GL 0.4133004) ((< GL 0.3767377) TD TD)
 ((< ST 0.4587903) DB ((< ST 0.5382546) TD TD))) DB) DB)))
 ((< IN 0.8183197) DB DB)))
 ((< DP 0.0028336048) ((< GL 0.3227681) DB TD)
 ((< PR 0.8104538) TD DB))) ((< BM 0.32779098)
 ((< PR 0.3551054) ((< IN 0.965765) TD ((< DP 0.874076)

(((< AG 0.26613235) TD ((< ST 0.2661457) ((< PR 0.45051122) DB TD))
((< AG 0.3738824) TD TD))) ((< IN 0.8183197) DB DB)))
((< GL 0.115789056) TD DB)) DB))) DB))
((< AG 0.12635815) TD ((< DP 0.874076) ((< AG 0.26613235) TD
((< BM 0.32779098) ((< AG 0.10482597) TD ((< ST 0.15426922) DB TD))
DB)) TD)))

