PEMODELAN KOMPOSISI PAKAN KAMBING PERANAKAN ETAWA (PE) UNTUK OPTIMASI KANDUNGAN GIZI SUSU MENGGUNAKAN *ARTIFICIAL NEURAL NETWORK* (ANN) DAN *GENETIC ALGORITHM* (GA)

(Studi Kasus pada UPT Pembibitan Ternak dan Hijauan Makanan Ternak Singosari-Malang)

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh :
Latifah Hanum
NIM: 125150201111022

PROGRAM STUDI TEKNIK INFORMATIKA
JURUSAN TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2016

repository.ub.ac.id
UNIVERSITAS BRAWIJAYA

PENGESAHAN

PEMODELAN KOMPOSISI PAKAN KAMBING PERANAKAN ETAWA (PE) UNTUK OPTIMASI KANDUNGAN GIZI SUSU MENGGUNAKAN *ARTIFICIAL NEURAL NETWORK* (ANN) DAN *GENETIC ALGORITHM* (GA)

(Studi Kasus pada UPT Pembibitan Ternak dan Hijauan Makanan Ternak Singosari-Malang)

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh :
Latifah Hanum
NIM: 125150201111022

Skripsi ini telah diuji dan dinyatakan lulus pada
21 Juli 2016
Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I	Dosen Pembimbing II
Imam Cholissodin, S.Si, M.Kom	Marji, Drs., M.T
NIK. 201201 850719 1 001	NIP. 19670801 199203 1 001

Mengetahui
Ketua Jurusan Teknik Informatika

Tri Astoto 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 pustaka.

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, 21 Juli 2016

Latifah Hanum

NIM: 125150201111022

KATA PENGANTAR

Puji syukur kehadirat Allah SWT yang telah melimpahkan rahmat, taufik dan hidayah-Nya, sehingga laporan skripsi yang berjudul **"Pemodelan Komposisi Pakan Kambing Peranakan Etawa (PE) untuk Optimasi Kandungan Gizi Susu Menggunakan *Artificial Neural Network* (ANN) dan *Genetic Algorithm* (GA) (Studi Kasus pada UPT Pembibitan Ternak dan Hijauan Makanan Ternak Singosari-Malang)"** ini dapat terselesaikan dengan baik. Skripsi ini disusun untuk memenuhi syarat memperoleh gelar Sarjana Komputer pada Program Studi Informatika/ Ilmu Komputer, Fakultas Ilmu Komputer Universitas Brawijaya.

Penulis menyadari bahwa skripsi ini tidak akan berhasil tanpa bantuan dari beberapa pihak. Oleh karena itu, penulis ingin menyampaikan rasa hormat dan terima kasih kepada:

1. Bapak Imam Cholissodin, S.Si., M.Kom selaku dosen pembimbing skripsi 1 yang telah menyediakan waktu, tenaga dan pemikirannya untuk membimbing, berdiskusi dengan penulis dan membantu penyelesaian penelitian ini dengan baik.
2. Bapak Marji, Drs., M.T selaku dosen pembimbing skripsi 2 yang telah menyediakan waktu dan memberikan banyak masukan dan motivasi kepada penulis sehingga skripsi ini dapat terselesaikan dengan baik.
3. Bapak Rokim, S.Pt, Ibu drh. Jaya Wulandari dan Bapak Kayat selaku pakar di bidang pakan, kesehatan hewan dan hasil uji laboratorium kandungan gizi susu kambing PE di UPT Pembibitan Ternak dan Hijauan Makanan Ternak Singosari-Malang yang telah bersedia menjadi narasumber serta membimbing penulis dalam pengerjaan skripsi ini.
4. Bapak Heru Nurwarsito, Ir., M.Kom dan Bapak Putra Pandu Adikara, S.Kom., M.Kom selaku dosen penguji 1 dan 2 yang telah memberikan saran perbaikan skripsi ini.
5. Kedua orang tua, kakak, adik dan seluruh keluarga besar atas segala nasehat, motivasi, kasih sayang, perhatian, kesabarannya dalam membesarkan dan mendidik penulis serta yang selalu senantiasa mendukung dalam doa dan semangat demi terselesaikannya skripsi ini.
6. Seluruh teman-teman dan kerabat atas segala bentuk dukungan, baik moral atau materiil, serta ketersediaan waktu untuk berdiskusi dan membagi ilmunya sehingga sangat membantu penulis dalam setiap proses pengerjaan skripsi ini.
7. Saudari Canny Amerilyse Caesar, Yuni Widyaningtyas, Maulida Affiani Larasati, Kal Seliana Kuswantini, Kristi Widya, Maretta Asmarakandi, Yuanita Muharyani, Rizky Salsabella, Zahra Swastika Putri, Novinda Fiqih, Uswatun Hasanah, Sri Retno Mia, saudara Leo Justicia Silalahi dan Martua

Indah Christian Saragih Napitu atas segala bentuk bantuan, dukungan dan perhatian yang diberikan kepada penulis.

8. Seluruh civitas akademika Universitas Brawijaya, khusunya program studi Informatika/ Ilmu Komputer.
9. Pihak-pihak lain yang tidak dapat penulis sebut satu persatu, yang turut membantu penyelesaian skripsi baik secara langsung maupun tidak langsung.

Penulis menyadari bahwa dalam penyusunan skripsi ini masih banyak kekurangan. Oleh karena itu, penulis bersedia menerima kritik dan saran yang membangun demi memperbaiki diri. Harapan penulis semoga skripsi ini dapat memberikan manfaat pada semua pihak, terutama bagi penulis dan pengembang yang akan mengembangkan penelitian dari penulis. Akhir kata penulis berharap skripsi ini dapat membawa manfaat bagi semua pihak yang menggunakannya.

Malang, Juli 2016

Penulis

hanum_latifah@ymail.com

ABSTRAK

Susu merupakan salah satu sumber protein hewani yang mengandung semua zat yang mudah dicerna dan dimanfaatkan oleh tubuh. Ternak penghasil susu utama di Indonesia adalah sapi perah Peranakan *Friesian Holstein* (PFH) yang produksi susunya relatif rendah sehingga tidak dapat mencukupi kebutuhan protein hewani yang dibutuhkan oleh masyarakat. Permasalahan tersebut dapat diatasi dengan menjadikan kambing Peranakan Etawa (PE) sebagai ternak perah alternatif. Tingginya produksi susu dan kualitas kandungan gizi susu sangat dipengaruhi oleh faktor pakan yang berfungsi sebagai produksi dan pengatur proses-proses yang ada di dalam tubuh. Pihak pakan pada UPT Pembibitan Ternak dan Hijauan Makanan Ternak Singosari-Malang masih menghadapi permasalahan, yaitu tidak pernah memberikan komposisi pakan yang berbeda-beda sehingga susu kambing yang dihasilkan tidak pernah mencapai kandungan gizi susu yang optimal. Oleh karena itu, penulis menggunakan metode *Artificial Neural Network* (ANN) dan *Genetic Algorithm* (GA) untuk membuat pemodelan komposisi pakan kambing PE agar kandungan gizi susunya dapat optimal dengan hasil rata-rata kenaikan protein dan penurunan lemak untuk kualitas Premium dan Baik yang diperoleh oleh sistem sebesar 0.0852%, 2.3254%, 0.0292% dan 1.4372% dari bobot badan kambing sebesar 36 kg dengan menggunakan Parameter ANN dan GA terbaik yang dicapai oleh sistem berupa jumlah iterasi = 1000, jumlah *hidden layer* = 10, *learning rate* = 0.9, ukuran populasi = 100, jumlah generasi = 1000, *cr* = 0.1 dan *mr* = 0.9.

Kata kunci: Kambing Peranakan Etawa (PE), kandungan nutrisi pakan, kandungan gizi susu, *Artificial Neural Network* (ANN), *Genetic Algorithm* (GA)

ABSTRACT

Milk is one of the animal protein sources which it contains all of the substances easy to be digested and to be used for the body. The main dairy cattle in Indonesia is the Friesian Holstein (PFH) crossbreed dairy cow which has relatively low milk production so it can't to fulfill the animal protein needed by the society. This problem can be solved by making the Etawa crossbreed (PE) goat as an alternative dairy cattle. The high milk production and quality of milk nutrients contain are greatly influenced by the food factor which has the function as the production and the controller of the existing processes inside of the body. The food divisions of the UPT Cattle Breeding and the Cattle Food Greenery in Singosari-Malang still face the problem, that they never give a different feed composition so that the goat milk production never reached the optimum nutrition milk. Therefore, this research uses the method of ANN and GA to model the PE goat food composition in order to obtain the optimum nutrition milk with the result of an average increase of protein and decrease of fat to Premium and Good quality that obtained by the system is 0.0852%, 2.3254%, 0.0292% and 1.4372% from 36 kg goat weight with using the best parameter of ANN and GA that is iteration number = 1000, hidden layer amount = 10, learning rate = 0.9, population size = 100, generation amount = 1000, cr = 0.1 and mr = 0.9.

Keywords: *Etawa crossbreed (PE) goat, food nutrients contain, milk nutritions, Artificial Neural Network (ANN), Genetic Algorithm (GA)*

DAFTAR ISI

PENGESAHAN ii
PERNYATAAN ORISINALITAS iii
KATA PENGANTAR iv
ABSTRAK vi
ABSTRACT vii
DAFTAR ISI viii
DAFTAR TABEL xii
DAFTAR GAMBAR xv
DAFTAR LAMPIRAN xvii
BAB 1 PENDAHULUAN 1
1.1 Latar belakang 1
1.2 Rumusan masalah 3
1.3 Tujuan 3
1.4 Manfaat 3
1.5 Batasan masalah 4
1.6 Sistematika Penulisan 4
BAB 2 LANDASAN KEPUSTAKAAN 6
2.1 Kajian Pustaka 6
2.2 Jenis Kambing 11
2.3 Kambing Peranakan Etawa (PE) 11
2.4 Bahan Pakan Kambing 12
2.5 Kebutuhan Nutrisi Pakan Kambing Peranakan Etawa (PE) 13
2.6 Kandungan Gizi Susu Kambing Peranakan Etawa (PE) 15
2.7 *Artificial Neural Network* (ANN) 16
2.7.1 Konsep Dasar *Artificial Neural Network* (ANN) 16
2.7.2 Arsitektur *Artificial Neural Network* (ANN) 17
2.7.3 *Feedforward* 18
2.7.4 *Backpropagation* 19
2.7.5 Normalisasi Data 20
2.7.6 Denormalisasi 21

2.7.7 *Mean Square Error* (MSE) 21
2.8 *Genetic Algorithm* (GA) 22
2.8.1 Struktur *Genetic Algorithm* (GA) 22
2.8.2 Parameter *Genetic Algorithm* (GA) 22
2.8.3 Penerapan *Genetic Algorithm* (GA) 23
2.9 *Random Injection* 27
2.10 *Artificial Neural Network* (ANN) dan *Genetic Algorithm* (GA) 28
BAB 3 METODOLOGI PENELITIAN 30
3.1 Studi Literatur 31
3.2 Pengumpulan Data 31
3.3 Analisis Kebutuhan Sistem 32
3.3.1 Deskripsi Umum Sistem 33
3.3.2 Kebutuhan Perangkat 33
3.4 Perancangan Sistem 34
3.5 Implementasi Sistem 34
3.6 Pengujian dan Evaluasi Hasil 35
3.7 Pengambilan Kesimpulan dan Saran 35
BAB 4 PERANCANGAN 36
4.1 Alir Perancangan Sistem 36
4.1.1 Perhitungan Kandungan Gizi Susu Menggunakan ANN 37
4.1.2 Perhitungan Optimasi Komposisi Pakan Menggunakan GA 45
4.2 Formulasi Permasalahan 54
4.3 Perhitungan Manualisasi *Artificial Neural Network* (ANN) dan *Genetic Algorithm* (GA) 55
4.3.1 Perhitungan Manual *Artificial Neural Network* (ANN) 55
4.3.2 Perhitungan Manual *Genetic Algorithm* (GA) 78
4.4 Perancangan Antarmuka 84
4.4.1 Proses Pemilihan Jenis Pakan Hijauan 84
4.4.2 Proses Pelatihan ANN 85
4.4.3 Proses Pengujian ANN 86
4.4.4 Optimasi Metode GA 87
4.5 Perancangan Pengujian Algoritma 89

4.5.1 Pengujian Jumlah Iterasi 89
4.5.2 Pengujian Jumlah *Hidden Layer* 90
4.5.3 Pengujian Nilai *Learning Rate* 90
4.5.4 Pengujian Ukuran Populasi 91
4.5.5 Pengujian Jumlah Generasi 91
4.5.6 Pengujian Kombinasi *Crossover Rate* dan *Mutation Rate* 92
4.5.7 Pengujian Kombinasi Variasi *Output* yang Dioptimalkan 92
BAB 5 IMPLEMENTASI 93
5.1 Implementasi Algoritma *Artificial Neural Network* (ANN) 93
5.1.1 Proses Pelatihan *Artificial Neural Network* (ANN) 93
5.1.2 Proses Pengujian *Artificial Neural Network* (ANN) 100
5.2 Implementasi Algoritma *Genetic Algorithm* (GA) 102
5.2.1 Proses Inisialisasi Populasi Awal 102
5.2.2 Proses *Crossover* 105
5.2.3 Proses Mutasi 107
5.2.4 Proses Evaluasi dan Seleksi 108
5.2.5 Proses *Random Injection* 113
5.2.6 Proses Pemilihan Individu yang Lolos ke Generasi Selanjutnya 114
5.2.7 Proses Pemilihan Individu Terbaik 114
5.3 Implementasi Antarmuka Sistem 115
5.3.1 Antarmuka Hasil Pemilihan Jenis Pakan Hijauan 115
5.3.2 Antarmuka Hasil Pelatihan ANN 116
5.3.3 Antarmuka Hasil Pengujian ANN 117
5.3.4 Antarmuka Hasil Optimasi GA 118
BAB 6 PENGUJIAN DAN ANALISIS 119
6.1 Pengujian 119
6.2 Hasil Pengujian dan Analisis 119
6.2.1 Hasil Pengujian Terhadap Jumlah Iterasi 119
6.2.2 Hasil Pengujian Terhadap Jumlah *Hidden Layer* 121
6.2.3 Hasil Pengujian Terhadap Variasi Nilai *Learning Rate* 122
6.2.4 Hasil Pengujian Terhadap Ukuran Populasi 123
6.2.5 Hasil Pengujian Terhadap Jumlah Generasi 125

6.2.6 Hasil Pengujian Terhadap Kombinasi *Crossover Rate* dan *Mutation Rate* 126
6.2.7 Hasil Pengujian Terhadap Kombinasi Variasi *Output* yang Dioptimalkan 128
BAB 7 PENUTUP 130
7.1 Kesimpulan 130
7.2 Saran 131
DAFTAR PUSTAKA 132

DAFTAR TABEL

Tabel 2.1 Perbedaan Penelitian Sebelumnya dengan Skripsi Penulis 7
Tabel 2.2 Kandungan Nutrisi Bahan Pakan Kambing PE 14
Tabel 2.3 Perbandingan Kandungan Gizi Susu Kambing, Susu Sapi dan ASI 15
Tabel 2.4 Penggolongan Kualitas Susu Kambing Segar 16
Tabel 2.5 Populasi Inisialisasi 24
Tabel 2.6 Hasil *Crossover* dan Mutasi 25
Tabel 2.7 Individu Baru Hasil Seleksi *Elitism* 27
Tabel 3.1 Analisis Kebutuhan Data Penelitian 32
Tabel 4.1 Data Latih Sebelum Normalisasi 55
Tabel 4.2 Data Uji Sebelum Normalisasi 56
Tabel 4.3 Data Latih Sesudah Normalisasi 56
Tabel 4.4 Data Uji Sesudah Normalisasi 57
Tabel 4.5 Data ke-1 Perlakuan Premium Sesudah Normalisasi 57
Tabel 4.6 Bobot *Input Layer* ke *Hidden Layer* 57
Tabel 4.7 Jumlah Z_in_j Bobot Sinyal *Input* 58
Tabel 4.8 Hasil Perhitungan Sinyal *Input* dengan Fungsi Sigmoid biner 59
Tabel 4.9 Bobot *Hidden Layer* ke *Output Layer* 59
Tabel 4.10 Jumlah Y_in_k Bobot Sinyal *Hidden Layer* ke *Output Layer* 60
Tabel 4.11 Hasil Perhitungan Sinyal *Output* dengan Fungsi Sigmoid biner 61
Tabel 4.12 Target Data ke-1 Sesudah Normalisasi 61
Tabel 4.13 Hasil Perhitungan *Error Output* ke Target 61
Tabel 4.14 Hasil Perhitungan Perubahan Bobot *Output Layer* 62
Tabel 4.15 Hasil Menjumlahkan Delta Unit *Hidden* ke *Output* 63
Tabel 4.16 Hasil Menghitung Informasi *Error* 63
Tabel 4.17 Hasil Perubahan Bobot *Hidden Layer* 64
Tabel 4.18 Hasil Hitung *Error* pada Iterasi Pertama 64
Tabel 4.19 Bobot Baru pada Unit *Hidden* 65
Tabel 4.20 Bobot Baru pada Unit *Output* 65
Tabel 4.21 Bobot *Input Layer* ke *Hidden Layer* Iterasi Kedua 66
Tabel 4.22 Jumlah Z_in_j Bobot Sinyal *Input* Iterasi Kedua 67

Tabel 4.23 Hasil Perhitungan Sinyal *Input* pada Iterasi Kedua 67
Tabel 4.24 Bobot *Hidden Layer* ke *Output Layer* Iterasi Kedua 68
Tabel 4.25 Jumlah Y_in_k Bobot Sinyal *Hidden Layer* ke *Output Layer* Iterasi Kedua 68
Tabel 4.26 Hasil Perhitungan Sinyal *Output* pada Iterasi Kedua 69
Tabel 4.27 Hasil Perhitungan *Error Output* ke Target Iterasi Kedua 70
Tabel 4.28 Hasil Perhitungan Perubahan Bobot 70
Tabel 4.29 Hasil Menjumlahkan Delta Unit *Hidden* ke *Output* Iterasi Kedua 71
Tabel 4.30 Hasil Menghitung Informasi *Error* Iterasi Kedua 72
Tabel 4.31 Hasil Perubahan Bobot *Hidden Layer* Iterasi Kedua 72
Tabel 4.32 Hasil Hitung *Error* pada Iterasi Kedua 73
Tabel 4.33 Bobot Baru pada Unit *Hidden Layer* Iterasi Kedua 73
Tabel 4.34 Bobot Baru pada Unit *Output Layer* Iterasi Kedua 74
Tabel 4.35 Bobot *Input Layer* ke *Hidden Layer* Data Uji 74
Tabel 4.36 Jumlah Z_in_j Bobot Sinyal *Input* Data Uji 75
Tabel 4.37 Hasil Perhitungan Sinyal *Input* Data Uji 75
Tabel 4.38 Bobot dari *Hidden Layer* ke *Output Layer* Data Uji 76
Tabel 4.39 Jumlah Y_in_k Bobot Sinyal *Hidden Layer* ke *Output Layer* Data Uji 76
Tabel 4.40 Hasil Perhitungan Sinyal *Output* Data Uji 77
Tabel 4.41 Hasil Denormalisasi Data Uji 78
Tabel 4.42 Inisialisasi Populasi Awal 78
Tabel 4.43 Penentuan Nilai Alpha (α) 79
Tabel 4.44 Hasil Perhitungan Manual Proses *Crossover* 79
Tabel 4.45 Hasil Perhitungan Manual Proses Mutasi 80
Tabel 4.46 Hasil Perhitungan Nilai *Fitness* 82
Tabel 4.47 Hasil Proses Evaluasi dan Seleksi 82
Tabel 4.48 Rancangan Pengujian Jumlah Iterasi 89
Tabel 4.49 Rancangan Pengujian Jumlah *Hidden Layer* 90
Tabel 4.50 Rancangan Pengujian Nilai *Learning Rate* 90
Tabel 4.51 Rancangan Pengujian Ukuran Populasi 91
Tabel 4.52 Rancangan Pengujian Jumlah Generasi 91
Tabel 4.53 Rancangan Pengujian Kombinasi *cr* dan *mr* 92

Tabel 4.54 Rancangan Pengujian Kombinasi Variasi *Output* yang Dioptimalkan . 92
Tabel 4.55 Rancangan Pengujian Kenaikan Protein dan Penurunan Lemak 92
Tabel 6.1 Hasil Pengujian Terhadap Jumlah Iterasi 120
Tabel 6.2 Hasil Pengujian Terhadap Jumlah *Hidden Layer* 121
Tabel 6.3 Hasil Pengujian Terhadap Nilai *Learning Rate* 122
Tabel 6.4 Hasil Pengujian Terhadap Ukuran Populasi 124
Tabel 6.5 Hasil Pengujian Terhadap Jumlah Generasi 125
Tabel 6.6 Hasil Pengujian Terhadap Kombinasi *cr* dan *mr*..... 127
Tabel 6.7 Hasil Pengujian Terhadap Kombinasi Variasi *Output*..... 128
Tabel 6.8 Hasil Pengujian Terhadap Kenaikan Protein dan Penurunan Lemak .. 128

DAFTAR GAMBAR

Gambar 2.1 Kambing Peranakan Etawa (PE) 12
Gambar 2.2 Proses Komunikasi Antar *Neuron* 17
Gambar 2.3 *Single Layer Net* 17
Gambar 2.4 *Multi Layer Net* 18
Gambar 2.5 *Pseudocode Elitism Selection* 27
Gambar 3.1 Diagram Alir Metodologi Penelitian 30
Gambar 3.2 Diagram Blok Sistem 34
Gambar 4.1 Diagram Alir Perancangan Sistem 36
Gambar 4.2 Diagram Alir Proses ANN *Backpropagation* 37
Gambar 4.3 Diagram Alir Proses Normalisasi Data 38
Gambar 4.4 Diagram Alir Proses Pelatihan ANN *Backpropagation* 39
Gambar 4.5 Diagram Alir Proses *Feedforward* pada ANN 40
Gambar 4.6 Diagram Alir Proses *Backpropagation* pada ANN 42
Gambar 4.7 Diagram Alir Proses Pengujian ANN *Backpropagation* 43
Gambar 4.8 Diagram Alir Proses Denormalisasi 44
Gambar 4.9 Diagram Alir Perhitungan Optimasi Menggunakan GA 45
Gambar 4.10 Diagram Alir Inisialisasi Populasi Awal 46
Gambar 4.11 Diagram Alir Proses Reprduksi *Crossover* 47
Gambar 4.12 Diagram Alir Proses Reproduksi Mutasi 49
Gambar 4.13 Diagram Alir Perhitungan Nilai *Fitness* 51
Gambar 4.14 Diagram Alir Proses Evaluasi dan Seleksi 53
Gambar 4.15 Diagram Alir Proses *Random Injection* 54
Gambar 4.16 Rancangan Antarmuka Proses Pemilihan Jenis Pakan Hijauan 84
Gambar 4.17 Rancangan Antarmuka Proses Pelatihan ANN 86
Gambar 4.18 Rancangan Antarmuka Proses Pengujian ANN 87
Gambar 4.19 Rancangan Antarmuka Proses Optimasi GA 88
Gambar 5.1 *Source Code* Proses Normalisasi Data 94
Gambar 5.2 *Source Code* Proses *Feedforward* 97
Gambar 5.3 *Source Code* Proses *Backpropagation* 100
Gambar 5.4 *Source Code* Proses Pengujian ANN dan Denormalisasi Data 102

Gambar 5.5 *Source Code* Proses Inisialisasi Populasi Awal 104
Gambar 5.6 *Source Code* Proses *Crossover* 106
Gambar 5.7 *Source Code* Proses Mutasi 108
Gambar 5.8 *Source Code* Proses Evaluasi dan Seleksi 112
Gambar 5.9 *Source Code* Proses *Random Injection* 114
Gambar 5.10 *Source Code* Proses Individu yang Lolos ke Generasi Selanjutnya 114
Gambar 5.11 *Source Code* Proses Pemilihan Individu Terbaik 115
Gambar 5.12 Antarmuka Hasil Pemilihan Jenis Pakan Hijauan 116
Gambar 5.13 Antarmuka Hasil Pelatihan ANN 117
Gambar 5.14 Antarmuka Hasil Pengujian ANN 117
Gambar 5.15 Antarmuka Hasil Optimasi GA 118
Gambar 6.1 Grafik Hasil Pengujian Terhadap Jumlah Iterasi 120
Gambar 6.2 Grafik Hasil Pengujian Terhadap Jumlah *Hidden Layer* 122
Gambar 6.3 Grafik Hasil Pengujian Terhadap Nilai *Learning Rate* 123
Gambar 6.4 Grafik Hasil Pengujian Terhadap Ukuran Populasi 124
Gambar 6.5 Grafik Hasil Pengujian Terhadap Jumlah Generasi 126
Gambar 6.6 Grafik Hasil Pengujian Terhadap Kombinasi *cr* dan *mr* 127
Gambar 6.7 Grafik Hasil Pengujian Terhadap Kombinasi Variasi *Output* 129
Gambar 6.8 Grafik Hasil Pengujian Terhadap Kenaikan Protein dan Penurunan Lemak 129

DAFTAR LAMPIRAN

LAMPIRAN A SURAT PERMOHONAN DATA SKRIPSI 135
LAMPIRAN B HASIL WAWANCARA 136
LAMPIRAN C DATA UTAMA HASIL UJI LABORATORIUM 140
LAMPIRAN D BOBOT DAN PERUBAHAN BOBOT PADA MANUALISASI 146
LAMPIRAN E HASIL PENGUJIAN SKENARIO UNTUK KUALITAS SUSU PREMIUM 150
LAMPIRAN F HASIL PENGUJIAN SKENARIO UNTUK KUALITAS SUSU BAIK 154

BAB 1 PENDAHULUAN

1.1 Latar belakang

Susu adalah bahan makanan istimewa yang mengandung zat-zat makanan yang mudah dicerna dan dimanfaatkan oleh tubuh. Susu adalah jenis protein hewani berbentuk cairan berwarna putih yang didapatkan dari hewan ternak seperti sapi, kerbau atapun kambing dengan cara pemerahan (Hamidah, Sukada, & Swacita, 2012). Menurut Kepala Badan Kependudukan dan Keluarga Berencana Nasional (BKKBN) Surya Chandra Surapaty mengatakan bahwa laju pertumbuhan penduduk Indonesia saat ini semakin meningkat hingga mencapai 1,49% setiap tahunnya dari jumlah penduduk Indonesia pada tahun sebelumnya (Akuntono, 2015). Seiring dengan bertambahnya populasi penduduk dan pendapatan serta kesadaran masyarakat akan gizi mengakibatkan kebutuhan protein hewani berupa susu menjadi meningkat sehingga perlu dilakukan pemenuhan kebutuhan protein hewani dengan cara meningkatkan salah satu produksi ternak, yaitu susu (Nur, Setyowati, & Wahyuningsih, 2015).

Ternak penghasil susu utama di Indonesia adalah ras sapi perah Peranakan *Friesian Holstein* (PFH) yang produksi susunya relatif masih rendah, sehingga pemenuhan kebutuhan protein hewani yang dibutuhkan oleh masyarakat tidak dapat tercukupi dengan baik. Kambing perah merupakan salah satu ternak perah alternatif yang cocok untuk dikembangkan dan dapat diterima oleh masyarakat. Salah satu jenis kambing perah yang dapat dioptimalisasi pengembangannya adalah kambing Peranakan Etawa (PE) (Nur, Setyowati, & Wahyuningsih, 2015). Kambing PE adalah salah satu penyedia protein hewani asal ternak berupa daging atau susu (Fitriyanto, Astuti, & Utami, 2013). Kemampuan produksi susu kambing PE disebabkan adanya genotipe Etawah yang temasuk ke dalam tipe kambing perah (Budiarsana & Sutama, 2006). Susu kambing memiliki salah satu kelebihan, yaitu kandungan gizi susunya relatif lebih lengkap dan seimbang (Fitriyanto, Astuti, & Utami, 2013).

Jika dibandingkan dengan susu sapi, susu kambing lebih mempunyai karakteristik yang unik dan istimewa dibandingkan dengan susu yang dihasilkan oleh sapi. Perbedaan antara susu sapi dengan susu kambing adalah susu kambing tidak mengandung karoten sehingga warna susu yang dihasilkan oleh kambing lebih berwarna putih daripada susu sapi. Kadar protein susu kambing berkisar antara 3-5%, sedangkan kandungan lemak pada susu kambing berkisar antara 3-6% dan 99% terdiri dari gliserida dan steroid. Butiran lemak pada susu kambing sama seperti butiran lemak pada susu sapi, yaitu berdiameter 1-10 mikromikon. Akan tetapi, butiran lemak pada susu kambing lebih mudah dicerna oleh tubuh dikarenakan butiran lemak berdiameter kecil yang ada pada susu kambing memiliki jumlah yang cukup banyak (Sutama & Budiarsana, 1997). Faktor pakan merupakan salah satu faktor yang mempengaruhi tingginya produksi susu dan kualitas kandungan gizi susu (Prihatminingsih, Purnomoadi, & Harjanti, 2015). Fungsi pemberian pakan pada hewan ternak adalah sebagai sumber energi,

pembangunan dan pemeliharaan tubuh serta mengatur semua proses dalam tubuh (Subekti, 2009).

Bagian peternakan kambing Peranakan Etawa (PE) di UPT Pembibitan Ternak dan Hijauan Makanan Ternak Singosari-Malang masih menghadapi beberapa permasalahan, yaitu rendahnya kemampuan dalam memberikan *treatment* komposisi pakan terhadap kambing PE sehingga berpengaruh terhadap rendahnya nutrisi yang diterima kambing PE dan mengakibatkan kambing tidak dapat tumbuh dan berkembang dengan baik serta rendahnya produksi susu yang dihasilkan (Budiarsana & Sutama, 2001). Susu kambing berkualitas adalah susu yang memiliki kandungan gizi sesuai Standar Nasional Indonesia nomor 01-3141-1998 (Zuriati, Maheswari, & Susanty, 2011). Selain itu, pihak pakan UPT Pembibitan Ternak dan Hijauan Makanan Ternak Singosari-Malang tidak pernah memberikan komposisi pakan yang berbeda-beda sehingga susu kambing yang dihasilkan tidak pernah mencapai kandungan gizi yang optimal, meskipun selama ini susu kambing yang dihasilkan tidak kurang dari standar yang ada.

Permasalahan tersebut dapat diselesaikan dengan menerapkan metode *Artificial Neural Network* (ANN) dan *Genetic Algorithm* (GA) untuk dapat memodelkan komposisi pakan kambing PE sehingga kandungan gizi susu kambing yang dihasilkan dapat optimal. *Artificial Neural Network* (ANN) merupakan algoritma yang digunakan untuk menyelesaikan permasalahan pemodelan proses yang kompleks. Salah satu keunggulan yang dimiliki oleh ANN adalah ANN mampu mempelajari data-data historikalnya. Terdapat beberapa *layer* yang ada di dalam ANN, yaitu *input*, *hidden* dan *output layers* (Istadi & Rahmayanti, 2010).

Genetic Algorithm (GA) adalah salah satu jenis algoritma pencarian heuristik yang cara kerjanya berdasarkan mekanisme evolusi biologis. Algoritma ini didasari oleh konsep evolusi biologi, sehingga algoritma ini dapat memberikan solusi alternatif terhadap suatu masalah yang ingin diselesaikan. Salah satu keunggulan yang dimiliki oleh GA adalah GA mampu menyelesaikan permasalahan dengan memberikan solusi yang terbaik menggunakan metode seleksi, *crossover* dan mutasi tertentu (Aribowo, Lukas, & Gunawan, 2008). GA mampu menyelesaikan permasalahan yang kompleks dan sering digunakan serta dimanfaatkan di berbagai bidang seperti bidang fisika, biologi, ekonomi, sosiologi dan lain-lain (Mahmudy, 2013).

Pada penelitian sebelumnya telah membahas permasalahan mengenai pemodelan *time series* untuk meramalkan suatu nilai karakteristik tertentu pada periode mendatang menggunakan *Artificial Neural Network* (ANN) dan *Genetic Algorithm* (GA). Pada penelitian ini digunakan data sekunder yang berupa data harian dari nilai tukar mata uang Dolar Australia (AUD) terhadap Rupiah pada periode 3 Januari 2011 sampai dengan 29 Agustus 2012. Terdapat 348 data kurs jual yang tercatat selama periode tersebut. *Genetic Algorithm* (GA) digunakan sebagai metode pelatihan dalam *Feed Forward Neural Network* (FFNN). Penelitian tersebut berhasil menunjukkan bahwa penerapan model ANN dan GA

mampu memberikan hasil prediksi yang cukup akurat yang diperlihatkan dari kedekatan antara *target* dengan *output*-nya (Yuliandar, Warsito, & Yasin, 2012).

Berdasarkan latar belakang yang telah diuraikan di atas, penulis berinisiatif untuk merancang sistem berbasis komputer untuk memberikan solusi optimasi kandungan gizi susu kambing dalam penelitian tugas akhir yang berjudul "Pemodelan Komposisi Pakan Kambing Peranakan Etawa (PE) untuk Optimasi Kandungan Gizi Susu Menggunakan *Artificial Neural Network* (ANN) dan *Genetic Algorithm* (GA)" dengan harapan aplikasi ini dapat membantu peternak kambing PE dalam memodelkan komposisi pakan kambing PE untuk dapat mengoptimalkan kandungan gizi susu kambingnya.

1.2 Rumusan masalah

Berdasarkan latar belakang diatas, penulis merumuskan beberapa masalah sebagai berikut:

1. Bagaimana mengimplementasikan metode *Artificial Neural Network* (ANN) dan *Genetic Algorithm* (GA) guna menyelesaikan permasalahan pemodelan komposisi pakan kambing Peranakan Etawa (PE) untuk mengoptimalkan kandungan gizi susu kambing?
2. Berapa nilai parameter optimal dari metode *Artificial Neural Network* (ANN) dan *Genetic Algorithm* (GA) untuk optimasi dari kandungan gizi susu kambing Peranakan Etawa (PE)?
3. Berapa hasil kualitas solusi dari optimasi kandungan gizi susu kambing Peranakan Etawa (PE) dengan menggunakan metode *Artificial Neural Network* (ANN) dan *Genetic Algorithm* (GA)?

1.3 Tujuan

1. Mengimplementasikan metode *Artificial Neural Network* (ANN) dan *Genetic Algorithm* (GA) guna menyelesaikan permasalahan pemodelan komposisi pakan kambing Peranakan Etawa (PE) untuk mengoptimalkan kandungan gizi susu kambing.
2. Mengetahui nilai parameter optimal dari metode *Artificial Neural Network* (ANN) dan *Genetic Algorithm* (GA) untuk optimasi kandungan gizi susu kambing Peranakan Etawa (PE).
3. Mengetahui hasil kualitas solusi dari optimasi kandungan gizi susu kambing Peranakan Etawa (PE) dengan menggunakan metode *Artificial Neural Network* (ANN) dan *Genetic Algorithm* (GA).

1.4 Manfaat

Berdasarkan rumusan masalah yang telah dipaparkan pada sub bab sebelumnya, penelitian ini diharapkan dapat bermanfaat untuk berbagai pihak. Manfaat dari penelitian ini sebagai berikut:

1. Membantu para peternak kambing Peranakan Etawa (PE) dalam membuat kombinasi pakan kambing PE yang terbaik.
2. Membantu para peternak kambing Peranakan Etawa (PE) untuk meningkatkan atau meminimalkan kandungan gizi susu kambing dengan tetap memperhatikan nutrisi yang dibutuhkan oleh kambing PE.

1.5 Batasan masalah

Batasan masalah yang akan dijadikan sebagai pedoman dalam pelaksanaan penelitian ini sebagai berikut:

1. Jenis kambing yang diteliti adalah kambing perah, yaitu kambing Peranakan Etawa (PE) betina pada masa laktasi yang ada di kandang pemerahan.
2. Bobot badan kambing Peranakan Etawa (PE) yang diteliti adalah 32 kg, 34 kg, 36 kg, 38 kg dan 40 kg.
3. Jenis pakan yang digunakan ada 10 jenis bahan pakan hijauan, yaitu rumput taiwan, rumput odot, rumput setaria, rumput raja, rumput *brachiaria decumbens*, kaliandra, gamal, lamtoro, turi dan tarum.
4. Jenis pakan hijauan yang dapat dimodelkan hanya terdiri dari satu jenis pakan hijauan hingga dua jenis pakan hijauan.
5. Kebutuhan nutrisi pakan kambing Peranakan Etawa (PE) yang digunakan adalah abu, protein kasar (PK), lemak kasar (LK) dan serat kasar (SK).
6. Kandungan gizi susu kambing Peranakan Etawa (PE) yang digunakan adalah protein, lemak (*fat*), laktosa (*lactose*) dan berat jenis (*density*).
7. Kandungan gizi susu yang dapat dioptimalkan adalah protein, laktosa (*lactose*) dan berat jenis (*density*).
8. Kandungan gizi susu yang dapat diminimalkan adalah lemak.

1.6 Sistematika Penulisan

Sistematika penulisan penelitian diperlihatkan untuk memberikan gambaran dan uraian dari penyusunan tugas akhir secara garis besar yang meliputi beberapa bab sebagai berikut:

BAB 1 PENDAHULUAN

Bab ini menjelaskan tentang latar belakang, rumusan masalah, tujuan penelitian, manfaat penelitian, batasan masalah dan sistematika penulisan yang terkait dengan Pemodelan Komposisi Pakan Kambing Peranakan Etawa (PE) untuk Optimasi Kandungan Gizi Susu Menggunakan *Artificial Neural Network* (ANN) dan *Genetic Algorithm* (GA).

BAB 2 LANDASAN KEPUSTAKAAN

Bab ini menjelaskan tentang dasar teori dan referensi yang mendasari pembuatan sistem Pemodelan Komposisi Pakan Kambing Peranakan Etawa (PE)

untuk Optimasi Kandungan Gizi Susu Menggunakan *Artificial Neural Network* (ANN) dan *Genetic Algorithm* (GA).

BAB 3 METODOLOGI PENELITIAN

Bab ini menjelaskan tentang metode dan langkah kerja yang dilakukan dalam penulisan tugas akhir yang terdiri dari studi literatur, metode pengambilan data, analisis kebutuhan, perancangan sistem, implementasi, pengujian dan analisis serta pengambilan kesimpulan.

BAB 4 PERANCANGAN

Bab ini menjelaskan tentang perancangan sistem, formulasi permasalahan, perhitungan manualisasi, perancangan antarmuka dan perancangan skenario pengujian dalam pembuatan perangkat lunak sistem Pemodelan Komposisi Pakan Kambing Peranakan Etawa (PE) untuk Optimasi Kandungan Gizi Susu Menggunakan *Artificial Neural Network* (ANN) dan *Genetic Algorithm* (GA) yang akan dibuat.

BAB 5 IMPLEMENTASI

Bab ini menjelaskan tentang implementasi dari sistem Pemodelan Komposisi Pakan Kambing Peranakan Etawa (PE) untuk Optimasi Kandungan Gizi Susu Menggunakan *Artificial Neural Network* (ANN) dan *Genetic Algorithm* (GA) sesuai dengan perancangan sistem yang telah dibuat beserta implementasi antarmuka.

BAB 6 PENGUJIAN DAN ANALISIS

Bab ini menjelaskan proses dari hasil pengujian terhadap sistem Pemodelan Komposisi Pakan Kambing Peranakan Etawa (PE) untuk Optimasi Kandungan Gizi Susu Menggunakan *Artificial Neural Network* (ANN) dan *Genetic Algorithm* (GA) yang telah dibangun serta hasil analisis untuk memastikan bahwa program telah sesuai dengan perancangan.

BAB 7 PENUTUP

Bab ini terdiri dari kesimpulan yang diperoleh dari pembuatan dan pengujian perangkat lunak sistem Pemodelan Komposisi Pakan Kambing Peranakan Etawa (PE) untuk Optimasi Kandungan Gizi Susu Menggunakan *Artificial Neural Network* (ANN) dan *Genetic Algorithm* (GA) serta saran-saran yang diberikan oleh penulis untuk pengembangan lebih lanjut.

BAB 2 LANDASAN KEPUSTAKAAN

Pada bab ini akan dibahas mengenai landasan kepustakaan yang mendukung penelitian, diantaranya tentang kajian pustaka, jenis kambing, kambing Peranakan Etawa (PE), bahan pakan kambing, kebutuhan nutrisi pakan kambing PE, kandungan gizi susu kambing PE, *Artificial Neural Network* (ANN) yang meliputi konsep dasar ANN, arsitektur ANN, *feedforward*, *backpropagation*, normalisasi data, denormalisasi dan *mean square error* (MSE), *Genetic Algorithm* (GA) yang meliputi struktur GA, parameter GA dan penerapan GA, *random injection* serta *Artificial Neural Network* (ANN) dan *Genetic Algorithm* (GA).

2.1 Kajian Pustaka

Pada penelitian sebelumnya mengenai penggunaan metode *Artificial Neural Network* (ANN) dan algoritma genetika telah dilakukan dengan berbagai macam objek penelitian. Referensi yang digunakan pada skripsi ini diantaranya penelitian pertama adalah "Pelatihan *Feed Forward Neural Network* Menggunakan Algoritma Genetika dengan Metode Seleksi Turnamen untuk Data *Time Series*" oleh David Yuliandar, Budi Warsito dan Hasbi Yasin (2012). Objek yang digunakan dalam penelitian tersebut adalah data kurs uang. Penelitian ini menerapkan dua algoritma secara bersamaan, yaitu algoritma genetika dan *neural network* untuk peramalan *time series*. Tujuan dari penelitian ini adalah untuk memperoleh bobot-bobot yang optimal dan *error* yang minimal. Hasil pelatihan dan pengujian pada data kurs Dolar Australia terhadap Rupiah menunjukkan bahwa nilai RMSE yang diperoleh sebesar 117.3599 dan 82.4917. Penggunaan metode *Neural Network* dan algoritma genetika baik digunakan karena mampu memberikan hasil prediksi yang cukup akurat yang diperlihatkan oleh kedekatan *target* dengan *output* (Yuliandar, Warsito, & Yasin, 2012).

Penelitian kedua adalah "Hibridisasi *Artificial Neural Network* dan *Genetic Algorithm* dalam Penentuan Harga Produk pada Industri Mobil" oleh Muhammad Najib, Yudhi Purwananto dan Rully Soelaiman (2012). Penentuan harga produk pada industri mobil adalah suatu hal yang sulit baik dari sisi produsen maupun pengguna. Untuk membantu penentuan harga mobil, digunakan metode *Neural Network* yang dihibridisasi dengan *Genetic Algorithm* untuk evolusi *connection weight*. Metodologi penggabungan keduanya adalah dengan teknik *real-valued encoding* untuk merepresentasikan *connection weight* sebagai satu kromosom. Dari hasil pengujian didapatkan hasil bahwa penerapan hibridisasi ANN dan GA dapat membantu dalam penentuan harga mobil berdasarkan fitur-fitur dan mekanis mobil, penggunaan teknik *real-valued encoding* dapat merepresentasikan *connection weight* dan bias dengan baik serta nilai *fitness* berbanding lurus dengan nilai MSE hasil simulasi (Najib, Purwananto, & Soelaiman, 2012).

Penelitian ketiga adalah "Penentuan Arsitektur Jaringan Syaraf Tiruan *Backpropagation* (Bobot Awal dan Bias Awal) Menggunakan Algoritma Genetika" oleh Christian Dwi Suhendra dan Retantyo Wardoyo (2015). Kelemahan dari

jaringan syaraf tiruan *backpropagation* adalah sangat lama untuk konvergen dan permasalahan lokal minimum yang membuat jaringan syaraf tiruan (JST) sering terjebak pada lokal minimum. Kombinasi parameter arsitektur, bobot awal dan bias awal yang baik sangat menentukan kemampuan belajar dari JST untuk mengatasi kelemahan dari JST *backpropagation*. Penelitian ini akan mengembangkan sebuah metode untuk menentukan kombinasi parameter arsitektur, bobot awal dan bias awal yang akan digunakan sebagai parameter dalam perhitungan nilai *fitness*. Individu dengan SSE terkecil merupakan individu terbaik dan individu dengan kombinasi parameter arsitektur, bobot awal dan bias awal yang terbaik akan digunakan sebagai parameter dalam pelatihan JST *backpropagation* (Suhendra & Wardoyo, 2015).

Tabel 2.1 Perbedaan Penelitian Sebelumnya dengan Skripsi Penulis

No	Judul	Objek	Metode	Keluaran
		Masukan dan Parameter	**Proses**	**Hasil Penelitian**
1.	Pelatihan *Feed Forward Neural Network* Menggunakan Algoritma Genetika dengan Metode Seleksi Turnamen untuk Data *Time Series*	Data kurs uang periode tertentu	*Neural Network* dan Algoritma Genetika	Nilai bobot atau parameter yang optimum
		a. Jumlah bobot atau parameter adalah 25 unit b. Ukuran populasi c. Ukuran turnamen "k" d. Ukuran Pc e. Ukuran Pm	1. Inisialisasi populasi 2. Evaluasi setiap kromosom untuk memperoleh *fitness* 3. Seleksi kromosom dengan metode seleksi turnamen untuk memperoleh generasi baru 4. Algoritma genetika akan konvergen ke kromosom terbaik setelah generasi yang ditentukan telah dicapai oleh sistem	1. Algoritma genetika baik digunakan untuk melatih jaringan FFNN dengan *error* yang minimum, yaitu nilai RMSE sebesar 117.3599 dan 82.4917 2. Bobot atau parameter yang dihasilkan mencapai optimum global pada generasi ke-25000 dengan nilai *fitness* sebesar 0.060982
2.	Hibridisasi *Artificial Neural Network* dan *Genetic Algorithm* dalam Penentuan Harga Produk pada Industri Mobil	Penentuan harga mobil	*Artificial Neural Network* dan *Genetic Algorithm*	Penentuan harga mobil berdasarkan fitur teknis dan fisis mobilnya
		Terdapat 16 tipe entitas antara lain: a. *Symboling* b. *Normalized losses* c. *Wheel-base* d. *Length* e. *Width* f. *Height* g. *Curb-weight* h. *Engine-size*	1. Inisialisasi populasi, yaitu *connection weight* 2. Hitung total *connection weight* dan nilai *bias* 3. Kalkulasi nilai *fitness* dari *connection weight* 4. Hitung total *Mean Square Error*-nya untuk nilai *fitness* setiap	1. Aplikasi GAs dalam optimasi *connection weight* dan *bias* ANN dapat membantu dalam penentuan harga mobil 2. Penggunaan *real-valued encoding* dapat merepresentasikan *connection weight* dan *bias* dengan baik

No	Judul	Objek	Metode	Keluaran
		Masukan dan Parameter	**Proses**	**Hasil Penelitian**
		i. *Bore* j. *Stroke* k. *Compression-ratio* l. *Horsepower* m. *City mpg* n. *Highway mpg* o. *Peak-rpm* p. *Price*	kromosom 5. Jalankan proses evolusi dengan operator-operator genetika (*selection*, *crossover* dan mutasi) berdasarkan nilai *fitness* 6. Proses evolusi berhenti ketika kondisi terminasi tercapai	3. Penambahan jumlah populasi memberikan hasil yang lebih baik, tapi waktu komputasi yang lama 4. Peningkatan nilai *crossover fraction* memberikan efek terhadap nilai *fitness* 5. Nilai *fitness* berbanding lurus dengan nilai MSE hasil simulasi
3.	Penentuan Arsitektur Jaringan Syaraf Tiruan *Backpropagation* (Bobot Awal dan Bias Awal) Menggunakan Algoritma Genetika	Iris *flower*, pengguna *knowledge* dan *wine quality*	Algoritma genetika	Kombinasi parameter arsitektur, bobot awal dan bias awal yang terbaik
		a. *Learning* rate b. Arsitektur c. Bobot awal d. Bias awal	1. *Load* data latih tentang iris *flower*, pengguna *knowledge* dan *wine quality* 2. *Input* parameter *learning rate* 3. Inisialisasi arsitektur, bobot awal dan bias awal menggunakan algoritma genetika 4. Lakukan pelatihan menggunakan algoritma genetika 5. Simpan bobot dan bias terbaik 6. Lakukan *load* data uji 7. Lakukan pengujian dan dapatkan hasil pengujian	1. Algen berhasil menyelesaikan masalah penentuan parameter arsitektur JST 2. Individu terbaik untuk data iris *flower* adalah dengan nilai *fitness* 0.0460 untuk arsitektur 1 *hidden layer* dan jumlah *neuron* 21 dengan nilai SSE 21.758 3. Individu terbaik untuk data pengguna *knowledge* adalah dengan nilai *fitness* 0.038 untuk arsitektur 2 *hidden layer* dan jumlah *neuron* 26 dan 25 dengan nilai SSE 25.90 4. Individu terbaik *wine quality* adalah dengan nilai *fitness* 0.0443 untuk arsitektur 1 *hidden layer* dan jumlah *neuron* 40 dengan nilai SSE 22.573
4.	*Implementing Artificial Neural Network and Genetic Algorithms*	Produksi biogas	*Artificial Neural Network* (ANN) dan *Genetic Algorithm* (GA)	Pemodelan dan optimasi tentang produksi biogas

No	Judul	Objek	Metode	Keluaran
		Masukan dan Parameter	**Proses**	**Hasil Penelitian**
	to Solve Modeling and Optimisation of Biogas Production	Parameter yang telah ditentukan antara lain: a. Desain ANN diatur menjadi 3 *input nodes*, 1 *output nodes* dan beberapa parameter *hidden nodes*, yaitu 2, 3, 5 dan 10 b. Jumlah populasi yang digunakan adalah 100 c. Menghasilkan 2 anak d. Metode seleksi yang digunakan adalah *Roulette Wheel selection*	1. Data *normalisation* masuk ke dalam proses pemodelan ANN 2. Di dalam pemodelan ANN terdapat 2 proses, yaitu metode desain NN dan pelatihan NN 3. Dihasilkan model NN yang terlatih 4. Model NN yang terlatih diproses kembali untuk dilakukan optimasi menggunakan GAs 5. Sistem akan menampilkan hasil optimasinya dan data akan di *denormalisation*	1. ANN dan GA mampu menghasilkan hasil yang lebih baik dalam beberapa kasus tertentu 2. Pemodelan ANN dengan 2 *hidden neurons* gagal menghasilkan hasil yang lebih baik daripada RSM 3. Pemodelan ANN dengan 3 atau lebih *hidden neurons* diperkirakan lebih tinggi kecuali pemodelan dengan 5 *hidden neurons* dengan pelatihan PSO 4. *Output* model ANN memiliki akurasi tinggi dengan MSE ke *target output* yang kecil
5.	Pemodelan dan Optimasi Hidrolisa Pati Menjadi Glukosa dengan Metode *Artificial Neural Network-Genetic Algorithm*	Pati	*Artificial Neural Network* dan *Genetic Algorithm* (ANN-GA)	Proses hidrolasi pati menjadi glukosa yang paling optimum
		a. Mengatur indeks generasi, jumlah populasi dan jumlah variasi bebas b. Konsentrasi pati c. Konsentrasi enzim d. Suhu e. waktu	1. Pengembangan pemodelan ANN 2. Pelatihan pemodelan ANN 3. Inisialisasi populasi 4. Perhitungan *fitness* 5. Skala dari nilai *fitness* 6. Seleksi *parent* 7. Reproduksi *children* 8. Perpindahan tempat *children* pada populasi 9. *Increment* indeks generasi hingga didapatkan hasil yang konvergen 10. Mendapatkan individu dengan nilai *fitness* terendah sebagai solusi	1. Kondisi optimal yang dicapai oleh sistem berada pada konsentrasi pati 7.13% (w/v), konsentrasi enzim 1.47% (w/v), suhu 41^{0}C dan waktu hidrolisa 166 menit dengan *yield* glukosa yang dihasilkan sebesar 6.08 mg/ml 2. ANN dan GA mampu memodelkan dan mengoptimalkan secara baik dengan koefisien regresi (R^2) yang dihasilkan sebesar 0.9755

Sumber: (Yuliandar, Warsito, & Yasin, 2012), (Najib, Purwananto, & Soelaiman, 2012), (Suhendra & Wardoyo, 2015), (Fakharudin, Sulaiman, Salihon, & Zainol, 2013), (Istadi & Rahmayanti, 2010)

Penelitian keempat adalah "*Implementing Artificial Neural Network and Genetic Algorithms to Solve Modeling and Optimisation of Biogas Production*" oleh Abdul Sahli Fakharudin, Md Nasir Sulaiman, Jailani Salihon dan Norazwina Zainol (2013). Pemodelan dan optimasi banyak digunakan pada domain biologi dan kimia untuk memperbaiki dan meningkatkan efisiensi proses. *Artificial Neural Network* (ANN) dan *Genetic Algorithm* (GA) digunakan secara bersama-sama. Kemampuan utama ANN adalah sebagai sistem pengenalan pola yang digunakan untuk membangkitkan suatu model. Model hasil prediksi dari ANN akan dioptimalkan menggunakan algoritma genetika untuk menemukan hasil optimal berdasarkan pencarian heurisitik. ANN dan algoritma genetika tersebut diimplementasikan untuk memecahkan masalah pemodelan dan optimasi dari produksi biogas. Hasil penelitian menunjukkan bahwa akurasi pemodelan dengan *error* yang minimum tidak akan memberikan hasil yang lebih baik dan peningkatan maksimal dari produksi biogas sebesar 0.44% (Fakharudin, Sulaiman, Salihon, & Zainol, 2013).

Penelitian kelima adalah "Pemodelan dan Optimasi Hidrolisa Pati Menjadi Glukosa dengan Metode *Artificial Neural Network-Genetic Algorithm*" oleh Istadi dan Dian Rahmayanti (2010). Pada penelitian ini, ANN dan GA digunakan oleh peneliti untuk mengembangkan pemodelan dan optimasi dari proses hidrolisa pati menjadi glukosa. Parameter-parameter dalam proses hidrolisa akan dimodelkan menggunakan ANN. Kemudian, model ANN yang dihasilkan akan dioptimasi menggunakan GA hingga memperoleh kondisi optimum secara global. Hasil penelitian menunjukkan bahwa kondisi optimum yang dicapai oleh sistem meliputi konsentrasi pati 7.13% (w/v), konsentrasi enzim 1.47% (w/v), suhu 41^0C dan waktu hidrolisa 166 menit dengan *yield* glukosa yang dihasilkan sebesar 6.08 mg/ml. Penelitian ini menunjukkan bahwa metode ANN dan GA yang telah diterapkan pada proses hidrolisa pati menjadi glukosa mampu memodelkan dan mengoptimasi secara baik dengan koefisien regresi (R^2) yang dihasilkan sebesar 0.9755.

Pada penelitian ini membahas tentang Pemodelan Komposisi Pakan Kambing Peranakan Etawa (PE) untuk Optimasi Kandungan Gizi Susu Menggunakan *Artificial Neural Network* (ANN) dan *Genetic Algorithm* (GA). Berdasarkan beberapa penelitian yang telah dijelaskan pada Tabel 2.1, menunjukkan bahwa penerapan *Artificial Neural Network* (ANN) dan *Genetic Algorithm* (GA) dapat menghasilkan luaran yang lebih baik untuk digunakan dalam menyelesaikan beberapa kasus tertentu karena *Artificial Neural Network* (ANN) dan *Genetic Algorithm* (GA) mampu memberikan prediksi yang cukup akurat. Penerapan *Artificial Neural Network* (ANN) dan *Genetic Algorithm* (GA) pada permasalahan pemodelan komposisi pakan kambing PE diharapkan dapat membantu peternak kambing Peranakan Etawa (PE) dalam memodelkan komposisi pakan kambing Peranakan Etawa (PE) untuk dapat mengoptimalkan atau meminimalkan kandungan gizi susu kambingnya.

2.2 Jenis Kambing

Kambing adalah salah satu jenis hewan ternak ruminansia yang diternak guna diambil berbagai manfaatnya seperti daging dan susunya. Kambing terdiri dari 2 jenis, yaitu (Gunawan, 2013):

a. Kambing Potong

Kambing potong adalah kambing yang diternak dengan tujuan utama adalah produksi daging. Kambing potong adalah komoditas yang menjamur dalam negeri dalam waktu yang tidak sebentar. Ciri-ciri yang dimiliki kambing potong, yaitu lemak dan bulu pada kambing potong relatif sedikit serta badannya berukuran lebih besar.

b. Kambing Perah

Kambing perah adalah kambing yang diternak dengan tujuan utama adalah produksi susu. Produksi susu kambing dapat ditingkatkan jumlah produksinya dengan cara memilih beberapa kambing unggul yang dapat menghasilkan susu dalam jumlah produksi yang cukup banyak.

2.3 Kambing Peranakan Etawa (PE)

Kambing merupakan bagian penting dari sistem usahatani bagi sebagian petani di Indonesia yang tersebar luas masuk ke dalam berbagai kondisi agroekosistem mulai dari daerah dataran rendah sampai dataran tinggi (Sutama I. K., 2011). Klasifikasi ilmiah kambing sebagai berikut:

Kingdom	: Animalia
Filum	: Chordata
Kelas	: Mamalia
Ordo	: Artiodactyla
Famili	: Bovidae
Upafamili	: Caprinae
Genus	: Copra
Spesies	: C. aegagrus
Upaspesies	: ***C.a. hircus***

Kambing Peranakan Etawa (PE) adalah salah satu penyedia protein hewani asal ternak berupa daging dan susu. Kambing PE merupakan hasil persilangan antara kambing kacang dengan kambing etawah yang keberadaannya sudah adaptif dengan topografi di Indonesia. Kambing PE merupakan salah satu jenis kambing perah yang unggul dikarenakan kambing PE mempunyai kemampuan produksi susu sebanyak 1.5-3 liter per hari. Selain itu, kambing PE sangat adaptif terhadap topografi di Indonesia, sehingga lahan yang digunakan tidak terlalu luas dan pembudidayaannya yang relatif mudah (Fitriyanto, Astuti, & Utami, 2013).

Ciri-ciri kambing PE sebagai berikut (Fitriyanto, Astuti, & Utami, 2013):

1. Bentuk tubuh tinggi.
2. Bagian hidung keatas melengkung.
3. Telinga menggantung ke bawah dan terkulai sampai dengan 15-30 cm dan sedikit kaku.
4. Warna bulu bervariasi antara hitam dan coklat.
5. Bulu kambing PE jantan bagian atas leher, pundak lebih tebal dan agak panjang, sedangkan kambing PE betina hanya mempunyai bulu panjang yang terdapat di bawah ekor ke arah garis kaki.
6. Bobot badan hidup kambing PE jantan sekitar 40-45 kg dan kambing PE betina sekitar 35 kg.

Berikut adalah gambar dari hewan kambing PE yang ada di UPT Pembibitan Ternak dan Hijauan Makanan Ternak Singosari-Malang.

Gambar 2.1 Kambing Peranakan Etawa (PE)

Sumber: Dinas Peternakan Provinsi Jawa Timur

2.4 Bahan Pakan Kambing

Pakan adalah salah satu faktor terbesar guna menghasilkan produksi susu yang tinggi dan pertumbuhan kambing yang bagus (Hidayati, 2012). Pemberian pakan pada hewan ternak berfungsi sebagai pembangunan dan pemeliharaan tubuh, mengatur semua proses dalam tubuh dan sumber energi. Bahan pakan merupakan bahan-bahan yang diberikan pada hewan ternak untuk dimakan, dicerna sebagian atau seluruhnya, diabsorpsi dan yang memberikan manfaat bagi hewan ternak yang telah memakannya (Subekti, 2009). Kambing merupakan hewan ternak ruminansia dimana hijauan adalah pakan utama yang harus diberikan pada kambing. Semakin berkualitas pakan hijauan yang diberikan, mengakibatkan reproduksi hewan ternak ruminansia menjadi semakin baik. Pakan hijauan merupakan kombinasi antara rumput-rumputan dan kacang-kacangan (*legume*) yang diaduk dan dicampur menjadi satu guna melengkapi

unsur nutrisi yang dibutuhkan oleh hewan ternak (Koten, Wea, Soetrisno, Ngadiyono, & Soewignyo, 2014).

Selain pakan hijauan, ada konsentrat atau makanan penguat yang juga harus diberikan pada kambing. Konsentrat diberikan pada kambing agar kambing dapat memperoleh kandungan gizi yang tinggi dari konsentrat yang tidak ada pada pakan hijauan. Seekor kambing tidak bisa hanya diberikan makanan tunggal berupa konsentrat saja karena pemberian pakan konsentrat dalam satu hari penuh dapat menyebabkan gangguan pencernaan bahkan kematian pada kambing tersebut. Perpaduan antara pakan hijauan dan konsentrat dalam satu hari dijadikan sebagai panduan pemberian pakan kambing yang baik dan tidak beresiko tinggi (Susilawati & dkk, 2011). Terdapat 4 macam golongan bahan pakan yang diberikan pada kambing sebagai berikut (Cahyono, 2006):

1. Golongan rumput
   a. Rumput taiwan
   b. Rumput odot
   c. Rumput setaria
   d. Rumput raja
   e. Rumput *brachiaria decumbens*
   f. Rumput gajah
   g. Rumput benggala
   h. Rumput meksiko
2. Golongan kacang atau *leguminose*
   a. Kaliandra
   b. Gamal
   c. Lamtoro
   d. Turi
   e. Tarum
   f. Daun kacang tanah
   g. Daun kacang-kacangan
   h. Daun kacang
3. Hasil limbah pertanian
   a. Daun nangka
   b. Daun pisang
   c. Daun waru
   d. Daun ketela rambat
   e. Daun ketela pohon
   f. Daun jagung
4. Golongan makanan penguat (konsentrat)
   a. Bungkil kelapa sawit
   b. Pollar
   c. Empok jagung
   d. Bungkil kedelai
   e. Mineral
   f. Dedak
   g. Ampas kecap
   h. Ampas tahu

2.5 Kebutuhan Nutrisi Pakan Kambing Peranakan Etawa (PE)

Beberapa bahan makanan memiliki kandungan nutrisi yang beragam. Kandungan gizi susu kambing Peranakan Etawa (PE) seperti protein, lemak, laktosa dan mineral dipengaruhi oleh kandungan nutrisi yang terdapat pada

pakan yang diberikan pada kambing PE (Susilowati, Utami, & Suratim, 2013). Zat nutrien merupakan kumpulan zat-zat gizi berupa karbohidrat, protein, mineral, lemak dan vitamin yang terkandung di dalam bahan pakan dan digunakan oleh hewan ternak untuk kelangsungan hidup hewan ternak tersebut (Umiyasih & Anggraeny, 2007). Berikut ini macam-macam kandungan nutrisi bahan pakan yang dibutuhkan oleh kambing Peranakan Etawa (PE) untuk penelitian:

1. Abu

   Abu adalah kandungan nutrisi yang mengandung unsur-unsur berupa Mg, Fe, Ca, Na, K, CL dan P yang terkandung dalam zat pakan anorganik.

2. Protein Kasar (PK)

   Protein adalah zat pakan organik yang kompleks dengan berat molekul yang tinggi dan sering digunakan untuk mengefisiensikan penggunaan energi dan pertumbuhan otot pada hewan ternak. Kandungan nutrisi yang terdapat pada pakan hijauan yang dibutuhkan oleh kambing PE berupa protein kasar.

3. Lemak Kasar (LK)

   Lemak adalah kelompok dari golongan lipida yang terkandung pada semua bahan pangan dengan prosentase kandungan lemak yang berbeda-beda antar bahan pakan. Kandungan lemak yang ada di dalam bahan pakan sering disebut sebagai lemak kasar, yaitu suatu kandungan yang berupa total lipida dalam jumlah yang sebenarnya.

4. Serat Kasar (SK)

   Serat kasar adalah kumpulan sisa-sisa sel bahan pakan yang tahan terhadap reaksi hidrolisis enzim-enzim pada saluran pencernaan hewan ternak yang tersusun dari karbohidrat. Jumlah pakan yang dikonsumsi oleh hewan ternak dapat menentukan jumlah serat kasar yang ada dalam pakan.

Berikut adalah tabel kandungan nutrisi dari 10 jenis bahan pakan Kambing PE berupa golongan rumput dan *leguminose* yang diperoleh dari UPT Pembibitan Ternak dan Hijauan Makanan Ternak Singosari-Malang setiap 1 kilogram.

Tabel 2.2 Kandungan Nutrisi Bahan Pakan Kambing PE

No	Bahan Pakan	Abu (%)	PK (%)	LK (%)	SK (%)
1	Rumput Taiwan	18.78	16.90	1.20	26.45
2	Rumput odot	16.97	17.03	1.76	24.84
3	Rumput setaria	11.5	8.5	1.76	32.5
4	Rumput raja	18.6	13.5	3.5	24.84
5	Rumput *brachiaria decumbens*	10.6	8.3	1.2	38.3
6	Kaliandra	7.27	21.55	2.99	13.96
7	Gamal	9.60	19.96	3.85	17.59
8	Lamtoro	0	24.2	3.7	21.5
9	Turi	0	29.2	3.4	17.2
10	Tarum	6.42	24.17	0	0

Sumber: Dinas Peternakan Provinsi Jawa Timur

2.6 Kandungan Gizi Susu Kambing Peranakan Etawa (PE)

Menurut Dewan Standarisasi Nasional (1998) dalam Standar Nasional Indonesia nomor 01-3141-1998 menjelaskan bahwa susu segar merupakan cairan yang diperoleh dari proses pemerahan terhadap ambing ternak sehat dengan prosedur pemerahan yang benar tanpa mengalami penambahan ataupun pengurangan komponen tertentu yang kemurnian susunya dijaga dengan baik. Penggolongan kualitas susu segar dapat dianalisis dengan cara melihat kadar protein, kadar lemak, berat jenis dan kadar laktosa yang ada di dalam susu segar tersebut.

Susu kambing dikonsumsi untuk dijadikan sebagai sumber gizi yang kualitas susunya tidak jauh berbeda dengan kualitas yang dimiliki oleh susu sapi dan Air Susu Ibu (ASI). Kelebihan susu kambing adalah susu kambing memiliki kandungan gizi yang relatif lebih lengkap dan seimbang jika dibandingkan dengan susu sapi (Fitriyanto, Astuti, & Utami, 2013). Selain itu, susu kambing warnanya lebih putih dibandingkan dengan warna dari susu sapi dikarenakan susu kambing tidak mengandung karoten. Kadar protein susu kambing berkisar 3-5%, kandungan lemak pada susu kambing berkisar 3-6% dan 99% terdiri dari gliserida dan steroid. Butiran lemak pada susu kambing berdiameter 1-10 mikromikron lebih banyak dibandingkan pada susu sapi sehingga mudah dicerna (Sutama & Budiarsana, 1997).

Susu kambing memiliki kandungan protein yang didominan oleh *Beta-casein* sama seperti jenis protein yang ada di dalam ASI. Susu kambing aman dikonsumsi oleh penderita penyakit maag (Gastritis) dikarenakan jenis protein dalam susu kambing yang bercampur dengan asam lambung dalam sistem pencernaan mampu menghasilkan senyawa-senyawa yang lebih mudah untuk dicerna dibandingkan dengan senyawa-senyawa dari protein susu sapi (Prawira, Djaja, & Imelda, 2007).

Berikut adalah tabel perbandingan kandungan gizi susu kambing, susu sapi dan Air Susu Ibu (ASI) per 100 gram (Sutama & Budiarsana, 1997) dan tabel penggolongan kualitas susu kambing segar berdasarkan karakteristiknya (Zuriati, Maheswari, & Susanty, 2011) yang diperlihatkan pada Tabel 2.3 dan Tabel 2.4.

Tabel 2.3 Perbandingan Kandungan Gizi Susu Kambing, Susu Sapi dan ASI

No	Nutrisi	Susu Kambing	Susu Sapi	Air Susu Ibu (ASI)
1	Protein	3.6	3.3	1.0
2	Lemak	4.2	3.3	4.4
3	Karbohidrat	4.5	4.7	6.9
4	Kalori	69	61	70
5	Posfor (P)	111	93	14
6	Kalsium (Ca)	134	19	32
7	Magnesium (Mg)	14	13	3
8	Besi (Fe)	0.05	0.05	0.03
9	Natrium (Na)	50	49	17
10	Kalium (K)	204	152	51

11	Vitamin A	185	126	241
12	Thiamin	0.05	0.04	0.014
13	Riboflavin	0.14	0.16	0.04
14	Niacin	0.28	0.08	0.18
15	Vitamin B-6	0.05	0.04	0.01

Sumber: Diadaptasi dari US Department of Agriculture (1976)

Tabel 2.4 Penggolongan Kualitas Susu Kambing Segar

Karakteristik	Kriteria Kualitas		
	Premium	Baik	Standar
Protein (%)	> 3.70	> 3.40 – 3.70	3.10 – 3.40
Lemak (%)	> 4.00	> 3.50 – 4.00	3.25 – 3.50
Bahan kering (%)	> 13.00	> 12.00 – 13.00	11.70 – 12.00

Sumber: Diadaptasi dari Thai Agricultural Standard (2008)

2.7 *Artificial Neural Network* (ANN)

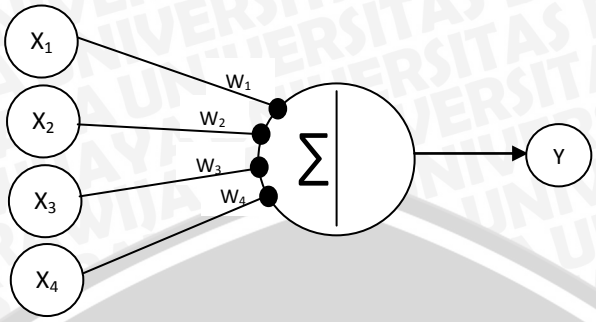
Artificial Neural Network (ANN) atau yang dikenal dengan istilah *Neural Network* (NN) adalah suatu sistem yang bekerja untuk memproses informasi yang masuk ke dalam sistem dimana cara kerja dari ANN memiliki kemiripan dengan cara kerja pada jaringan syaraf biologi manusia, yaitu bersifat adaptif, bekerja secara dinamis dan mampu mempelajari informasi-informasi yang belum diketahui sebelumnya. Terdapat dua tahapan pemrosesan informasi, yaitu tahapan pelatihan dan pengujian ANN (Yuliandar, Warsito, & Yasin, 2012).

Tahapan pelatihan ANN adalah tahap pemrosesan informasi dengan cara melakukan pelatihan, yaitu memasukkan data latih yang tersedia ke dalam jaringan. Data latih tersebut akan diproses oleh jaringan dengan cara mengubah-ubah bobot serta adanya evaluasi terhadap *output* jaringan setiap satu kali iterasi. Tahapan ini berlangsung selama beberapa iterasi dan sistem akan berhenti melakukan tahapan pelatihan ANN setelah jaringan telah menemukan bobot dan nilai *error* yang diinginkan atau jumlah iterasi telah mencapai nilai maksimal yang ditetapkan. Bobot terakhir yang diperoleh dari tahapan pelatihan ANN akan dijadikan sebagai *knowledge base* (dasar pengetahuan) yang digunakan pada tahapan pengujian ANN. Tahapan pengujian ANN adalah tahap pemrosesan informasi dengan cara memasukkan data uji dengan menggunakan bobot terakhir dari hasil tahapan pelatihan ANN. Bobot-bobot hasil tahapan pelatihan ANN diharapkan mampu menghasilkan nilai *error* yang minimal pada tahapan pengujian ANN (Yuliandar, Warsito, & Yasin, 2012).

2.7.1 Konsep Dasar *Artificial Neural Network* (ANN)

Arsitektur jaringan dalam ANN tersusun atas *neuron-neuron* yang tergabung dari seluruh lapisan yang memiliki keterhubungan antara lapisan yang satu dengan yang lain dimana *neuron-neuron* yang berada dalam satu lapisan tersebut mempunyai pola keterhubungan yang sama (Yuliandar, Warsito, & Yasin, 2012). Setiap *neuron* terhubung dan mengalirkan sinyal, sehingga sel-sel syaraf akan menggunakan fungsi aktivasi untuk menentukan sinyal keluaran yang

seharusnya (Nurmila, Sugiharto, & Sarwoko, 2012). Gambaran proses komunikasi antar *neuron* diperlihatkan pada Gambar 2.2 Proses Komunikasi Antar *Neuron*.



Gambar 2.2 Proses Komunikasi Antar *Neuron*

Sumber: (Nurmila, Sugiharto, & Sarwoko, 2012)

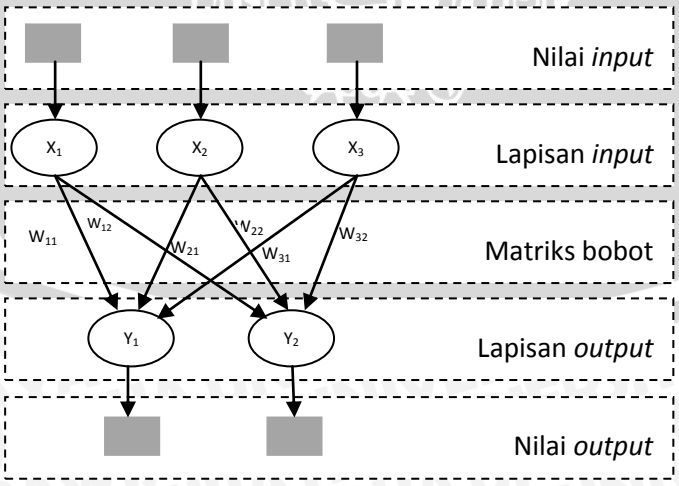
ANN terdiri atas beberapa lapisan-lapisan yang menyusunnya seperti *input layer*, *hidden layer* dan *output layer*. *Input layer* adalah salah satu lapisan dari ANN yang bertugas untuk menerima masukan atau *input* dari jaringan luar. *Hidden layer* adalah salah satu lapisan dari ANN yang tidak berhubungan langsung dengan kondisi yang ada di luar jaringan dan terletak dalam satu atau beberapa lapisan. *Output layer* adalah salah satu lapisan dari ANN yang bertugas untuk menghasilkan *output* dari jaringan (Yuliandar, Warsito, & Yasin, 2012).

2.7.2 Arsitektur *Artificial Neural Network* (ANN)

Terdapat 3 macam arsitektur ANN, yaitu (Wuryandari & Afrianto, 2012):

1. *Single Layer Net*

Single layer net adalah salah satu arsitektur ANN yang hanya memiliki satu lapisan saja yang digunakan untuk melakukan beberapa tugas secara bersamaan seperti menerima masukan dan mengolahnya menjadi keluaran tanpa harus melalui *hidden layer*. Berikut ini merupakan penggambaran *single layer net* yang diperlihatkan pada Gambar 2.3.

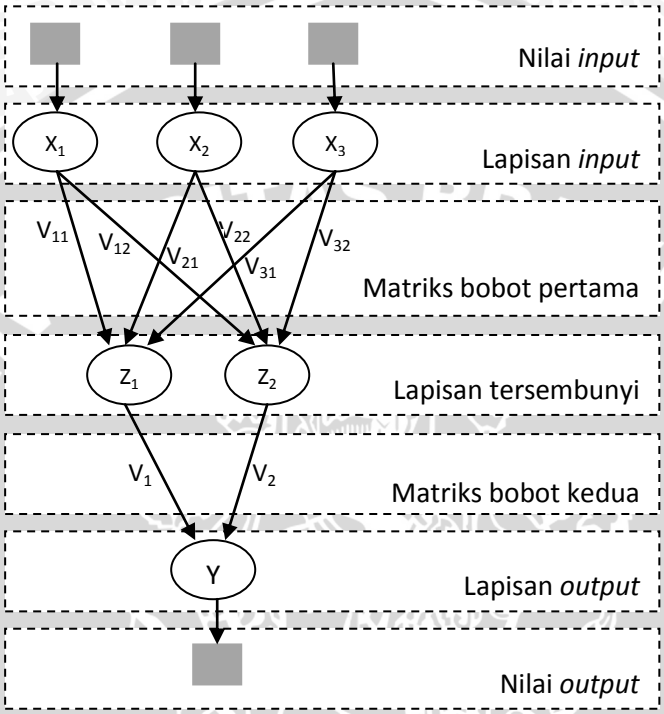


Gambar 2.3 *Single Layer Net*

Sumber: (Wuryandari & Afrianto, 2012)

2. *Multi Layer Net*

Multi layer net adalah salah satu arsitektur ANN yang memiliki satu lapisan atau lebih. Lapisan tersebut diletakkan antara *input layer* dan *output layer*. Arsitektur ANN jenis ini digunakan untuk menyelesaikan permasalahan yang lebih kompleks dengan proses pembelajaran yang lebih susah jika dibandingkan dengan *single layer net*. Berikut ini merupakan penggambaran *multi layer net* yang diperlihatkan pada Gambar 2.4 *Multi Layer Net*.



Gambar 2.4 *Multi Layer Net*

Sumber: (Wuryandari & Afrianto, 2012)

3. *Competitive Layer Net*

Competitive layer net adalah salah satu arsitektur ANN yang terdiri dari beberapa *neuron-neuron* yang dikumpulkan menjadi satu dan saling bersaing antara *neuron* yang satu dengan yang lain guna mendapatkan hak menjadi *neuron* yang aktif. Hubungan antar *neuron* yang ada pada lapisan kompetitif tidak diperlihatkan pada diagram arsitektur.

2.7.3 *Feedforward*

Feedforward atau *Multi Layer Perceptron* merupakan *neural network* yang tersusun dari beberapa *layer*. FFNN memiliki struktur yang tesusun atas satu *input layer*, satu atau lebih *hidden layer* dan satu *output layer*. Terdapat proses perhitungan fungsi aktivasi, yaitu fungsi matematis yang digunakan untuk membatasi dan menentukan jangkauan *output* suatu *neuron* hingga didapatkan nilai asimtot maksimal dan minimal. Fungsi aktivasi yang digunakan pada jaringan *backpropagation* terdiri dari 2 jenis fungsi aktivasi, yaitu fungsi sigmoid biner yang nilainya berkisar antara 0 dan 1 dan fungsi sigmoid bipolar yang

nilainya berkisar antara -1 dan 1 (Fernanda & Otok, 2012). Rumus untuk fungsi sigmoid biner diperlihatkan pada Persamaan 2-1, sedangkan rumus untuk fungsi sigmoid bipolar diperlihatkan pada Persamaan 2-2 berikut ini.

$$y = f(x) = \frac{1}{1+e^{-\alpha x}} \quad (2\text{-}1)$$

$$y = f(x) = \frac{1-e^{-x}}{1+e^{-x}} \quad (2\text{-}2)$$

Langkah-langkah yang dijalankan pada proses *feedforward* sebagai berikut (Fernanda & Otok, 2012):

1. Setiap unit *input* $(X_i, i = 1,\ldots,n)$ bertugas untuk menerima sinyal *input* X_i dan menyebarkan sinyal tersebut ke semua unit pada *hidden layer*.
2. Setiap unit *hidden* $(Z_j, j = 1,\ldots,p)$ bertugas untuk menjumlahkan bobot yang diperoleh dari sinyal *input* dengan menggunakan Persamaan 2-3.

   $$Z_{inj} = \sum_{i}^{n} x_i v_{ij} \quad (2\text{-}3)$$

3. Tahap selanjutnya adalah menerapkan fungsi aktivasi untuk menghitung sinyal *input*, kemudian mengirimkan sinyal tersebut ke semua unit pada *output layer* dengan menggunakan Persamaan 2-4.

   $$z_j = f(z_{in_j}) \quad (2\text{-}4)$$

4. Setiap unit *output* $(Y_k, k = 1,\ldots,m)$ bertugas untuk menjumlahkan bobot dari sinyal *output* dengan menggunakan Persamaan 2-5.

   $$y_{ink} = \sum_{j}^{p} z_j w_{jk} \quad (2\text{-}5)$$

5. Tahap selanjutnya adalah menerapkan fungsi aktivasi untuk menghitung sinyal *output* dengan menggunakan Persamaan 2-6.

   $$y_k = f(y_{in_k}) \quad (2\text{-}6)$$

2.7.4 *Backpropagation*

Algoritma *backpropagation* merupakan algoritma yang umumnya digunakan pada model *Artificial Neural Network* (ANN) dengan tujuan untuk melakukan proses pelatihan jaringan dengan menggunakan beberapa *layer* (Raharjo, 2013). Langkah-langkah yang dijalankan pada proses *backpropagation* sebagai berikut (Fernanda & Otok, 2012):

1. Setiap unit *output* $(Y_k, k = 1,\ldots,m)$ bertugas untuk menerima pola target berdasarkan pola yang ada pada pelatihan, kemudian melakukan proses perhitungan *error output* menggunakan Persamaan 2-7.

$$\delta_k = (t_k - y_k) f'(y_{in_k}) \quad (2\text{-}7)$$

2. Menghitung bobot koreksi untuk meng-*update* w_{jk} yang sebelumnya menggunakan Persamaan 2-8.

$$\Delta w_{jk} = \alpha \delta_k z_j \quad (2\text{-}8)$$

3. Setiap unit *hidden* $(Z_j, j = 1, \ldots, p)$ bertugas untuk menjumlahkan *input* delta yang berasal dari *layer* diatasnya menggunakan Persamaan 2-9.

$$\delta_{in\,j} = \sum_{k=1}^{m} \delta_k w_{jk} \quad (2\text{-}9)$$

4. Mengalikan hasil yang diperoleh dari Persamaan 2-9 dengan fungsi aktivasi untuk menghitung informasi *error* menggunakan Persamaan 2-10.

$$\delta_j = \delta_{in\,j} f'(z_{in\,j}) \quad (2\text{-}10)$$

5. Menghitung koreksi bobot untuk meng-*update* v_{ij} yang sebelumnya menggunakan Persamaan 2-11.

$$\Delta v_{ij} = \alpha \delta_j x_i \quad (2\text{-}11)$$

6. Tahap perubahan bobot yang dilakukan dengan ketentuan sebagai berikut:
   a. Setiap unit *output* $(Y_k, k = 1, \ldots, m)$ bertugas untuk meng-*update* bobotnya menggunakan Persamaan 2-12.

$$w_{jk}(baru) = w_{jk}(lama) + \Delta w_{jk} \quad (2\text{-}12)$$

   b. Setiap unit *hidden* $(Z_j, j = 1, \ldots, p)$ bertugas untuk meng-*update* bobotnya menggunakan Persamaan 2-13.

$$v_{ij}(baru) = v_{ij}(lama) + \Delta v_{ij} \quad (2\text{-}13)$$

2.7.5 Normalisasi Data

Normalisasi data merupakan suatu proses untuk melakukan skala data sehingga suatu data berada dalam suatu rentang nilai tertentu. Normalisasi data dapat digunakan untuk meningkatkan keakurasian dari hasil *output* serta memfasilitasi proses *learning* dari jaringan syaraf. Normalisasi data dilakukan untuk masing-masing data latih dan data uji yang akan digunakan (Sabati, Dania, & Putri, 2014).

Jenis metode normalisasi data yang digunakan dalam penelitian ini adalah metode normalisasi data *Min-Max*, yaitu sebuah metode yang digunakan untuk dapat me-*rescale* suatu data dari *range* yang lama ke *range* yang baru terhadap rentang nilai yang telah ditetapkan sebelumnya. Persamaan matematika dari metode normalisasi data *Min-Max* diperlihatkan pada Persamaan 2-14 sebagai berikut (Chamidah, Wiharto, & Salamah, 2012):

$$x' = \left(0.8 * \frac{x - \min_value}{\max_value - \min_value} \right) + 0.1 \quad (2\text{-}14)$$

Keterangan:

x : data

x' : hasil normalisasi data

min_value : nilai minimum data

max_value : nilai maksimum data

2.7.6 Denormalisasi

Denormalisasi merupakan tahapan dimana *output* dikembalikan ke kondisi aslinya setelah *input* mengalami normalisasi pada saat proses *preprocessing*. Tujuan denormalisasi adalah untuk mengkonversikan kembali hasil keluaran menjadi data *real* yang telah diprediksi. Persamaan matematika dari denormalisasi diperlihatkan pada Persamaan 2-15 sebagai berikut (Gema, 2014):

$$y'' = \frac{(\max_value - \min_value) * (y - 0.1)}{0.8} + \min_value \quad (2\text{-}15)$$

Keterangan:

y" : hasil denormalisasi

y : hasil perhitungan sinyal *output*

min_value : nilai minimum data

max_value : nilai maksimum data

2.7.7 *Mean Square Error* (MSE)

Mean Square Error (MSE) pada *Artificial Neural Network* (ANN) adalah metode yang biasanya digunakan untuk melakukan proses evaluasi kesalahan dan meningkatkan optimasi dari proses yang telah dilakukan sebelumnya. Persamaan matematika dari MSE diperlihatkan pada Persamaan 2-16 sebagai berikut (Edo, 2014):

$$MSE = \frac{1}{k} \sum_{k=1}^{n} (t_k - y_k)^2 \quad (2\text{-}16)$$

Keterangan:

n : jumlah data

y_k : nilai *output* data

t_k : nilai target data

2.8 *Genetic Algorithm* (GA)

Genetic Algorithm (GA) atau algoritma genetika adalah salah satu algoritma evolusi yang popular dalam hal menyelesaikan permasalahan optimasi yang kompleks. Saat ini, *Genetic Algorithm* semakin banyak digunakan dalam berbagai bidang seperti biologi, sosiologi, fisika, ekonomi dan bidang-bidang lainnya dikarenakan GA memiliki kemampuan dalam menyelesaikan masalah sederhana sampai kompleks (Mahmudy, 2013). *Genetic Algorithm* adalah algoritma pencarian heuristik yang didasari oleh konsep evolusi biologi dan dapat memberikan solusi alternatif atas suatu masalah yang hendak diselesaikan dengan memanfaatkan metode *crossover*, mutasi dan seleksi (Aribowo, Lukas, & Gunawan, 2008).

Proses pencarian solusi pada *Genetic Algorithm* dilakukan melalui beberapa tahapan. Tahapan pertama adalah inisialisasi populasi awal yang terdiri dari kromosom-kromosom yang menggambarkan solusi atas permasalahan yang ingin dipecahkan secara acak. Tahapan kedua adalah memilih satu atau beberapa populasi untuk dilakukan proses reproduksi berupa *crossover* dan mutasi guna menghasilkan keturunan baru. Proses *crossover* adalah proses persilangan antar kromosom yang berada dalam satu generasi. Proses mutasi adalah proses perubahan satu nilai gen kromosom dalam satu populasi. Tahapan ketiga adalah proses evaluasi dengan melakukan perhitungan *fitness function*. Tahapan keempat adalah proses seleksi berdasarkan hasil perhitungan *fitness function* hingga ditentukan populasi yang akan dipertahankan untuk generasi selanjutnya (Aribowo, Lukas, & Gunawan, 2008).

2.8.1 Struktur *Genetic Algorithm* (GA)

Proses algoritma genetika berawal dari proses inisialisasi. Proses inisialisasi merupakan proses untuk membuat individu baru secara acak yang mempunyai kromosom tertentu dimana kromosom tersebut menjadi salah satu solusi untuk masalah yang akan diselesaikan. Setelah melakukan proses inisialisasi, selanjutnya melakukan proses reproduksi yang berfungsi untuk menghasilkan keturunan (*offspring*). Proses reproduksi terdiri dari 2 operator genetika, yaitu tukar silang (*crossover*) dan mutasi (*mutation*). Setelah melakukan proses reproduksi dilanjutkan dengan proses evaluasi, dimana proses evaluasi ini berfungsi untuk menghitung nilai *fitness*. Langkah selanjutnya adalah proses seleksi, yaitu memilih individu yang akan dipertahankan hidup pada generasi selanjutnya yang diambil dari himpunan populasi dan *offspring* (Mahmudy, 2013).

2.8.2 Parameter *Genetic Algorithm* (GA)

Proses *Genetic Algorithm* (GA) dilakukan berdasarkan parameter-parameter tertentu yang memberikan pengaruh terhadap kinerja dan perilaku dari algoritma ini. Parameter-parameter yang terdapat pada GA meliput ukuran populasi (*popsize*), jumlah generasi, probabilitas *crossover* (Pc) dan probabilitas

mutasi (Pm) (Bramantyo, 2006). Parameter penting yang mempengaruhi performa GA yaitu:

1. Fungsi *fitness* yang dimiliki oleh masing-masing individu.
2. Jumlah populasi (*popsize*) yang digunakan pada setiap generasi.
3. *Crossover rate* yang terjadi pada proses *crossover* pada setiap generasi.
4. *Mutation rate* yang terjadi pada proses mutasi pada setiap generasi.
5. Jumlah generasi yang digunakan untuk menentukan lama penerapan algoritma genetika.

Rekomendasi untuk menentukan nilai parameter sebagai berikut:

1. Jika permasalahan yang sedang diselesaikan mempunyai kawasan yang cukup besar, maka De Jong mempunyai rekomendasi untuk nilai parameter yang bisa digunakan, yaitu (*popsize*; pc; pm) = (100; 0.6; 0.001).
2. Jika permasalahan yang sedang diselesaikan memilih rata-rata *fitness* setiap generasi sebagai indikator utamanya, maka Grefenstette mempunyai rekomendasi untuk menggunakan nilai parameter yang berupa (*popsize*; pc; pm) = (30; 0.95; 0.005).
3. Jika permasalahan yang sedang diselesaikan memilih untuk memantau nilai *fitness* dari individu terbaik untuk setiap generasinya, maka rekomendasi yang diberikan mengenai penentuan nilai parameter yang digunakan adalah (*popsize*; pc; pm) = (80; 0.45; 0.01).
4. Ukuran populasi yang seharusnya digunakan untuk sembarang jenis permasalahan yang akan diselesaikan tidak diperbolehkan lebih kecil dari 30.

2.8.3 Penerapan *Genetic Algorithm* (GA)

2.8.3.1 Membangkitkan Populasi Awal

Populasi merupakan kumpulan beberapa kromosom. Langkah pertama yang dilakukan adalah membangkitkan populasi awal dengan cara membangkitkan bilangan acak sehingga didapatkan solusi awal (Aribowo, Lukas, & Gunawan, 2008). Populasi ini terdiri dari sejumlah kromosom dan untuk membangkitkan populasi dibutuhkan *popsize* untuk menentukan jumlah individu.

2.8.3.2 Representasi Kromosom

Representasi kromosom adalah proses pengkodean yang merupakan kunci pokok persoalan menggunakan *Genetic Algorithm* dari penyelesaian asli suatu permasalahan (Mahmudy, 2013). Proses pengkodean sangat berkaitan dengan peran kromosom sebagai representasi penyelesaian masalah. Pada prinsipnya tidak ada aturan khusus tentang pengkodean dalam proses *Genetic Algorithm* ini, kromosom dapat dibuat dengan kode tertentu dengan syarat bisa diproses oleh operator genetika serta merupakan representasi penyelesaian masalah (Zukhri, 2014). Metode pengkodean diklasifikasikan berdasarkan jenis simbol yang digunakan sebagai suatu gen menurut Gen dan Cheng pada tahun 2000 dalam

buku Desiani tahun 2006, yaitu pengkodean biner, pengkodean riil, bilangan bulat dan struktur data. Pada contoh persoalan ini, pengkodean yang digunakan adalah pengkodean riil. Pengkodean riil tepat sekali jika digunakan pada masalah optimalisasi fungsi dan optimalisasi kendala (Pramesti, Mahmudy, & Indriati, 2015).

2.8.3.3 Inisialisasi Populasi Awal

Inisialisasi populasi awal adalah proses pembangkitan populasi awal secara acak sebanyak *popsize* yang telah ditentukan sebelumnya. Misalkan ditentukan *popsize* = 10, maka populasi awal yang akan dihasilkan diperlihatkan pada Tabel 2.5 berikut (Mahmudy, 2013):

Tabel 2.5 Populasi Inisialisasi

Parents	*Chromosome*		*Fitness*
	x_1	x_2	$f(x_1, x_2)$
P_1	1.4898	2.0944	19.8206
P_2	8.4917	2.5754	34.7058
P_3	1.4054	6.3035	20.6707
P_4	5.8114	5.0779	14.5624
P_5	-1.8461	1.7097	11.5858
P_6	4.0206	4.4355	24.7106
P_7	-0.1634	2.974	19.653
P_8	5.2742	0.7183	22.1813
P_9	9.4374	6.6919	12.4694
P_{10}	-4.5575	0.1679	28.4324

Sumber: (Mahmudy, 2013)

2.8.3.4 Reproduksi

Proses reproduksi terdiri dari 2 operator genetika, yaitu tukar silang (*crossover*) dan mutasi (*mutation*) yang masing-masing digunakan untuk menghasilkan keturunan (*offspring*) dari individu-individu yang ada di populasi (Mahmudy, 2013).

a. *Crossover*

Crossover adalah salah satu proses reproduksi yang dilakukan dengan cara menyilangkan dua kromosom yang dipilih secara acak sehingga didapat kromosom yang mempunyai karakteristik yang mirip dengan induknya (Aribowo, Lukas, & Gunawan, 2008). Metode *crossover* yang digunakan dalam kasus ini adalah *extended intermediate crossover*. Misalkan P_1 dan P_2 adalah dua kromosom yang telah dipilih secara acak untuk melakukan proses reproduksi *crossover*, maka *offspring* C_1 dan C_2 bisa dibangkitkan dengan menggunakan Persamaan 2-17 dan Persamaan 2-18 sebagai berikut (Mahmudy, 2013):

$$C_1 = P_1 + \alpha(P_2 - P_1) \quad (2\text{-}17)$$

$$C_2 = P_2 + \alpha(P_1 - P_2) \quad (2\text{-}18)$$

Denga ketentuan nilai α yang dipilih secara acak pada interval [-0.25, 1.25].

Misalkan yang terpilih sebagai induk adalah P_4 dan P_9, a = [0.1104, 1.2336] maka akan dihasilkan dua *offspring* (C_1 dan C_2) sebagai berikut (Mahmudy, 2013):

$$C_1: \quad x_1 = 5.8114 + 0.1104(9.4374 - 5.8114) = 6.2118$$
$$x_2 = 5.0779 + 1.2336(6.6919 - 5.0779) = 7.0690$$

$$C_2: \quad x_1 = 9.4374 + 0.1104(5.8114 - 9.4374) = 9.0370$$
$$x_2 = 6.6919 + 1.2336(5.0779 - 6.6919) = 4.7008$$

Jika ditentukan *crossover rate* = 0.4, maka ada 0.4 x 10 = 4 *offspring* yang dihasilkan. Setiap satu kali proses *crossover* akan menghasilkan dua buah *offspring*, sehingga proses *crossover* akan dilakukan sebanyak 2 kali. Anggap dua *offspring* berikutnya adalah C_3 dan C_4 (Mahmudy, 2013).

b. Mutasi

Proses mutasi merupakan proses dimana suatu gen akan mengalami penyimpangan dari kromosom induknya sehingga sifat kromosom anak tersebut akan mengalami perbedaan dari kromosom induknya. Banyaknya jumlah kromosom yang akan dimutasi tergantung dari probabilitas yang telah ditentukan nilainya (Aribowo, Lukas, & Gunawan, 2008). Metode mutasi yang digunakan dalam kasus ini adalah *random mutation*. Metode mutasi dilakukan dengan cara memilih satu induk secara acak kemudian melakukan proses penambahan atau pengurangan nilai gen yang telah terpilih. Misalkan domain variabel x_j adalah [min_j, max_j] dan *offspring* yang dihasilkan adalah C = [$x'_1..x'_n$], maka nilai gen *offspring* bisa dibangkitkan sebagai berikut (Mahmudy, 2013):

$$x'_i = x'_i + r(\max_i - \min_j) \qquad (2\text{-}19)$$

Range r misalkan [-0.1, 0.1]

Misalkan yang terpilih sebagai induk adalah P_2, gen yang terpilih nomer 2 (x_2) dan r = -0.0584. Maka akan dihasilkan *offspring* (C_5) sebagai berikut (Mahmudy, 2013):

$$C_5: \quad x_1 = 8.4917(tetap)$$
$$x_2 = 2.5754 - 0.0584(7.3 - 0.0) = 2.1491$$

Jika nilai *mutation rate* = 0.2, maka ada 0.2 x 10 = 2 *offspring* yang dihasilkan dari proses mutasi sehingga *offspring* yang dihasilkan berikutnya adalah C_6. Berikut adalah Tabel 2.6 Hasil *Crossover* dan Mutasi yang memperlihatkan tentang kromosom dari keseluruhan *offspring* yang dihasilkan pada proses reproduksi, yaitu *offspring* ketika proses *crossover* maupun *offspring* ketika proses mutasi sebagai berikut (Mahmudy, 2013):

Tabel 2.6 Hasil *Crossover* dan Mutasi

Childs	***Chromosome***		***Fitness***
	x_1	**x_2**	**$f(x_1, x_2)$**
C_1	6.2118	7.0690	22.2048

C_2	9.0370	4.7008	22.2313
C_3	7.1636	5.1370	15.4774
C_4	7.5479	7.3000	9.3531
C_5	8.4917	2.1494	31.0389
C_6	-1.1238	1.7097	12.0177

Sumber: (Mahmudy, 2013)

2.8.3.5 Perhitungan Fitness

Algoritma genetika dapat menyelesaikan masalah kompleks, yaitu permasalahan optimasi. Untuk mendapatkan nilai *fitness*, suatu individu harus dievaluasi berdasarkan suatu nilai fungsi objektif tertentu. Jika permasalahannya adalah optimasi untuk memaksimalkan fungsi h, maka digunakan fungsi objektif $f = h$ untuk mencari nilai *fitness*nya. Akan tetapi, jika permasalahannya adalah optimasi untuk meminimalkan fungsi h, maka digunakan fungsi objektif $f = 1/h$ untuk mencari nilai *fitness*nya. Rumus *fitness* diperlihatkan pada Persamaan 2-20 dan 2-21 berikut (Pramesti, Mahmudy, & Indriati, 2015).

$$f = h \tag{2-20}$$

$$f = \frac{1}{(h + a)} \tag{2-21}$$

Keterangan:
a = bilangan kecil yang sesuai dengan permasalahan yang dihadapi
h = fungsi yang akan dimaksimalkan atau diminimalkan
f = fungsi *fitness*

2.8.3.6 Seleksi

Proses seleksi adalah proses pemilihan individu dari keseluruhan populasi yang ada untuk dipertahankan pada generasi selanjutnya (Mahmudy, 2013). Tahap awal untuk melakukan seleksi, yaitu pencarian nilai *fitness*. Proses seleksi dilakukan guna mengetahui kromosom-kromosom yang memiliki nilai *fitness* tertinggi dalam satu populasi (Zamani, Amaliah, & Munif, 2012). Macam-macam metode seleksi yang biasanya digunakan adalah metode seleksi *roulette wheel*, *binary tournament* dan *elitism*. Metode yang digunakan dalam penyelesaian kasus ini adalah *elitism selection*, yaitu proses seleksi yang dilakukan dengan cara mengumpulkan semua individu kemudian diseleksi dengan mencari individu yang nilai *fitness*-nya tertinggi untuk dipertahankan hidup pada generasi selanjutnya (Mahmudy, 2013).

Pseudo-code seleksi *elitism* dideskripsikan sebagai berikut (Mahmudy, 2013):

```
PROCEDURE ElitismSelection
Input:
   POP: himpunan individu pada populasi
   Pop_size: ukuran populasi
   OS:  himpunan  individu  anak  (offspring)  hasil  reproduksi
        menggunakan crossover dan mutasi
```

```
Output:
   POP: himpunan individu pada populasi setelah proses seleksi
        selesai

/*gabungkan individu pada POP dan OS ke dalam TEMP */
TEM ← Merge (POP, OS)

/*urutkan individu berdasarkan fitness secara ascending */
OrderAscending (Temp)

/* copy pop_size individu terbaik ke POP */
POP ← CopyBest (Temp, pop_size)

END PROCEDURE
```

Gambar 2.5 *Pseudocode Elitism Selection*

Berikut adalah tabel tentang individu baru yang dihasilkan dari hasil seleksi menggunakan seleksi *elitism*.

Tabel 2.7 Individu Baru Hasil Seleksi *Elitism*

P(t+1)	Asal P(t)	*Fitness*
P_1	P_2	34.7058
P_2	C_5	31.0389
P_3	P_{10}	28.4324
P_4	P_6	24.7106
P_5	C_2	22.2313
P_6	C_1	22.2048
P_7	P_8	22.1813
P_8	P_3	20.6707
P_9	P_1	19.8206
P_{10}	P_7	19.653

2.9 *Random Injection*

Pada ruang pencarian yang tidak terlalu besar, sering dijumpai permasalahan tentang konvergensi dini. Hal tersebut terjadi dikarenakan individu yang telah dibangkitkan secara acak lebih cepat untuk mendapatkan nilai *fitness* yang sama antar individu sebelum tercapainya titik optimum yang diinginkan (Mahmudy, 2013). Untuk mengatasi hal tersebut dapat digunakan salah satu metode sederhana yang dapat diterapkan pada sistem untuk mengatasi masalah tersebut, yaitu dengan menggunakan metode *random injection*. *Random injection* adalah proses seleksi yang dilakukan dengan hanya memilih *popsize*-n individu. Individu yang terakhir akan dibangkitkan secara acak seperti pada saat inisialisasi populasi awal. Dengan adanya individu baru yang diacak, maka keragaman populasi yang dihasilkan akan tetap terjaga dikarenakan individu tersebut juga akan melakukan proses reproduksi. Pemasukan individu *random injection* dapat dilakukan terhadap setiap *g* interval generasi dimana nilai *g* tersebut ditentukan melalui percobaan pendahuluan.

2.10 *Artificial Neural Network* (ANN) dan *Genetic Algorithm* (GA)

Artificial Neural Network (ANN) merupakan salah satu algoritma yang digunakan untuk menyelesaikan masalah pemodelan yang kompleks dimana ANN memiliki kemampuan untuk dapat mempelajari data-data dari historikalnya. Sedangkan, *Genetic Algorithm* (GA) merupakan salah satu algoritma pencarian heuristik yang digunakan untuk menyelesaikan masalah optimasi yang kompleks dengan baik (Istadi & Rahmayanti, 2010). Salah satu contoh kasus yang diselesaikan dengan menggunakan penerapan ANN dan GA adalah kasus mengenai reaksi hidroksilasi benzena. ANN dan GA telah berhasil dalam memodelkan dan mengoptimasi suatu objek dengan kondisi tertentu hingga dihasilkan hasil optimum secara global. Tahapan dalam pemodelan dan optimasi menggunakan ANN dan GA sebagai berikut (Istadi & Rahmayanti, 2010):

1. Pengembangan pemodelan dengan ANN

   Pada tahapan ini dilakukan penentuan yang akan menjadi *input* dan *output* pada data percobaan yang akan dilatih dan diuji dengan menggunakan metode ANN berupa proses *feedforward* dan *backpropagation* yang akan menunjukkan hubungan *input* dan *output* yang lebih baik.

2. Pelatihan pemodelan ANN

   Tahapan ini akan melakukan proses pelatihan terhadap data latih yang telah dipilih hingga didapat hubungan antara *input* dan *output*.

3. Inisialisasi populasi

   Pada tahapan ini dilakukan inisialisasi populasi yang dihasilkan secara acak sebanyak jumlah populasi yang sudah ditetapkan. Melakukan pengaturan indeks generasi, jumlah populasi dan jumlah variabel bebas.

4. Perhitungan *fitness*

   Pada tahapan ini, dilakukan proses pelatihan terhadap vektor *input* dengan menggunakan pemodelan ANN. Kemudian, *output* yang dihasilkan akan kembali ditransformasikan menjadi nilai awal dan dimanfaatkan untuk melakukan proses perhitungan *fitness* setiap data.

5. Skala dari nilai *fitness*

   Tahapan ini dilakukan untuk memilih *parent* pada generasi selanjutnya.

6. Seleksi *parent*

   Tahapan ini dilakukan untuk melakukan proses pemilihan *parent* berdasarkan pada nilai skala. Individu dapat dipilih lebih dari satu kali sebagai *parent* dan individu yang memiliki skala yang tertinggi kemungkinan besar akan terpilih dan lanjut ke generasi selanjutnya.

7. Reproduksi *children*

   Pemilihan *parent* yang terpilih memiliki pengaruh yang besar terhadap pembuatan *children* pada algoritma genetika dalam generasi selanjutnya.

8. Perpindahan tempat *children* pada populasi

   Tahapan ini dilakukan dengan cara memindah tempatkan populasi dengan *children* berdasarkan jenis metode seleksi yang digunakan serta berdasarkan nilai *fitness* dari setiap individu hingga diperoleh individu-individu yang lolos ke generasi selanjutnya.

9. *Increment* indeks generasi

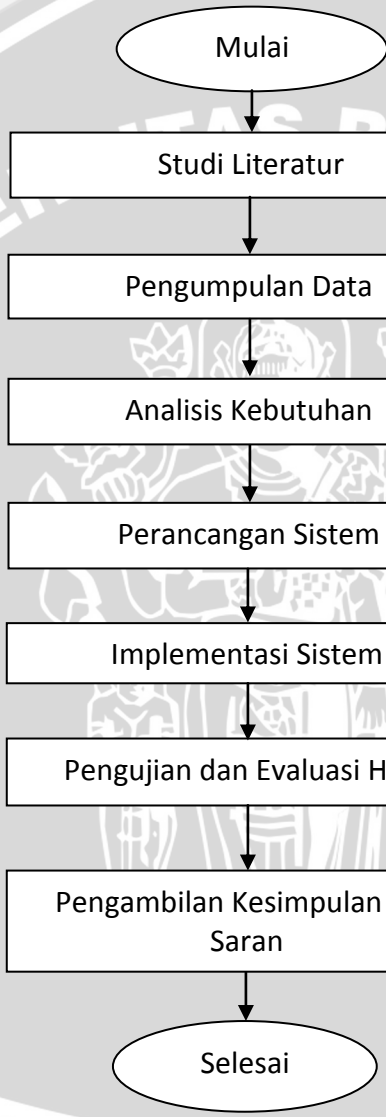
   Pada tahapan ini dilakukan penambahan indeks generasi hingga kondisi yang telah ditentukan dapat tercapai.

10. Mengulangi langkah dari nomer 4 sampai dengan nomer 9 hingga sistem akan memperoleh hasil yang konvergen

11. Memperoleh individu yang tertinggi (individu terbaik) sebagai vektor yang paling optimal dan akan dijadikan sebagai solusi untuk memecahkan masalah tertentu yang sedang diselesaikan

BAB 3 METODOLOGI PENELITIAN

Pada bab ini akan dibahas mengenai langkah-langkah yang digunakan dalam pembuatan sistem Pemodelan Komposisi Pakan Kambing Peranakan Etawa (PE) untuk Optimasi Kandungan Gizi Susu Menggunakan *Artificial Neural Network* (ANN) dan *Genetic Algorithm* (GA) dengan studi kasus pada UPT Pembibitan Ternak dan Hijauan Makanan Ternak Singosari-Malang. Adapun langkah-langkahnya dapat dilihat pada Gambar 3.1 Diagram Alir Metodologi Penelitian.



Gambar 3.1 Diagram Alir Metodologi Penelitian

Penjelasan dari setiap tahapan yang ada pada Gambar 3.1 Diagram Alir Metodologi Penelitian sebagai berikut:

1. Melakukan studi literatur tentang kambing Peranakan Etawa (PE), kandungan nutrisi pakan kambing PE, kandungan gizi susu kambing PE, *Artificial Neural Network* (ANN) dan *Genetic Algorithm* (GA) yang dibahas pada penelitian ini.

2. Melakukan pengumpulan data dengan berbagai cara, seperti melakukan wawancara, penyebaran kuisioner, membaca buku atau jurnal dan lain-lain.
3. Menganalisa kebutuhan sistem yang digunakan untuk membuat sistem Pemodelan Komposisi Pakan Kambing Peranakan Etawa (PE) untuk Optimasi Kandungan Gizi Susu Menggunakan *Artificial Neural Network* (ANN) dan G*enetic Algorithm* (GA).
4. Merancang sistem yang akan dibuat untuk menyelesaikan permasalahan Pemodelan Komposisi Pakan Kambing Peranakan Etawa (PE) untuk Optimasi Kandungan Gizi Susu Menggunakan *Artificial Neural Network* (ANN) dan *Genetic Algorithm* (GA).
5. Mengimplementasikan sistem berdasarkan hasil dari tahapan analisis kebutuhan sistem dan perancangan sistem yang telah dilakukan sebelumnya.
6. Melakukan pengujian dan evaluasi hasil terhadap hasil yang diperoleh dari msaing-masing uji coba tersebut.
7. Mengambil kesimpulan dan saran berdasarkan hasil yang diperoleh dari pengujian dan evaluasi hasil.

3.1 Studi Literatur

Studi literatur merupakan tahapan pada metodologi penelitian yang menjelaskan tentang dasar-dasar teori yang digunakan oleh penulis, yaitu mempelajari literatur dari beberapa bidang ilmu yang berhubungan dengan pembuatan sistem Pemodelan Komposisi Pakan Kambing Peranakan Etawa (PE) untuk Optimasi Kandungan Gizi Susu Menggunakan *Artificial Neural Network* (ANN) dan G*enetic Algorithm* (GA) yang diperoleh dari buku, jurnal, *website*, *e-book* dan penelitian sebelumnya. Teori-teori pendukung tersebut diantaranya:

1. Kambing Peranakan Etawa (PE)
2. Bahan pakan kambing Peranakan Etawa (PE)
3. Kandungan nutrisi pakan kambing Peranakan Etawa (PE)
4. Kandungan gizi susu kambing Peranakan Etawa (PE)
5. *Artificial Neural Network* (ANN) dengan algoritma *backpropagation*
6. *Genetic Algorithm* (GA)
7. Pemrograman bahasa Java

3.2 Pengumpulan Data

Teknik pengumpulan data yang digunakan oleh penulis adalah melakukan wawancara terhadap pihak UPT Pembibitan Ternak dan Hijauan Makanan Ternak Singosari-Malang dengan narasumber, yaitu Bapak Rokim, S.Pt bidang pakan ternak, Ibu drh. Jaya Wulandari dan Bapak Kayat bidang laboratorium. Selain itu teknik pengumpulan data lainnya yang digunakan oleh penulis adalah membaca jurnal-jurnal yang terkait dengan Pemodelan Komposisi Pakan Kambing

Peranakan Etawa (PE) untuk Optimasi Kandungan Gizi Susu Menggunakan *Artificial Neural Network* (ANN) dan *Genetic Algorithm* (GA). Data yang digunakan pada penelitian ini meliputi:

1. Data jenis pakan yang diberikan pada kambing Peranakan Etawa (PE)
2. Data kandungan nutrisi pakan untuk kambing Peranakan Etawa (PE)
3. Data hasil laboratorium untuk kandungan gizi susu kambing PE
4. Data *range* kandungan gizi susu kambing Peranakan Etawa (PE)

Analisis kebutuhan data beserta sumber, metode dan kegunaannya dijelaskan pada Tabel 3.1 Analisis Kebutuhan Data Penelitian sebagai berikut:

Tabel 3.1 Analisis Kebutuhan Data Penelitian

No.	Kebutuhan Data	Sumber Data	Metode	Kegunaan Data
1.	Data jenis-jenis pakan yang diberikan pada kambing PE	Jurnal, pakar tentang pakan, yaitu Bapak Rokim, S.Pt	Literatur dan wawancara	Sebagai masukan tentang jenis-jenis pakan yang akan dipilih oleh pengguna untuk diberikan pada kambing PE
2.	Data kandungan nutrisi pakan berdasarkan jenis-jenis pakan yang diberikan pada kambing PE	Literatur buku, brosur, pakar tentang pakan, yaitu Bapak Rokim, S.Pt	Literatur dan wawancara	Sebagai masukan yang akan digunakan untuk proses pelatihan ANN dan pengujian ANN serta perhitungan dalam proses GA untuk optimasi
3.	Data hasil laboratorium untuk kandungan gizi susu kambing PE	Pakar tentang kesehatan dan hasil uji laboratorium, yaitu Ibu drh. Jaya Wulandari dan Bapak Kayat	Wawancara	Digunakan pada proses pelatihan ANN sebagai target luaran
4.	Dara *range* kandungan gizi susu kambing PE	Jurnal, pakar tentang kesehatan dan hasil uji laboratorium, yaitu Ibu drh. Jaya Wulandari	Literatur dan wawancara	Digunakan pada proses pengujian ANN untuk dapat menentukan level kualitas susu kambing PE yang dihasilkan

3.3 Analisis Kebutuhan Sistem

Analisis Kebutuhan sistem adalah tahapan menganalisa hal-hal apa saja yang dibutuhkan untuk merancang dan membangun sistem sehingga sistem dapat berjalan sesuai dengan tujuan yang diinginkan.

3.3.1 Deskripsi Umum Sistem

Sistem yang dibangun pada penelitian ini adalah sistem yang mengimplementasikan *Artificial Neural Network* (ANN) dan *Genetic Algorithm* (GA) ke dalam permasalahan penyusunan komposisi bahan pakan pada kambing Peranakan Etawa (PE). Membuat komposisi pakan kambing Peranakan Etawa (PE) adalah pekerjaan yang tidak mudah dilakukan sehingga untuk mendapatkan kombinasi bahan pakan dengan kandungan nutrisi yang sesuai dan kandungan gizi susu kambing yang lebih optimal, maka dibutuhkan perhitungan dengan menggunakan metode *Artificial Neural Network* (ANN) dan *Genetic Algorithm* (GA) agar dapat memodelkan komposisi bahan pakan kambing PE.

Masukan yang dibutuhkan oleh sistem meliputi jenis pakan hijauan yang akan diberikan kepada kambing PE beserta presentase pemberian pakannya, bobot badan kambing PE serta kandungan gizi susu yang ingin dioptimalkan atau diminimalkan sesuai kebutuhan pengguna. *Output* yang dihasilkan oleh sistem adalah hasil rekomendasi komposisi pakan yang diberikan kepada kambing PE guna menghasilkan kandungan gizi susu yang lebih optimal. *Artificial Neural Network* (ANN) dan *Genetic Algorithm* (GA) diharapkan dapat memberikan solusi terbaik dalam pembuatan sistem pemodelan komposisi pakan kambing Peranakan Etawa (PE) untuk optimasi kandungan gizi susu.

3.3.2 Kebutuhan Perangkat

Pada sub bab kebutuhan perangkat ini akan membahas tentang perangkat keras (*hardware*) dan perangkat lunak (*software*) yang digunakan untuk mengimplementasikan sistem yang akan dibuat.

3.3.2.1 Perangkat Keras

Perangkat keras yang digunakan dalam pembuatan sistem ini antara lain:

1. Prosesor intel® Core™ i3-2328M 2.2GHz
2. RAM 4.00 GB
3. *Harddisk* 500 GB
4. Monitor 14”

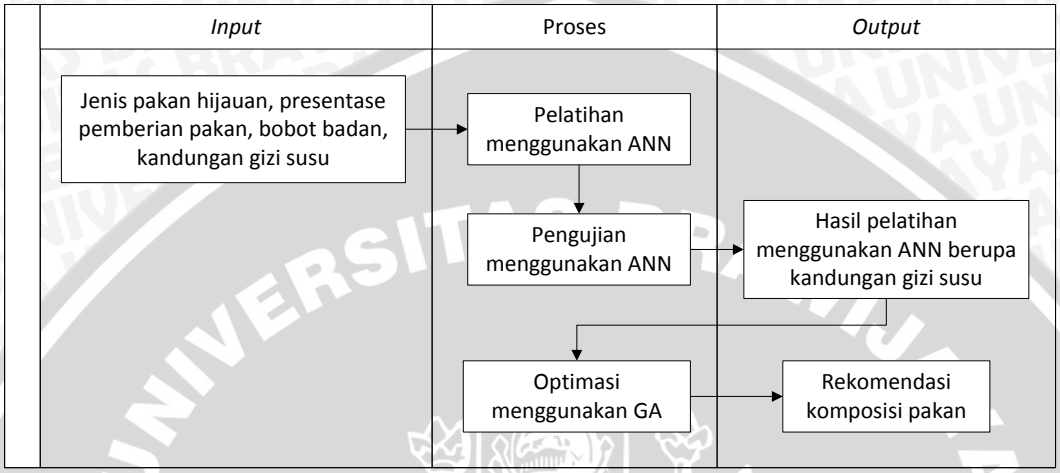
3.3.2.2 Perangkat Lunak

Perangkat lunak yang digunakan dalam pembuatan sistem ini antara lain:

1. Sistem operasi Windows 8.1 64 bit
2. *Microsoft Office* 2007
3. *Microsoft Visio* 2007
4. Netbeans IDE 8.0.2
5. Pencil

3.4 Perancangan Sistem

Tahapan perancangan sistem dibuat berdasarkan hasil dari analisis kebutuhan sistem. Perancangan sistem untuk Pemodelan Komposisi Pakan Kambing Peranakan Etawa (PE) untuk Optimasi Kandungan Gizi Susu Menggunakan *Artificial Neural Network* (ANN) dan *Genetic Algorithm* (GA) akan dijelaskan pada Gambar 3.2 Diagram Blok Sistem.



Gambar 3.2 Diagram Blok Sistem

Pada Gambar 3.2 Diagram Blok Sistem, yang akan dilakukan oleh pengguna adalah memasukkan satu atau dua jenis pakan hijauan yang digunakan, presentase pemberian pakan setiap jenis pakan hijauannya, bobot badan kambing dan kandungan gizi susu yang ingin dioptimalkan atau diminimalkan. Sistem melakukan proses pelatihan dan pengujian menggunakan ANN hingga diperoleh hasil pelatihannya yang berupa kandungan gizi susu. Kemudian, sistem akan melakukan proses optimasi GA dengan menggunakan hasil pelatihan dari ANN yang telah didapatkan pada proses sebelumnya hingga diperoleh hasil akhir dari sistem yang berupa rekomendasi komposisi pakan.

3.5 Implementasi Sistem

Implementasi sistem adalah tahapan membangun sistem yang mengacu pada hasil analisis kebutuhan sistem dan perancangan sistem pada tahapan sebelumnya mengenai Pemodelan Komposisi Pakan Kambing Peranakan Etawa (PE) untuk Optimasi Kandungan Gizi Susu Menggunakan *Artificial Neural Network* (ANN) dan *Genetic Algorithm* (GA). Implementasi dari penelitian ini akan dibangun menggunakan Bahasa pemrograman Java. Berikut adalah tahapan-tahapan yang ada di dalam implementasi sistem antara lain:

1. Implementasi antarmuka (*interface*).
2. Implementasi perhitungan algoritma ANN dengan data latih dan data uji untuk mendapatkan hasil kandungan gizi susu.
3. Implementasi perhitungan algoritma GA untuk mengoptimalkan nilai *fitness* yang dipilih dari ANN hingga menghasilkan rekomendasi komposisi pakannya.

3.6 Pengujian dan Evaluasi Hasil

Pengujian sistem dan evaluasi hasil dilakukan untuk mengetahui apakah sistem berjalan dengan baik dan sesuai dengan spesifikasi kebutuhan yang telah ditetapkan serta untuk mengetahui tingkat akurasi dari penerapan metode *Artificial Neural Network* (ANN) dan *Genetic Algorithm* (GA) dalam sistem. Hasil dari uji coba tersebut akan dievaluasi agar sesuai dengan tujuan yang diharapkan. Pengujian dilakukan dengan cara melakukan pelatihan dan pengujian pada ANN dengan menggunakan tiga parameter ANN, yaitu jumlah iterasi maksimum, jumlah *hidden layer* dan nilai *learning rate*. Pengujian tersebut dilakukan guna menentukan jumlah iterasi maksimum, jumlah *hidden layer* dan nilai *learning rate* yang optimal sehingga dapat diambil sebuah konfigurasi ANN yang terbaik untuk digunakan dalam memodelkan komposisi pakan kambing PE berdasarkan dengan skenario pengujian yang telah dibuat.

Pengujian juga dilakukan dengan cara melakukan optimasi pada GA dengan menggunakan tiga parameter GA, yaitu ukuran populasi, jumlah generasi dan kombinasi dari *crossover rate* (*cr*) dan *mutation rate* (*mr*). Pengujian tersebut dilakukan guna menentukan ukuran populasi, jumlah generasi dan kombinasi dari *cr* dan *mr* yang optimal, sehingga dapat diambil sebuah konfigurasi GA yang terbaik untuk digunakan dalam mengoptimasi kandungan gizi susu kambing PE berdasarkan dengan skenario pengujian yang telah dibuat. Selain itu, pengujian juga dilakukan terhadap kombinasi variasi *output* serta kenaikan protein dan penurunan lemak.

3.7 Pengambilan Kesimpulan dan Saran

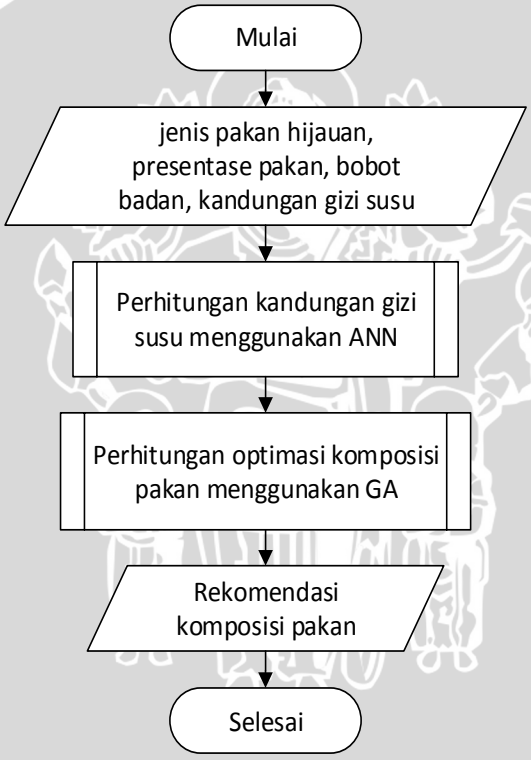
Tahapan metodologi penelitian yang terakhir, yaitu pengambilan kesimpulan dan saran. Pengambilan kesimpulan dilakukan setelah semua tahapan selesai dilakukan. Tahapan-tahapan tersebut meliputi tahapan analisis kebutuhan sistem, perancangan sistem, implementasi sistem serta pengujian dan evaluasi hasil. Kesimpulan yang diambil akan mengacu pada hasil pengujian dan evaluasi hasil sistem dengan tujuan untuk menjawab pertanyaan-pertanyaan yang telah dirumuskan sebelumnya pada Bab 1, sub bab 1.2 rumusan masalah. Selain pengambilan kesimpulan, juga akan ditambahkan saran yang berguna untuk memperbaiki kesalahan dan memberikan pertimbangan untuk pengembangan lebih lanjut.

BAB 4 PERANCANGAN

Pada bab ini akan menjelaskan mengenai alir perancangan sistem yang berupa diagram alir dari algoritma *Artificial Neural Network* (ANN) dan *Genetic Algorithm* (GA), formulasi permasalahan dan perhitungan manualisasi *Artificial Neural Network* (ANN) dan *Genetic Algorithm* (GA) yang berupa perhitungan manual ANN dan perhitungan manual GA, perancangan antarmuka serta perancangan pengujian algoritma.

4.1 Alir Perancangan Sistem

Alir perancangan sistem menggunakan *Artificial Neural Network* (ANN) dan *Genetic Algorithm* (GA) merupakan urutan penyelesaian masalah menggunakan ANN dan GA secara sekuensial. Berikut adalah Gambar 4.1 Diagram Alir Perancangan Sistem yang akan dilakukan secara bertahap.



Gambar 4.1 Diagram Alir Perancangan Sistem

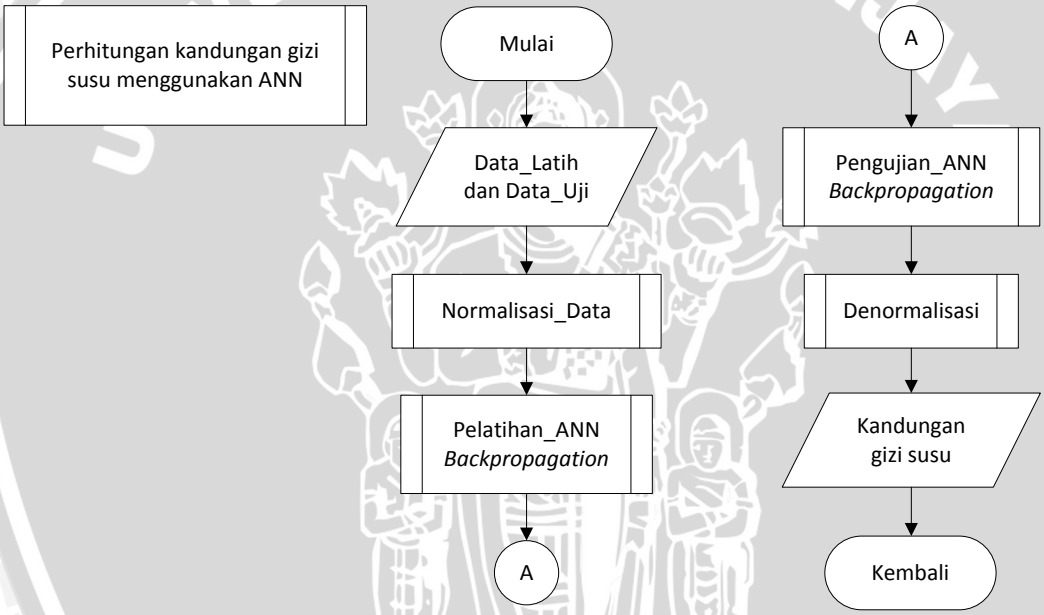
Pada Gambar 4.1 Diagram Alir Perancangan Sistem menjelaskan tentang alur kerja sistem untuk membuat pemodelan komposisi pakan kambing PE guna mengoptimasi kandungan gizi susu menggunakan ANN dan GA sebagai berikut:

1. Sistem membutuhkan beberapa masukan (*input*) dari pengguna berupa pemilihan jenis pakan hijauan dengan ketentuan minimal satu pakan dan maksimal dua pakan, presentase pemberian pakan, bobot badan kambing serta memilih kandungan gizi susu yang ingin dioptimalkan atau diminimalkan.

2. Sistem akan melakukan proses perhitungan kandungan gizi susu yang meliputi penentuan target, bobot dan pengujian dari pelatihan ANN menggunakan *backpropagation* hingga diperoleh hasil pemodelan komposisi pakan yang terbaik berdasarkan pakan hijauan yang dipilih.
3. Sistem akan melakukan proses perhitungan optimasi komposisi pakan menggunakan GA dengan mengoptimalkan atau meminimalkan kandungan gizi susu yang tersedia.
4. Sistem akan menghasilkan *output* yang berupa hasil rekomendasi komposisi pakan terbaik.

4.1.1 Perhitungan Kandungan Gizi Susu Menggunakan ANN

Proses perhitungan kandungan gizi susu dilakukan menggunakan ANN dengan menerapkan *backpropagation* di dalam sistem seperti yang diperlihatkan pada Gambar 4.2 di bawah ini.



Gambar 4.2 Diagram Alir Proses ANN *Backpropagation*

Pada Gambar 4.2 Diagram Alir Proses ANN *Backpropagation* menjelaskan tentang alur kerja dari ANN *backpropagation* yang dilakukan dengan tahapan-tahapan sebagai berikut:

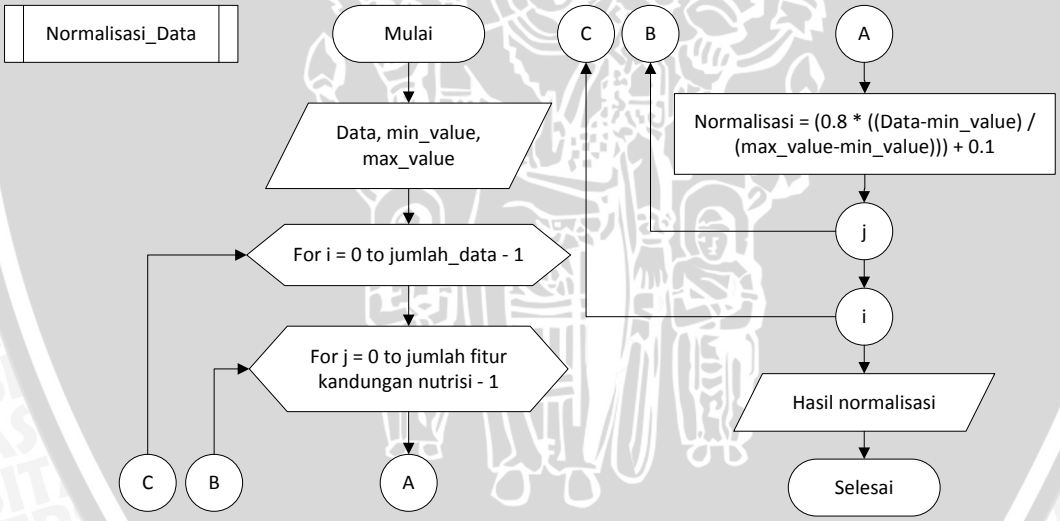
1. Masukan data latih yang belum dinormalisasi yang akan digunakan pada proses pelatihan ANN *backpropagation* dan data uji yang belum dinormalisasi yang akan digunakan pada proses pengujian ANN *backpropagation*.
2. Sistem melakukan proses normalisasi data, baik data latih ataupun data uji dengan rentang nilai antara 0.1-0.9. Perhitungan proses normalisasi data telah diperlihatkan pada Persamaan 2-15.
3. Sistem melakukan proses pelatihan terhadap data latih menggunakan ANN *backpropagation* hingga ditemukan bobot terbaik yang digunakan untuk

menentukan target atau *output* yang diharapkan dari kandungan gizi susu yang ingin dioptimalkan atau diminimalkan.

4. Sistem melakukan proses pengujian terhadap data uji menggunakan ANN *backpropagation* untuk membuktikan bahwa bobot terakhir yang dihasilkan dari pelatihan ANN *backpropagation* sudah menghasilkan *output* yang terbaik atau mendekati nilai target.
5. Sistem melakukan proses denormalisasi terhadap *output* yang dihasilkan pada proses pengujian ANN *backpropagation* tentang kandungan gizi susu untuk mengembalikan nilainya ke rentang semula. Perhitungan proses denormalisasi telah diperlihatkan pada Persamaan 2-16.
6. Sistem memberikan hasil *output* yang berupa kandungan gizi susu dengan nilai MSE terkecil.

4.1.1.1 Proses Normalisasi Data

Normalisasi data adalah proses perubahan nilai data yang berupa data latih dan data uji dimana nilai atribut data tersebut akan diskalakan dengan rentang antara 0.1 sampai 0.9. Proses normalisasi data yang digunakan pada penelitian ini adalah metode normalisasi *Min-Max*. Diagram alir proses normalisasi data diperlihatkan pada Gambar 4.3 sebagai berikut.



Gambar 4.3 Diagram Alir Proses Normalisasi Data

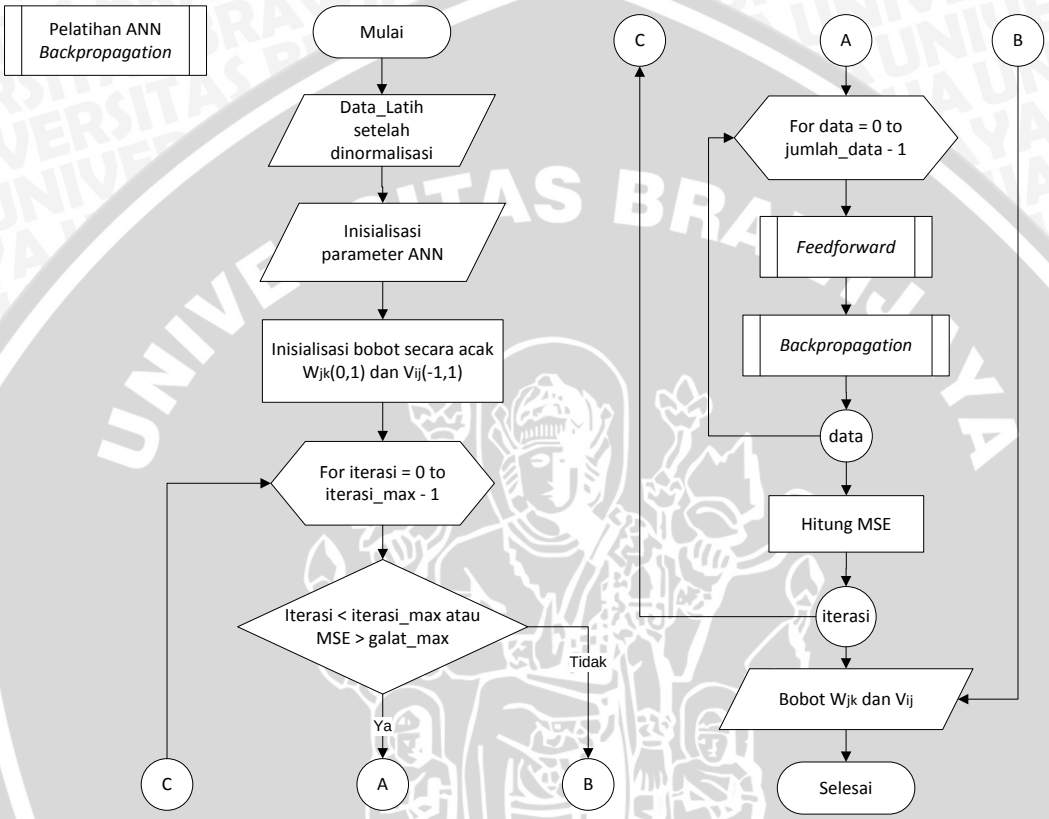
Pada Gambar 4.3 Diagram Alir Proses Normalisasi Data dilakukan dengan tahapan-tahapan sebagai berikut:

1. Masukan data yang berupa data latih atau data uji yang belum dinormalisasi, *min_value* dan *max_value*.
2. Sistem melakukan pengulangan sebanyak jumlah data yang telah dipilih sebelumnya, baik data yang berupa data latih ataupun data uji.
3. Sistem melakukan pengulangan sebanyak jumlah fitur kandungan nutrisi yang ada pada bahan pakan seperti abu, protein kasar (PK), lemak kasar (LK) dan serat kasar (SK).

4. Sistem melakukan proses normalisasi data menggunakan Persamaan 2-14.
5. Sistem akan menghasilkan hasil normalisasi dari data latih ataupun data uji.

4.1.1.2 Proses Pelatihan ANN Backpropagation

Berikut adalah gambar diagram alir yang menjelaskan tentang proses pelatihan ANN *backpropagation* secara bertahap yang diperlihatkan pada Gambar 4.4 Diagram Alir Proses Pelatihan ANN *Backpropagation* sebagai berikut.



Gambar 4.4 Diagram Alir Proses Pelatihan ANN *Backpropagation*

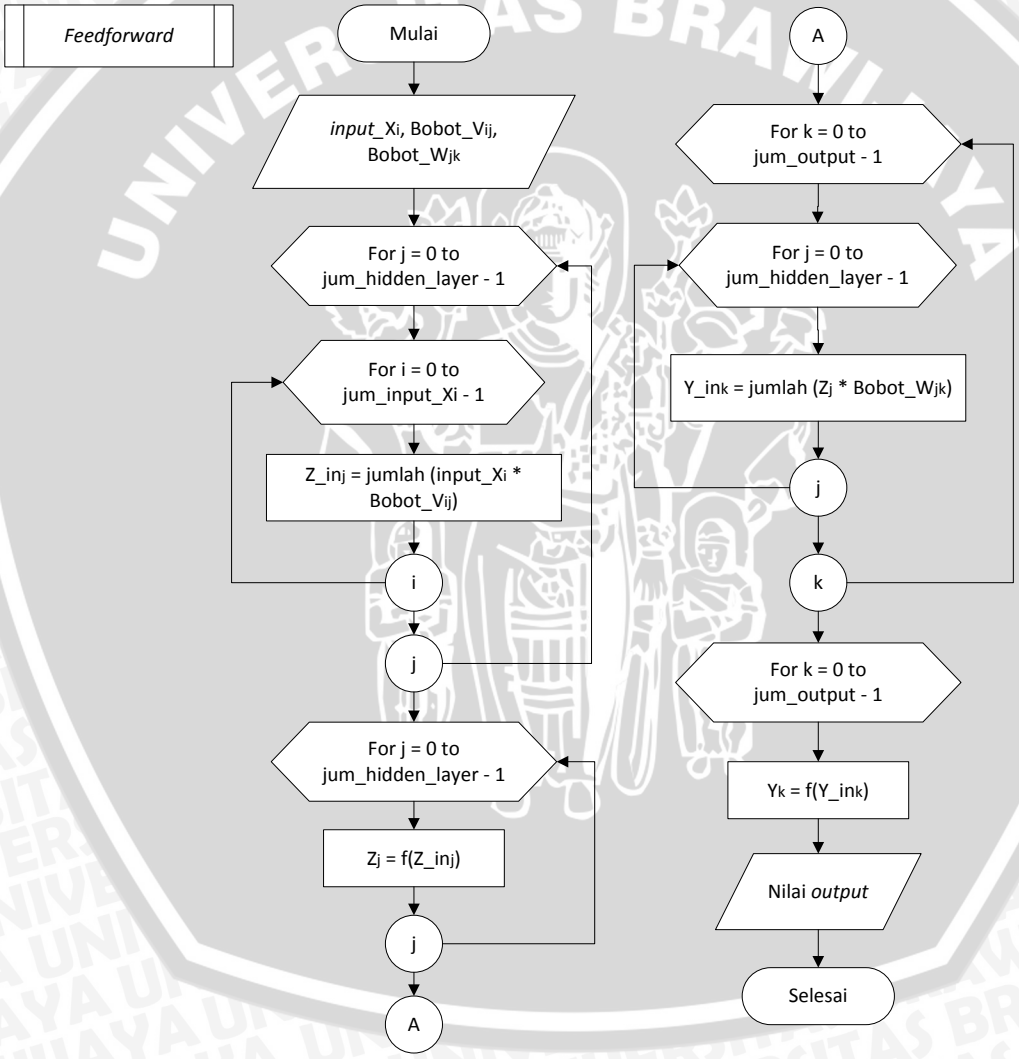
Pada Gambar 4.4 Diagram Alir Proses Pelatihan ANN *Backpropagation* dilakukan dengan tahapan-tahapan sebagai berikut:

1. Masukan data latih yang telah melewati proses normalisasi data.
2. Masukan parameter inisialisasi parameter yang dimiliki oleh ANN *backpropagation* yang berupa jumlah iterasi maksimal, jumlah varian *hidden layer*, nilai *learning rate* dan besarnya nilai galat maksimal (epsilon).
3. Sistem membangkitkan bobot *neuron* secara *random*, yaitu bobot *neuron input* dibangkitkan antara *range* -1 sampai 1 dan bobot *neuron output* dibangkitkan antara *range* 0 sampai 1.
4. Sistem melakukan pengulangan sebanyak iterasi_max yang ditetapkan.
5. Sistem melakukan percabangan, jika iterasi masih < iterasi_max atau MSE > galat_max, maka lakukan langkah selanjutnya. Sedangkan, jika iterasi atau MSE sudah tercapai, maka lakukan langkah ke-9.

6. Sistem melakukan pengulangan sebanyak jumlah data yang dimasukan.
7. Sistem melakukan proses *feedforward* dan proses *backpropagation*.
8. Sistem menghitung nilai MSE dengan menggunakan Persamaan 2-17.
9. Sistem menghasilkan *output* berupa bobot v_{ij} dan w_{jk} yang terakhir.

4.1.1.3 Proses Feedforward pada ANN

Pada proses *feedforward*, setiap unit *input* menerima sebuah masukan sinyal dan melanjutkan ke *hidden layer* serta melakukan proses perhitungan aktivasi. Kemudian, hasilnya akan dikirimkan ke unit *output* dan menghitung aktivasinya untuk memberikan pola masukan. Berikut adalah diagram alir dari proses *feedforward* pada ANN yang diperlihatkan pada Gambar 4.5 di bawah ini.



Gambar 4.5 Diagram Alir Proses *Feedforward* pada ANN

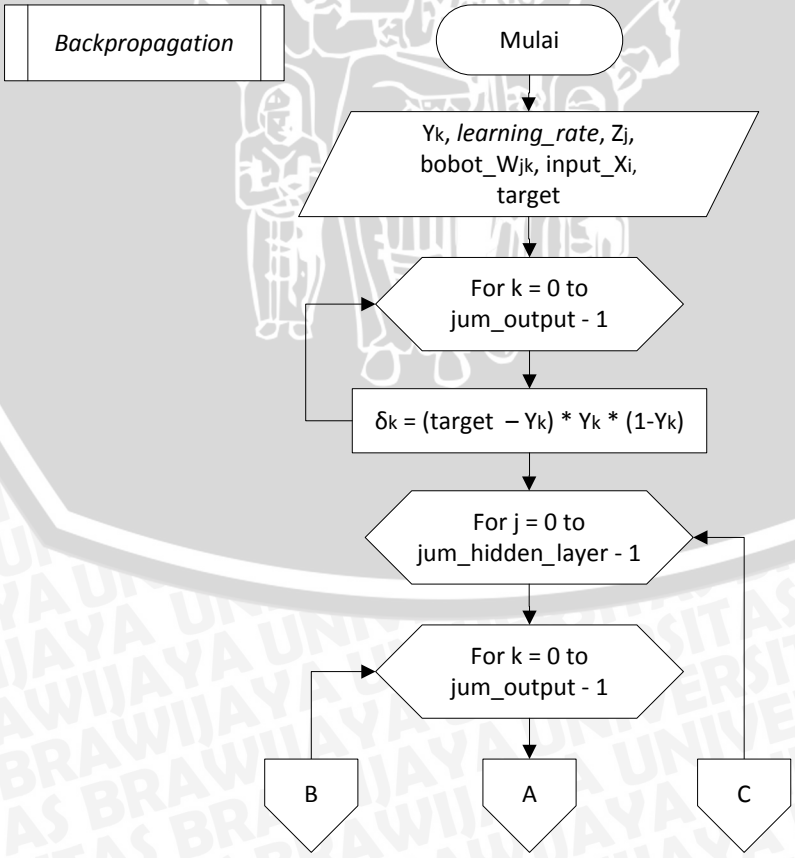
Pada Gambar 4.5 Diagram Alir Proses *Feedforward* pada ANN dilakukan dengan tahapan-tahapan sebagai berikut:

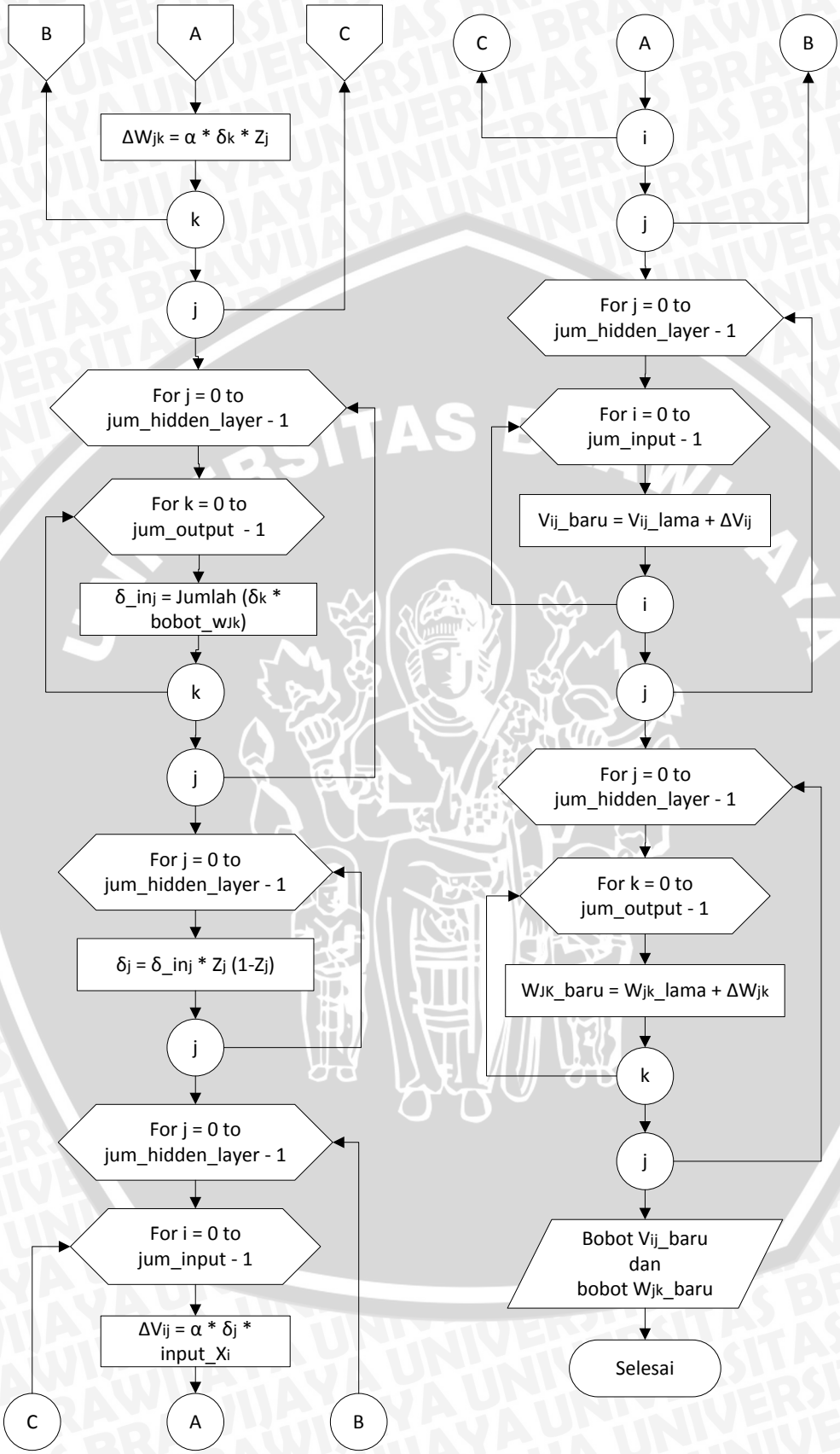
1. Masukan unit *input* X_i sebagai data *input*, bobot V_{ij} dan bobot W_{jk}.

2. Sistem melakukan pengulangan sebanyak jumlah *hidden layer* dan pengulangan sebanyak jumlah *input* X_i yang telah dimasukan sebelumnya.
3. Sistem menjumlahkan bobot sinyal *input* dengan melakukan proses perhitungan terhadap nilai Z_in_j dengan menggunakan Persamaan 2-3.
4. Sistem melakukan pengulangan sebanyak jumlah *hidden layer*.
5. Sistem menghitung fungsi aktivasi sigmoid biner pada *hidden layer* (Z_j) dengan menggunakan Persamaan 2-4.
6. Sistem melakukan pengulangan sebanyak jumlah *output layer* yang telah ditetapkan sebelumnya dan pengulangan sebanyak jumlah *hidden layer*.
7. Sistem menjumlahkan bobot sinyal *output* dengan melakukan proses perhitungan terhadap nilai Y_in_k dengan menggunakan Persamaan 2-5.
8. Sistem melakukan pengulangan sebanyak jumlah *output layer*.
9. Sistem menghitung fungsi aktivasi sigmoid biner pada *output layer* (Y_k) dengan menggunakan Persamaan 2-6.
10. Sistem menghasilkan nilai *output* dari proses *feedforward* seperti nilai Z_in_j, Z_j, Y_in_k dan Y_k.

4.1.1.4 Proses Backpropagation pada ANN

Berikut adalah gambar diagram alir yang menjelaskan tentang proses *backpropagation* ANN secara bertahap yang diperlihatkan pada Gambar 4.6.





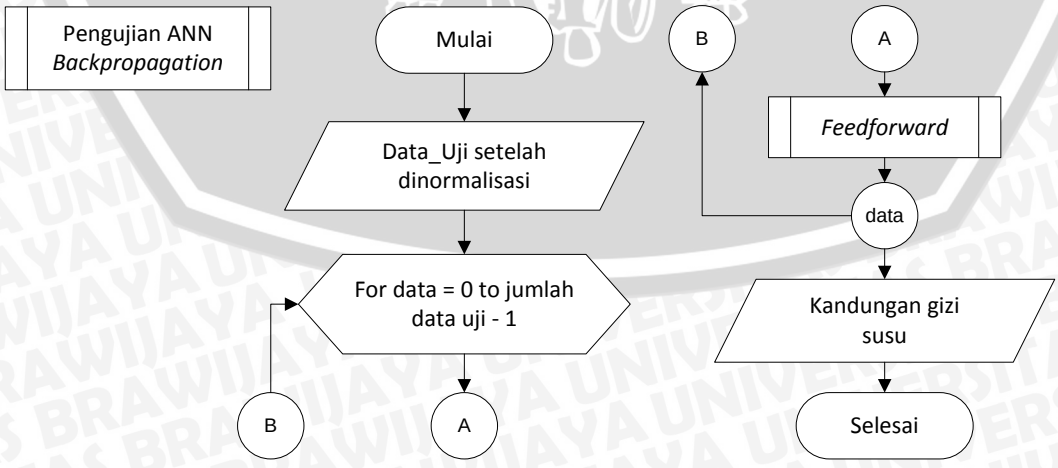
Gambar 4.6 Diagram Alir Proses *Backpropagation* pada ANN

Pada Gambar 4.6 Diagram Alir Proses *Backpropagation* pada ANN dilakukan dengan tahapan-tahapan sebagai berikut:

1. Masukan Y_k, *learning rate*, Z_j, bobot W_{jk}, *input* X_i dan *target* sebagai *input*.
2. Sistem melakukan pengulangan sebanyak jumlah *output layer* kemudian menghitung *error output* dengan menggunakan Persamaan 2-7.
3. Sistem melakukan pengulangan sebanyak jumlah *hidden layer* dan pengulangan sebanyak jumlah *output layer*.
4. Sistem akan melakukan proses perhitungan perubahan bobot pada *output layer* dengan menggunakan Persamaan 2-8.
5. Lakukan langkah ke-3, kemudian menjumlahkan setiap *error* pada *output layer* terhadap *hidden layer* dengan menggunakan Persamaan 2-9.
6. Sistem melakukan pengulangan sebanyak jumlah *hidden layer* dan menghitung informasi *error* dengan menggunakan Persamaan 2-10.
7. Sistem melakukan pengulangan sebanyak jumlah *hidden layer* dan pengulangan sebanyak jumlah data *input* (*input* X_i).
8. Sistem menghitung perubahan bobot pada *hidden layer* dengan menggunakan Persamaan 2-11.
9. Lakukan langkah ke-7, kemudian menghitung bobot V_{ij} yang baru dengan menggunakan Persamaan 2-13.
10. Lakukan langkah ke-3, kemudian menghitung bobot W_{jk} yang baru dengan menggunakan Persamaan 2-12.
11. Sistem menghasilkan *output* berupa bobot V_{ij}_baru dan W_{jk}_baru.

4.1.1.5 Proses Pengujian ANN Backpropagation

Berikut adalah gambar diagram alir yang menjelaskan tentang proses pengujian ANN *backpropagation* secara bertahap yang diperlihatkan pada Gambar 4.7 sebagai berikut.



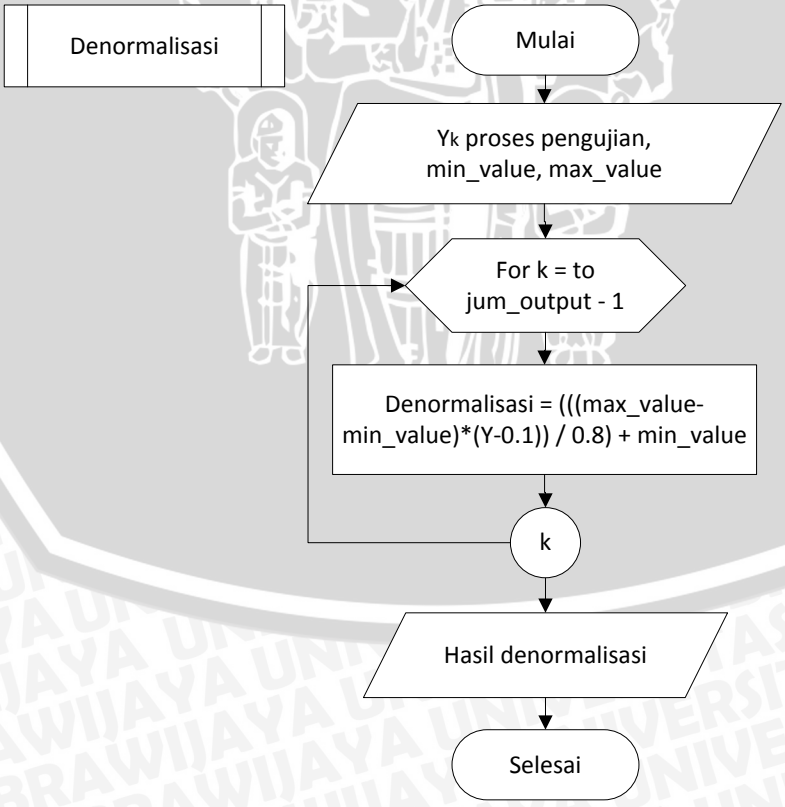
Gambar 4.7 Diagram Alir Proses Pengujian ANN *Backpropagation*

Proses pengujian ANN *backpropagation* dilakukan hanya pada tahapan *feedforward* saja. Bobot terakhir yang dihasilkan dari proses pelatihan ANN *backpropagation* akan digunakan pada proses pengujian ANN *backpropagation*. Pada tahap pengujian ini, jaringan diharapkan mampu mengenali pola berdasarkan data baru yang diberikan. Berikut adalah Gambar 4.7 Diagram Alir Proses Pengujian ANN *Backpropagation* sebagai berikut:

1. Masukan data uji yang sudah dinormalisasi.
2. Sistem melakukan pengulangan sebanyak jumlah data uji yang dimasukan.
3. Setiap data uji melakukan proses *feedforward* seperti pada sub bab 4.1.1.3.
4. Setelah pengulangan selesai, sistem menghasilkan *output* berupa kandungan gizi susu pada setiap data uji.

4.1.1.6 Proses Denormalisasi

Proses denormalisasi dilakukan untuk mengembalikan nilai kandungan gizi susu kambing PE ke dalam *range* aslinya. Perubahan tersebut dilakukan guna mempermudah dalam pemberian takaran kandungan nutrisi setiap komposisi pakan yang telah dipilih sebelumnya guna memperoleh hasil kandungan gizi susu yang diinginkan. Langkah-langkah proses denormalisasi sama dengan proses normalisasi, hanya saja yang membedakan adalah posisi pembagi dan pembilang dari masing-masing persamaan tersebut. Berikut adalah Gambar 4.8 Diagram Alir Proses Denormalisasi secara bertahap.



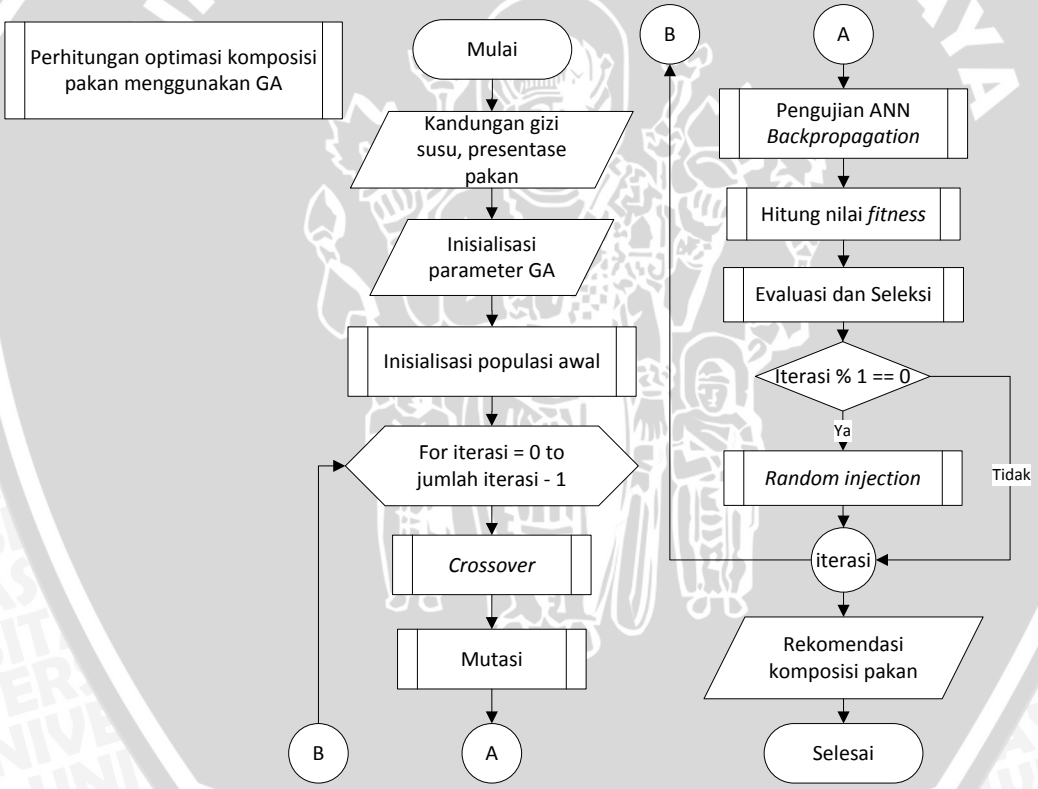
Gambar 4.8 Diagram Alir Proses Denormalisasi

Pada Gambar 4.8 Diagram Alir Proses Denormalisasi dilakukan dengan tahapan-tahapan sebagai berikut:

1. Masukan nilai Y_k dari hasil proses pengujian ANN *backpropagation*, *min_value* dan *max_value*.
2. Sistem melakukan pengulangan sebanyak jumlah *output layer* dan menghitung denormalisasi data dengan menggunakan Persamaan 2-15.
3. Sistem menghasilkan *output* hasil perhitungan denormalisasi data.

4.1.2 Perhitungan Optimasi Komposisi Pakan Menggunakan GA

Luaran yang dihasilkan ANN berupa nilai kandungan gizi susu kambing PE, GA akan mengoptimalkan nilai kandungan gizi susu yang ingin dioptimasi dan diminimalkan oleh pengguna. Hasil luaran yang akan diperoleh dari perhitungan optimasi menggunakan GA berupa komposisi pakan beserta kandungan nutrisi yang terbaik sesuai dengan kandungan gizi susu yang diharapkan. Berikut adalah Gambar 4.9 Diagram Alir Perhitungan Optimasi Menggunakan GA.



Gambar 4.9 Diagram Alir Perhitungan Optimasi Menggunakan GA

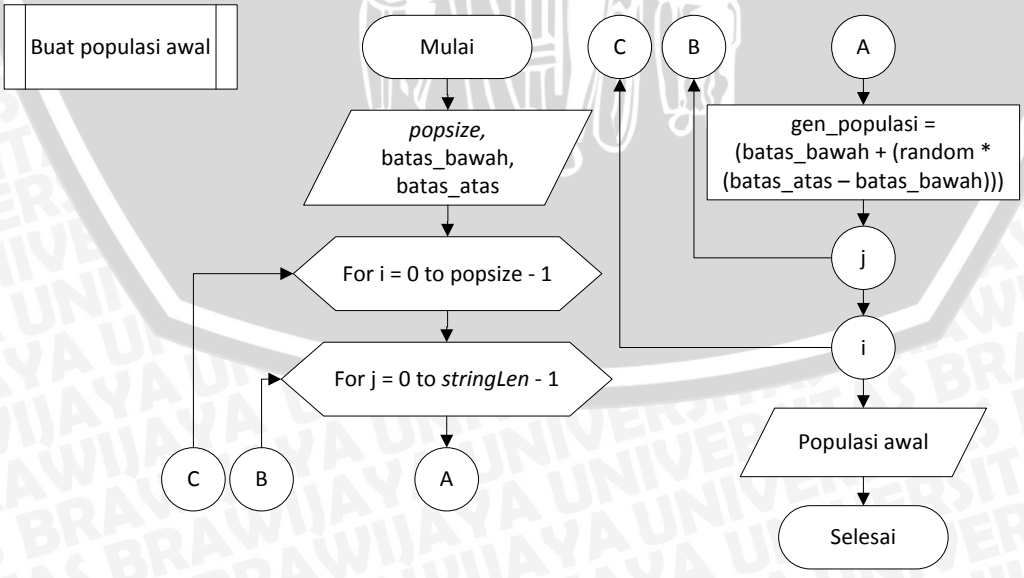
Pada Gambar 4.9 Diagram Alir Perhitungan Optimasi Menggunakan GA dilakukan dengan tahapan-tahapan sebagai berikut:

1. Masukan satu atau lebih kandungan gizi susu yang ingin dioptimalkan atau diminimalkan sesuai dengan model optimasi yang dipilih oleh pengguna pada tahapan sebelumnya serta presentase pemberian pakan yang digunakan untuk masing-masing pakan hijauan yang telah dipilih.

2. Masukan inisialisasi parameter GA, yaitu ukuran populasi (*popsize*), *crossover rate* (*cr*), *mutation rate* (*mr*) dan jumlah generasi (iterasi *max*).
3. Sistem melakukan proses inisialisasi populasi awal dengan membangkitkan populasi awal secara acak sejumlah *n-popsize* yang telah dimasukan.
4. Sistem melakukan pengulangan sebanyak jumlah iterasi yang dimasukan.
5. Sistem melakukan proses reproduksi *crossover* menggunakan *extended intermediate crossover* guna menghasilkan keturunan dengan menggunakan Persamaan 2-17 dan Persamaan 2-18.
6. Sistem melakukan proses reproduksi mutasi menggunakan *random mutation* guna menghasilkan keturunan dengan menggunakan Persamaan 2-19.
7. Sistem melakukan proses pengujian ANN yang ada pada sub bab 4.1.1.5
8. Sistem melakukan proses perhitungan nilai *fitness* semua individu (populasi dan *offspring*) dengan menggunakan Persamaan 2-20 dan Persamaan 2-21.
9. Sistem melakukan proses evaluasi dan seleksi menggunakan *elitism selection*.
10. Jika iterasi % 1 == 0 atau kelipatan iterasi 1, maka akan dilakukan proses *random injection*. Sedangkan, jika tidak maka populasi yang telah terpilih akan lanjut pada generasi selanjutnya tanpa dilakukan *random injection*.
11. Sistem menghasilkan *output* berupa rekomendasi komposisi pakan berdasarkan presentase pemberian pakan yang telah dimasukan.

4.1.2.1 Proses Inisialiasi Populasi Awal

Pada tahap ini akan dilakukan proses pembangkitan populasi awal sebanyak ukuran populasi yang telah ditentukan. Panjang kromosom sama dengan jumlah kandungan nutrisi bahan pakan hijauan yang telah dipilih sebelumnya. Berikut adalah Gambar 4.10 Diagram Alir Inisialisasi Populasi Awal.



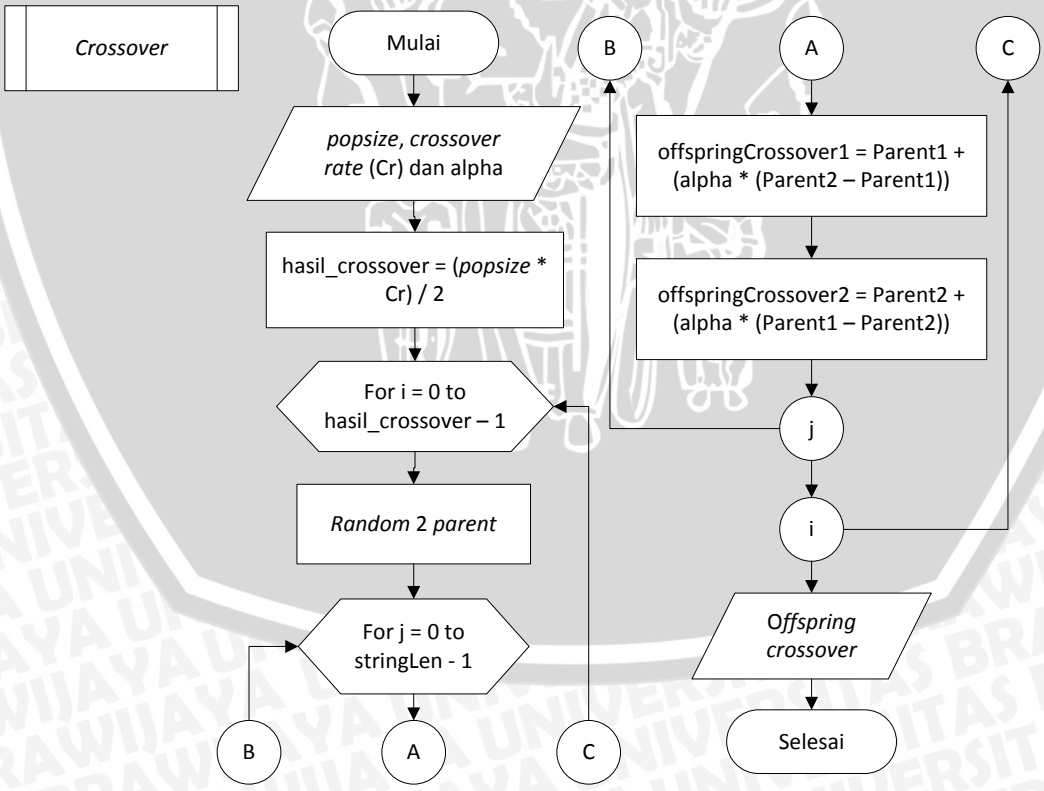
Gambar 4.10 Diagram Alir Inisialisasi Populasi Awal

Pada Gambar 4.10 Diagram Alir Inisialisasi Populasi Awal dilakukan dengan tahapan-tahapan sebagai berikut:

1. Masukan *popsize* yang diinginkan, nilai batas_bawah dan batas_atas yang didapatkan dari method `Batas_Bawah` dan `Batas_Atas` pada sistem.
2. Sistem melakukan pengulangan sebanyak *popsize* yang telah dimasukan dan melakukan pengulangan sebanyak jumlah kandungan nutrisi dari bahan pakan hijauan yang telah dipilih sebelumnya (*stringLen*).
3. Sistem melakukan pembangkitan populasi awal dengan menggunakan persamaan seperti yang diperlihatkan oleh diagram alir di atas secara acak.
4. Sistem menghasilkan *output* berupa populasi awal yang telah terbentuk secara acak.

4.1.2.2 Proses Reproduksi Crossover

Pada tahap ini, *extended intermediate crossover* adalah metode *crossover* yang digunakan untuk menghasilkan *offspring* dari kombinasi nilai dua induk yang terpilih secara acak. Jumlah *offspring* yang dihasilkan dalam proses reproduksi *crossover* bergantung pada nilai *crossover rate* (*cr*). Banyaknya *offspring* yang dihasilkan dari proses *crossover* adalah hasil perkalian antara *popsize* dengan *crossover rate*. Berikut adalah Gambar 4.11 Diagram Alir Proses Reproduksi *Crossover*.



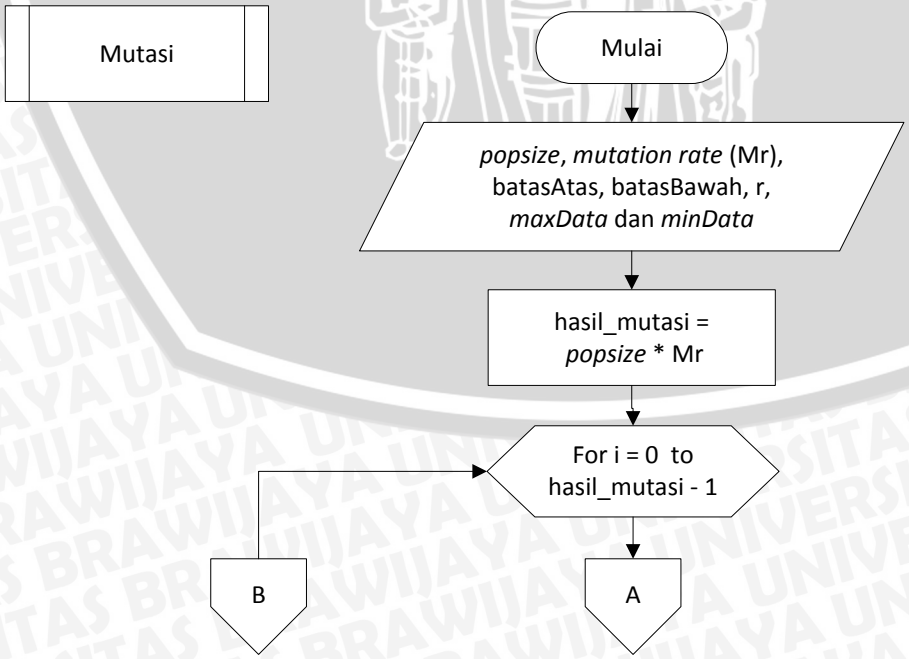
Gambar 4.11 Diagram Alir Proses Reprduksi *Crossover*

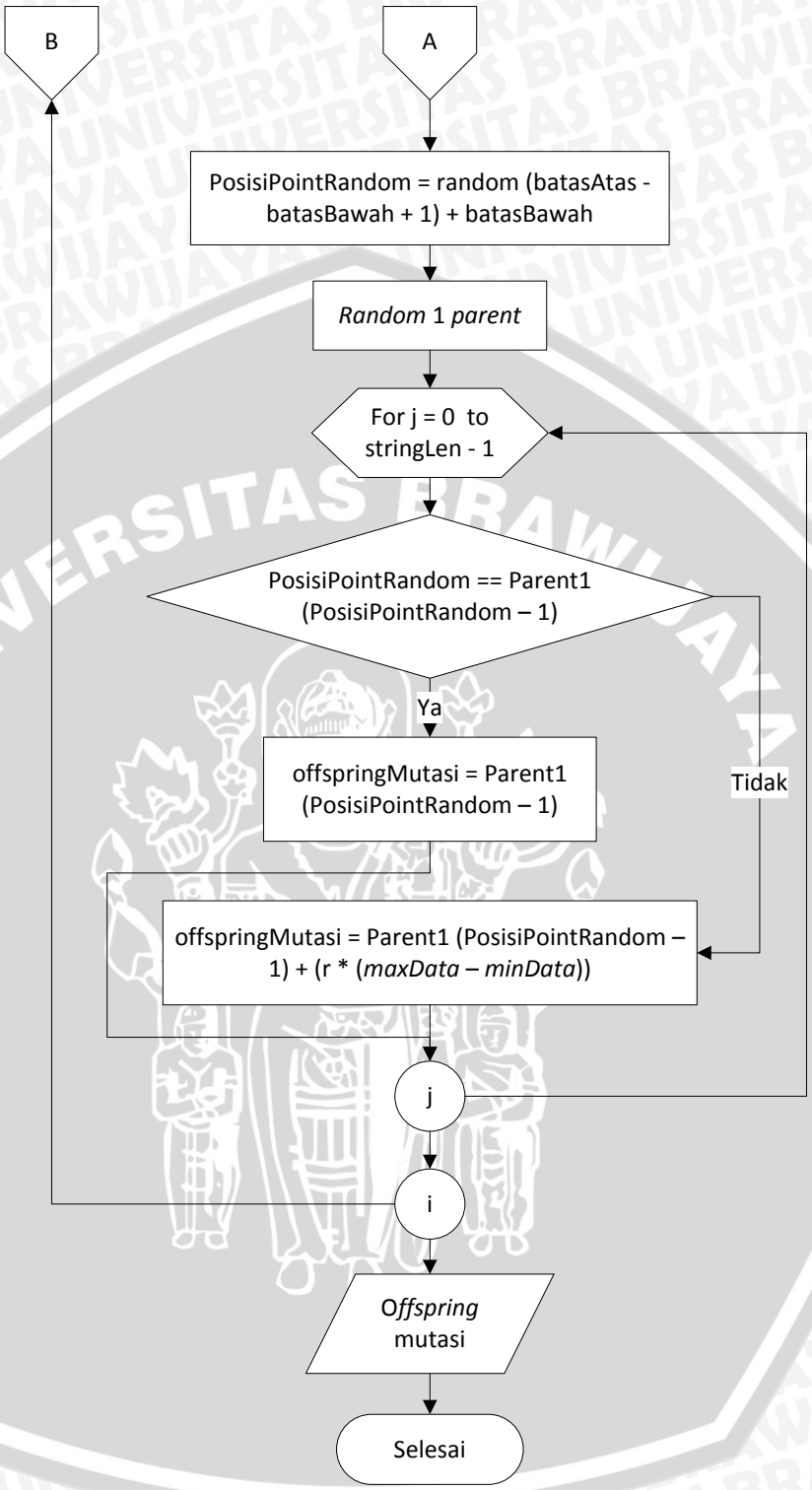
Pada Gambar 4.11 Diagram Alir Proses Reproduksi *Crossover* dilakukan dengan tahapan-tahapan sebagai berikut:

1. Masukan *popsize*, *crossover rate* (*cr*) dan nilai alpha yang diperoleh dari *method* `Nilai_Alpha`.
2. Sistem meghitung hasil *crossover* yang didapat, yaitu untuk mengetahui berapa kali proses *crossover* yang harus dilakukan dengan menggunakan persamaan yang terdapat pada diagram alir di atas.
3. Sistem melakukan pengulangan sebanyak hasil *crossover* yang didapatkan, kemudian melakukan pemilihan terhadap dua *parent* secara acak untuk dilakukan proses *crossover* nantinya.
4. Sistem melakukan pengulangan sebanyak panjang *stringLen* yang ditetapkan.
5. Ambil bobot dari *parent* 2 yang telah dipilih.
6. Sistem menghitung hasil *offspring crossover* pertama dengan menggunakan Persamaan 2-17 dan menghitung hasil *offspring crossover* kedua dengan menggunakan Persamaan 2-18.
7. Sistem menghasilkan *output* berupa *offspring* dari hasil *crossover* yang telah dilakukan.

4.1.2.3 Proses Reproduksi Mutasi

Pada tahap ini, *random mutation* merupakan metode mutasi yang digunakan pada penelitian ini. Metode mutasi ini dilakukan dengan cara melakukan penambahan atau pengurangan nilai gen yang terpilih (Mahmudy, 2013) seperti pada Persamaan 2-19. Banyaknya *offspring* yang dihasilkan pada proses reproduksi mutasi adalah hasil perkalian *popsize* dengan *mutation rate*. Berikut adalah Gambar 4.12 Diagram Alir Proses Reproduksi Mutasi.





Gambar 4.12 Diagram Alir Proses Reproduksi Mutasi

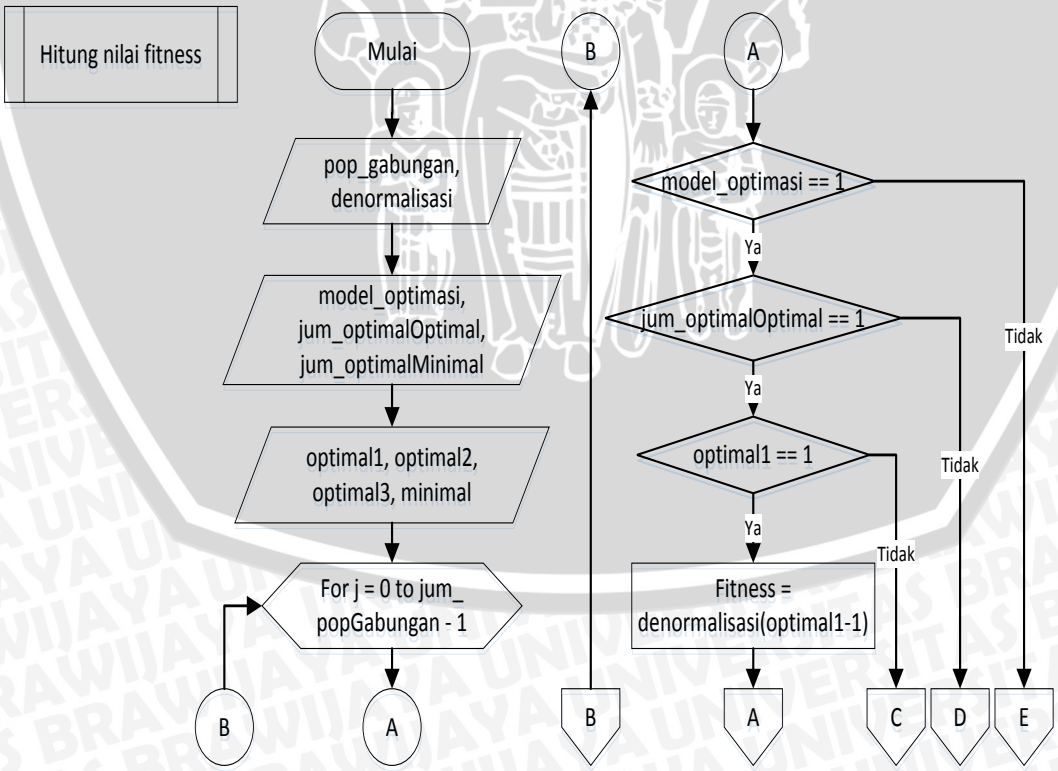
Pada Gambar 4.12 Diagram Alir Proses Reproduksi Mutasi dilakukan dengan tahapan-tahapan sebagai berikut:

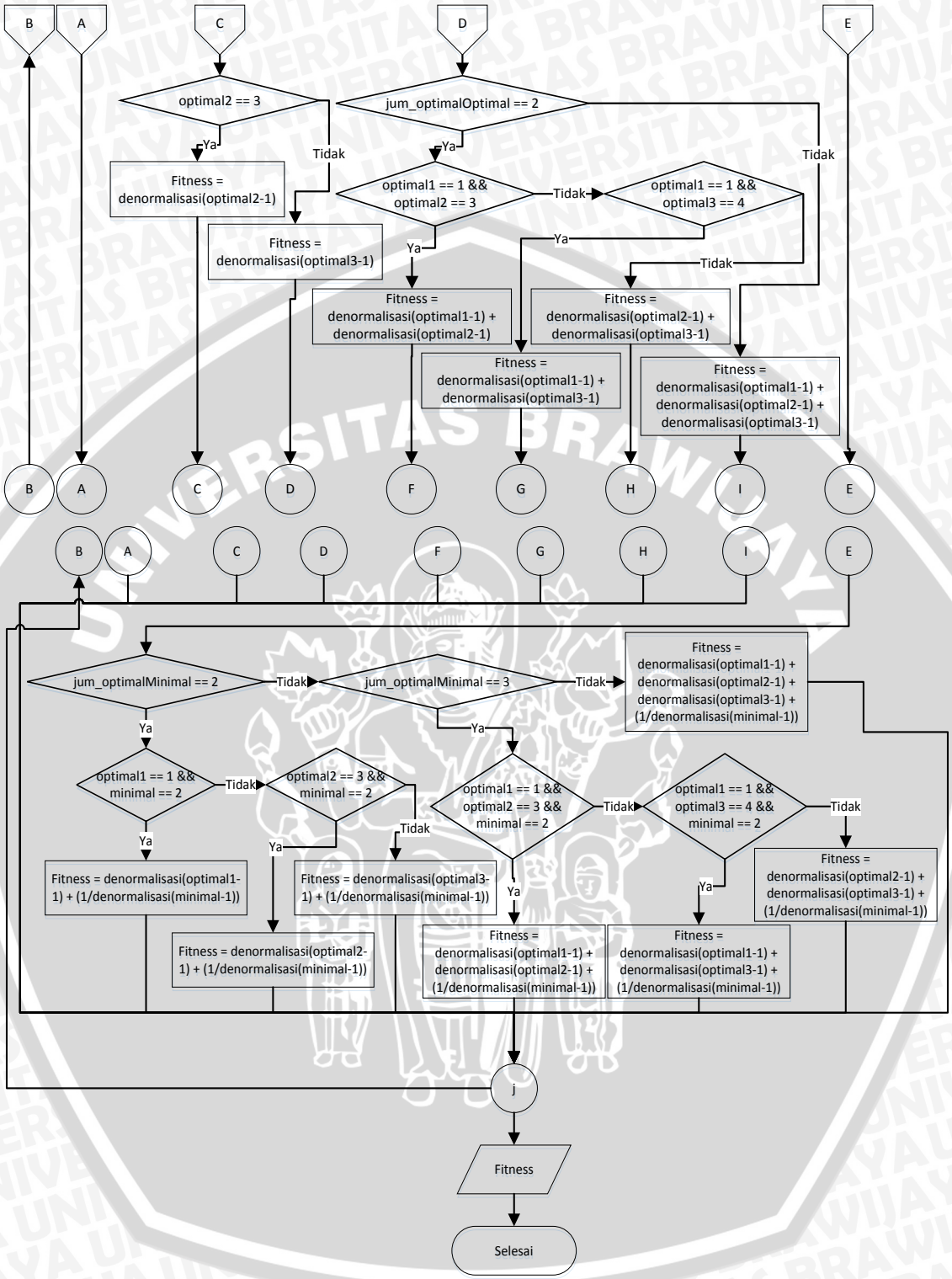
1. Masukan *popsize*, *mutation rate* (*mr*) yang dimasukkan oleh pengguna, batasAtas, batasBawah, nilai *r*, *maxData* dan *minData* yang ditentukan dalam program.

2. Sistem menghitung hasil mutasi yang didapat, yaitu untuk mengetahui berapa kali proses mutasi yang harus dilakukan dengan menggunakan persamaan yang terdapat pada diagram alir di atas.
3. Sistem melakukan pengulangan sebanyak hasil mutasi yang didapatkan kemudian menghitung *PosisiPointRandom* dengan menggunakan persamaan yang terdapat pada diagram alir di atas.
4. Sistem melakukan pemilihan terhadap satu *parent* secara acak.
5. Sistem melakukan pengulangan sebanyak panjang *stringLen* yang ditetapkan.
6. Sistem melakukan percabangan dan mengecek kondisi jika *PosisiPointRandom == Parent1 (PosisiPointRandom – 1)*, maka sistem menghitung hasil mutasi dengan menggunakan persamaan yang terdapat pada diagram alir di atas.
7. Sedangkan, jika tidak memenuhi kondisi percabangan, maka sistem menghitung hasil mutasi dengan menggunakan Persamaan 2-19.
8. Sistem menghasilkan *output* berupa *offspring* dari hasil mutasi.

4.1.2.4 Perhitungan Nilai Fitness

Pada tahap ini dilakukan perhitungan nilai *fitness.* Fungsi dari nilai *fitness* adalah untuk mengetahui seberapa baik suatu individu tersebut untuk dijadikan sebagai solusi (Mahmudy, 2013). Berikut adalah diagram alir yang menjelaskan tentang proses perhitungan nilai *fitness* yang diperlihatkan pada Gambar 4.13.





Gambar 4.13 Diagram Alir Perhitungan Nilai *Fitness*

Pada Gambar 4.13 Diagram Alir Perhitungan Nilai *Fitness* dilakukan dengan tahapan-tahapan sebagai berikut:

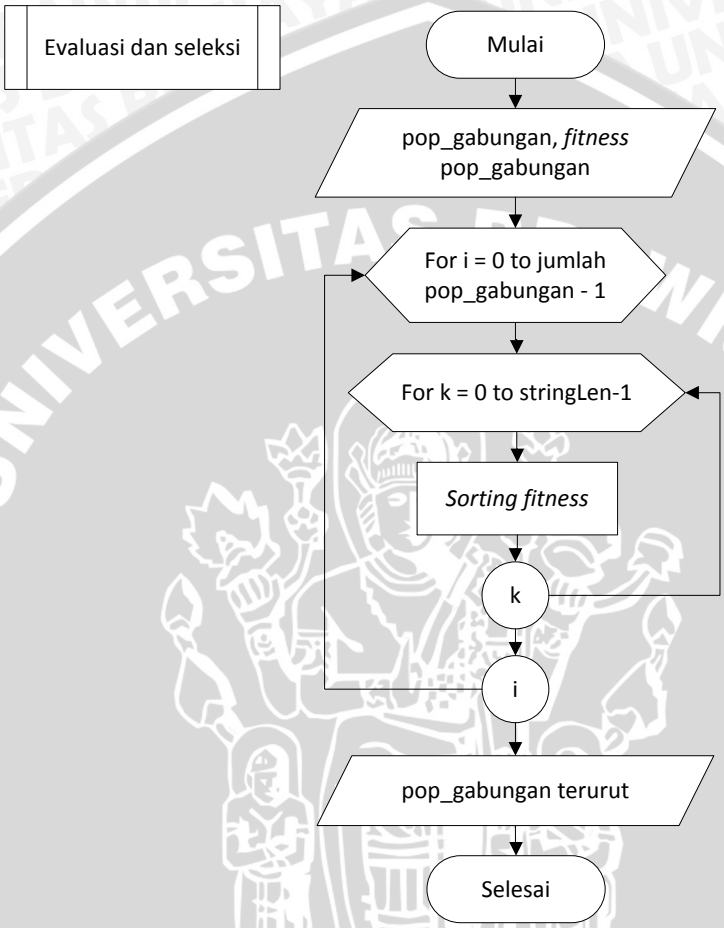
1. Masukan pop_gabungan yang diperoleh dari hasil inisialisasi populasi awal dan proses reproduksi *crossover* dan mutasi, denormalisasi dari proses pengujian ANN *backpropagation* pada sub bab 4.1.1.5 terhadap

pop_gabungan pada GA, model_optimasi merupakan hasil pilihan pemodelan, jum_optimalOptimal merupakan jumlah kandungan gizi susu yang ingin dioptimalkan, jum_optimalMinimal merupakan jumlah kandungan gizi susu yang ingin dioptimalkan dan diminimalkan, optimal1 merupakan kandungan gizi susu protein, optimal2 merupakan kandungan gizi susu laktosa, optimal3 merupakan kandungan gizi susu *density*, minimal merupakan kandungan gizi susu lemak.

2. Sistem melakukan pengulangan sebanyak jumlah pop_gabungan.
3. Sistem melakukan percabangan. Jika model_optimasi == 1, maka sistem akan melakukan percabangan terhadap jum_optimalOptimal == 1. Sedangkan, jika jum_optimalOptimal != 1, maka sistem melakukan langkah ke-5. Jika model_optimasi != 1, maka sistem melakukan langkah ke-7.
4. Sistem melakukan percabangan. Jika optimal1 == 1, maka sistem menghitung nilai *fitness* sesuai rumus yang ada pada diagram. Sedangkan, jika optimal2 == 3, maka sistem menghitung nilai *fitness* sesuai rumus yang ada pada diagram. Sedangkan, jika optimal2 != 3, maka sistem menghitung nilai *fitness* sesuai rumus yang ada pada diagram.
5. Jika jum_optimalOptimal == 2, maka sistem melakukan percabangan berikutnya pada langkah ke-6. Sedangkan, jika jum_optimalOptimal != 2, maka sistem menghitung nilai *fitness* sesuai rumus yang ada pada diagram.
6. Jika optimal1 == 1 && optimal2 == 3, maka sistem menghitung nilai *fitness* sesuai rumus yang ada pada diagram. Jika optimal1 == 1 && optimal3 == 4, maka sistem menghitung nilai *fitness* sesuai rumus yang ada pada diagram. Jika tidak memenuhi kondisi yang ada, maka sistem menghitung nilai *fitness* sesuai rumus yang ada pada diagram.
7. Jika jum_optimalMinimal == 2, maka sistem melakukan langkah ke-8. Jika jum_optimalMinimal == 3, maka sistem melakukan langkah ke-9. Jika jum_optimalMinimal != 3, maka sistem menghitung nilai *fitness* sesuai rumus yang ada pada diagram.
8. Jika optimal1 == 1 && minimal == 2, maka sistem menghitung nilai *fitness* sesuai rumus yang ada pada diagram. Jika optimal2 == 3 && minimal == 2, maka sistem menghitung nilai *fitness* sesuai rumus yang ada pada diagram. Jika optimal2 != 3 && minimal == 2, maka sistem menghitung nilai *fitness* sesuai rumus yang ada pada diagram.
9. Jika optimal1 == 1 && optimal2 == 3 && minimal == 2, maka sistem menghitung nilai *fitness* sesuai rumus yang ada pada diagram. Jika optimal1 == 1 && optimal3 == 4 && minimal == 2, maka sistem menghitung nilai *fitness* sesuai rumus yang ada pada diagram. Jika optimal1 != 1 && optimal3 != 4 && minimal == 2, maka sistem menghitung nilai *fitness* sesuai rumus yang ada pada diagram.
10. Sistem menghasilkan *fitness* dari pop_gabungan (*parent* dan *offspring*).

4.1.2.5 Proses Evaluasi dan Seleksi

Proses evaluasi adalah proses untuk menggabungkan seluruh populasi dan menghitung nilai *fitness*-nya. Sedangkan, proses seleksi adalah proses memilih individu dari seluruh populasi yang dipertahankan hidup pada generasi selanjutnya. Metode seleksi yang digunakan adalah *elitism selection* (Mahmudy, 2013). Berikut adalah Gambar 4.14 Diagram Alir Proses Evaluasi dan Seleksi.



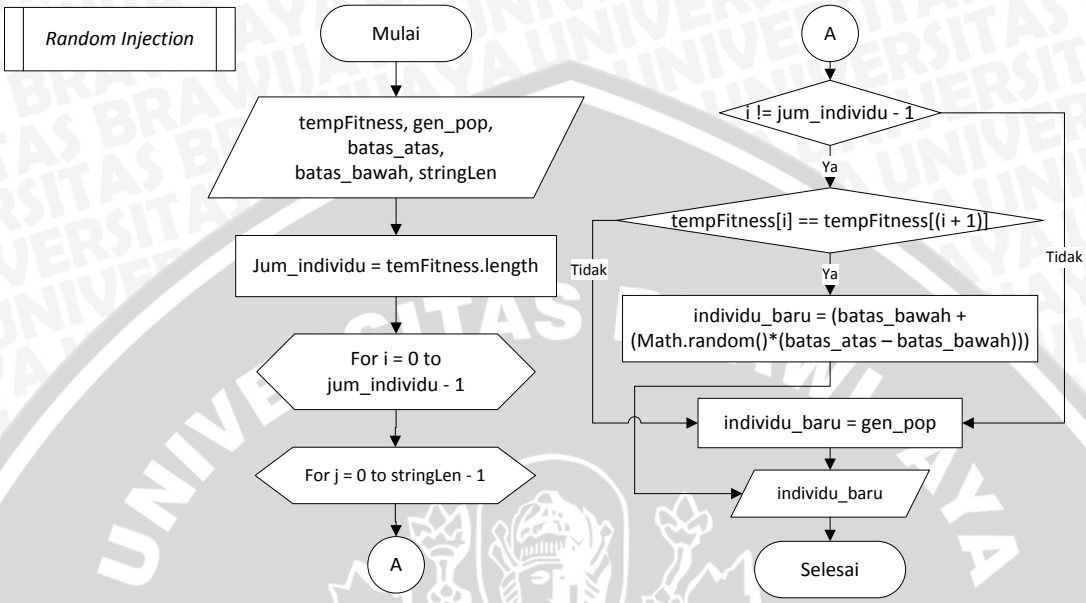
Gambar 4.14 Diagram Alir Proses Evaluasi dan Seleksi

Pada Gambar 4.14 Diagram Alir Proses Evaluasi dan Seleksi dilakukan dengan tahapan-tahapan sebagai berikut:

1. Masukan pop_gabungan, yaitu himpunan dari populasi awal dan *offspring* yang didapat dari proses *crossover* dan mutasi beserta nilai *fitness*-nya.
2. Sistem melakukan pengulangan sebanyak jumlah pop_gabungan dan pengulangan sebanyak panjang stringLen yang telah ditetapkan.
3. Sistem melakukan proses *sorting* nilai *fitness* dari pop_gabungan. *Sorting* ini dilakukan dengan cara mengurutkan individu yang memiliki nilai *fitness* tertinggi hingga terendah menggunakan metode *elitism selection*.
4. Sistem menghasilkan pop_gabungan yang telah terurut, yaitu menampilkan individu yang memiliki nilai *fitness* tertinggi hingga terendah.

4.1.2.6 Proses Random Injection

Jika pada interval iterasi kelipatan 1, maka sistem akan melakukan proses *random injection* sesuai dengan teori yang telah dijelaskan pada Bab 2. Berikut adalah Gambar 4.15 Diagram Alir Proses *Random Injection*.



Gambar 4.15 Diagram Alir Proses *Random Injection*

4.2 Formulasi Permasalahan

Formulasi permasalah adalah tahap perancangan yang akan menjelaskan tentang permasalahan yang akan diselesaikan, yaitu memodelkan komposisi pakan kambing PE untuk dapat mengoptimalkan kandungan gizi susu kambing PE. Pada contoh permasalahan ini, pengguna memilih 2 jenis pakan hijauan, yaitu rumput odot dan rumput raja. Kandungan nutrisi bahan pakan rumput odot dalam 1 kg adalah abu sebesar 16.97%, PK sebesar 17.03%, LK sebesar 1.76% dan SK sebesar 24.84%. Sedangkan, kandungan nutrisi bahan pakan rumput raja dalam 1 kg adalah abu sebesar 18.6%, PK sebesar 13.5%, LK sebesar 3.5% dan SK sebesar 24.84%.

Selain itu, pengguna menggunakan presentase pemberian pakan untuk rumput odot dan rumput raja sebesar 70%:30%, bobot badan kambing yang digunakan adalah 32 kg, kualitas susu yang digunakan adalah Premium dan kandungan gizi susu yang ingin dioptimalkan adalah protein dan yang diminimalkan adalah lemak. Banyaknya pemberian pakan hijauan pada kambing PE sangat berpengaruh terhadap bobot badan yang dimiliki setiap kambing PE. Minimal dan maksimal pemberian pakan hijauan pada satu ekor kambing PE adalah 10% dari bobot badan kambing PE dan 12% dari bobot badan kambing PE.

Data latih dan data uji tentang kandungan gizi susu yang digunakan adalah 9 data latih dan 6 data uji dalam format .csv. Fungsi aktivasi yang digunakan pada metode ANN adalah fungsi sigmoid biner. Parameter perhitungan pada ANN dan GA yang digunakan sebagai berikut:

a. Iterasi maksimal = 2
b. Jumlah *hidden layer* = 4
c. *Learning rate* (α) = 0.64
d. Galat maksimal = 0.001
e. *Popsize* = 5
f. Jumlah generasi = 1
g. *Crossover rate* (*cr*) = 0.4
h. *Mutation rate* (*mr*) = 0.2

Setelah menentukan parameter ANN dan GA, langkah pertama adalah masuk pada proses algoritma ANN yang terdiri dari 2 tahap, yaitu proses perhitungan *feedforward* dan proses perhitungan *backpropagation*. Langkah kedua adalah masuk pada proses algoritma GA dimana sistem akan melakukan proses perhitungan dengan metode *extended intermediate crossover*, *random mutation* dan *elitism selection*. Hasil akhir yang diperoleh pada proses algoritma GA adalah hasil pemodelan komposisi pakan dengan kandungan gizi susu yang paling optimal dan/atau minimal.

4.3 Perhitungan Manualisasi *Artificial Neural Network* (ANN) dan *Genetic Algorithm* (GA)

Alir perancangan sistem ANN dan GA telah diuraikan secara rinci pada sub bab 4.1. Alir perancangan sistem tersebut kemudian disederhanakan ke dalam perhitungan manual ANN dan GA agar cara kerja dari metode ANN dan GA dapat lebih mudah untuk dipahami. Parameter ANN dan GA yang akan digunakan pada perhitungan manual ini telah dijelaskan pada sub bab 4.2.

4.3.1 Perhitungan Manual *Artificial Neural Network* (ANN)

Pada perhitungan manual ANN ini akan dilakukan beberapa proses perhitungan sesuai dengan diagram alir perancangan sistem yang telah dibuat pada sub bab 4.1.2. Proses perhitungan manual ANN yang akan dilakukan meliputi proses normalisasi data latih dan data uji, perhitungan pelatihan data latih ANN, perhitungan pengujian data uji ANN dan proses denormalisasi.

4.3.1.1 Proses Normalisasi Data Latih dan Data Uji

Pada perhitungan ini akan dilakukan normalisasi untuk data latih dan data uji. Sampel data latih kandungan nutrisi pakan hijauan rumput odot dan rumput raja beserta kandungan gizi susu pada penelitian ini berjumlah 9 data. Berikut adalah tabel yang menjelaskan tentang data latih kandungan nutrisi bahan pakan hijauan rumput odot dan rumput raja beserta kandungan gizi susunya sebelum normalisasi pada Tabel 4.1.

Tabel 4.1 Data Latih Sebelum Normalisasi

No	Nama Perlakuan	Kandungan Nutrisi				*Output*			
		Abu	PK	LK	SK	Protein	Lemak	Laktosa	Density
1	Premium	1.2940	1.0570	0.1810	1.7340	4.11	7.18	3.93	28.49
2	Premium	1.3170	1.0010	0.1920	1.7270	4.08	7.07	3.89	28.95

3	Premium	1.3010	1.0350	0.1980	1.7150	3.83	8.19	3.67	25.33
4	Premium	1.2890	1.1630	0.1800	1.9070	3.97	6.52	3.79	27.8
5	Premium	1.1840	1.1220	0.1800	1.8160	4.13	7.48	3.94	28.36
6	Premium	1.3090	1.1000	0.1850	1.6710	3.89	6.25	3.14	22.3
7	Premium	1.3340	0.9890	0.1920	1.7710	3.84	7.5	3.67	25.93
8	Premium	1.1920	1.0430	0.1860	1.6540	3.81	7.45	3.64	25.71
9	Premium	1.2360	1.0020	0.1710	1.6530	3.83	6.72	3.65	26.45

Adapun sampel data uji kandungan nutrisi bahan pakan hijauan rumput odot dan rumput raja yang digunakan pada penelitian ini berjumlah 6 data. Berikut adalah tabel yang menjelaskan tentang data uji kandungan nutrisi bahan pakan hijauan rumput odot dan rumput raja sebelum normalisasi pada Tabel 4.2.

Tabel 4.2 Data Uji Sebelum Normalisasi

No	Nama Perlakuan	Kandungan Nutrisi			
		Abu	PK	LK	SK
1	Premium	1.1750	1.1480	0.1840	1.9050
2	Premium	1.1980	1.0610	0.1830	1.8430
3	Premium	1.3580	1.0520	0.1710	1.8270
4	Premium	1.3360	1.0440	0.2000	1.6890
5	Premium	1.3000	0.9930	0.1750	1.8870
6	Premium	1.2740	1.0980	0.1820	1.7200

Proses normalisasi data pada Tabel 4.1 dan Tabel 4.2 dilakukan pada abu, PK, LK, SK, protein, lemak, laktosa dan *density*. Berikut adalah hasil normalisasi untuk data latih pada Tabel 4.3 Data Latih Sesudah Normalisasi.

Tabel 4.3 Data Latih Sesudah Normalisasi

No	Nama Perlakuan	Kandungan Nutrisi				*Output*			
		Abu	PK	LK	SK	Protein	Lemak	Laktosa	Density
1	Premium	0.6867	0.4126	0.3963	0.3551	0.8500	0.4835	0.8900	0.8447
2	Premium	0.8093	0.1552	0.7222	0.3331	0.7750	0.4381	0.8500	0.9000
3	Premium	0.724	0.3115	0.9	0.2953	0.1500	0.9000	0.6300	0.4645
4	Premium	0.66	0.9	0.3667	0.9	0.5000	0.2113	0.7500	0.7617
5	Premium	0.1	0.7115	0.3667	0.6134	0.9	0.6	0.9	0.8
6	Premium	0.7667	0.6103	0.5148	0.1567	0.3000	0.1	0.1	0.1
7	Premium	0.9	0.1	0.7222	0.4717	0.1750	0.6155	0.6300	0.5367
8	Premium	0.1427	0.3483	0.5444	0.1031	0.1	0.5948	0.6000	0.5102
9	Premium	0.3773	0.1598	0.1	0.1	0.1500	0.2938	0.6100	0.5992

Pada Tabel 4.1 diketahui *min value* dan *max value* untuk kandungan nutrisi abu dari sembilan data adalah 1.1840 dan 1.3340, sehingga hasil normalisasi untuk abu adalah 0.6867 dengan data asli kandungan nutrisi abu adalah 1.2940. Contoh perhitungan normalisasinya sebagai berikut:

Normalisasi kandungan abu = (0.8*(1.2940-1.1840)/(1.3340-1.1840)) + 0.1

= 0.6867

Berikut adalah hasil proses normalisasi data untuk data uji yang diperlihatkan pada Tabel 4.4 Data Uji Sesudah Normalisasi dengan melakukan perhitungan seperti yang dicontohkan sebelumnya.

Tabel 4.4 Data Uji Sesudah Normalisasi

No	Nama Perlakuan	Kandungan Nutrisi			
		Abu	PK	LK	SK
1	Premium	0.1	0.9	0.4586	0.9
2	Premium	0.2005	0.4510	0.4310	0.6704
3	Premium	0.9	0.4045	0.1	0.6111
4	Premium	0.8038	0.3632	0.9	0.1
5	Premium	0.6464	0.1	0.2103	0.8333
6	Premium	0.5328	0.6419	0.4034	0.2148

4.3.1.2 Perhitungan Pelatihan Data Latih ANN

Proses pelatihan ANN digunakan untuk melatih data latih yang tersedia ke dalam jaringan untuk dapat melakukan perubahan bobot hingga diperoleh nilai MSE yang terendah. Pada kasus permasalahan ini, proses pelatihan ANN dilakukan pada 9 data latih dengan iterasi sebanyak dua kali. Berikut adalah proses manualisasi untuk iterasi pertama terhadap data latih:

1. Iterasi Pertama

Pada iterasi pertama, iterasi dimulai dari t=1. Iterasi pertama pada ANN merupakan langkah awal dari pembangkitan kecepatan (V_{ij}) antara bobot dari unit *input* (i) menuju unit *hidden* (j) secara *random*.

1.1 Perhitungan *Feedforward*

a. Proses *Input Layer* ke *Hidden Layer*

Proses *input layer* ke *hidden layer* (V_{ij}) dibangkitkan secara acak untuk data pertama dari (-1,1). Jumlah *hidden layer* yang digunakan pada manualisasi adalah 4 *hidden layer*. Berikut adalah data ke-1 dari perlakuan Premium sesudah normalisasi pada iterasi pertama yang diperlihatkan pada Tabel 4.5 dan bobot *input layer* ke *hidden layer* diperlihatkan pada Tabel 4.6 sebagai berikut:

Tabel 4.5 Data ke-1 Perlakuan Premium Sesudah Normalisasi

No	Nama Perlakuan	Kandungan Nutrisi			
		Abu	PK	LK	SK
1	Premium	0.6867	0.4126	0.3963	0.3551

Tabel 4.6 Bobot *Input Layer* ke *Hidden Layer*

V11	V21	V31	V41	V12	V22	V32	V42
0.7105	-0.5918	-0.6764	-0.1384	0.2525	-0.0935	0.1381	-0.7931
V13	**V23**	**V33**	**V43**	**V14**	**V24**	**V34**	**V44**
-0.0664	-0.3608	0.7026	-0.6888	0.4017	-0.5565	-0.0508	-0.3314

Bobot untuk data ke-2 sampai ke-9 didapatkan pada proses perubahan bobot data sebelumnya yang diperlihatkan secara lengkap pada Lampiran D. Hanya data ke-1 pada iterasi pertama saja yang bobotnya dibangkitkan secara acak.

b. Menjumlahkan Bobot Sinyal *Input*

Langkah selanjutnya adalah menjumlahkan bobot sinyal *input*. Contoh manualisasi perhitungan bobot sinyal *input* dari data ke-1 perlakuan Premium sebagai berikut:

$$Z_in_1 = (0.6867 * 0.7105) + (0.4126 * -0.5918) + (0.3963 * -0.6764) + (0.3551 * -0.1384)$$
$$= -0.0735$$

$$Z_in_2 = (0.6867 * 0.2525) + (0.4126 * -0.0935) + (0.3963 * 0.1381) + (0.3551 * -0.7931)$$
$$= -0.0921$$

$$Z_in_3 = (0.6867 * -0.0664) + (0.4126 * -0.3608) + (0.3963 * 0.7026) + (0.3551 * -0.6888)$$
$$= -0.1606$$

$$Z_in_4 = (0.6867 * 0.4017) + (0.4126 * -0.5565) + (0.3963 * -0.0508) + (0.3551 * -0.3314)$$
$$= -0.0916$$

Berikut adalah hasil perhitungan penjumlahan bobot sinyal *input* yang didapatkan dari 9 data latih yang telah dinormalisasi pada Tabel 4.7 Jumlah Z_in_j Bobot Sinyal *Input* sebagai berikut:

Tabel 4.7 Jumlah Z_in_j Bobot Sinyal *Input*

Z_in1	Z_in1	Z_in3	Z_in4
-0.0735	-0.0921	-0.1606	-0.0916
-0.0512	0.0287	0.1676	0.0921
-0.3230	0.0472	0.2661	-0.0282
-0.4586	-0.5951	-0.7455	-0.5658
-0.7156	-0.4980	-0.4472	-0.5982
-0.2210	0.0650	-0.0379	-0.1299
-0.0611	-0.1254	0.0222	0.0443
-0.5411	-0.0460	0.1381	-0.2386
0.0494	-0.0197	-0.1112	-0.0074

c. Menerapkan Fungsi Aktivasi Menghitung Sinyal *Input*

Fungsi aktivasi yang digunakan untuk menyelesaikan permasalahan ini adalah fungsi aktivasi sigmoid biner. Berikut adalah contoh manualisasi proses perhitungan sinyal *input*:

e = bilangan *eural*, yaitu 2.71828

$$Z_1 = \frac{1}{(1 + 2.71828^{-(-0.0735)})} = 0.4816$$

$$Z_2 = \frac{1}{(1+2.71828^{-(-0.0921)})} = 0.4470$$

$$Z_3 = \frac{1}{(1+2.71828^{-(-0.1606)})} = 0.4599$$

$$Z_4 = \frac{1}{(1+2.71828^{-(-0.0916)})} = 0.4771$$

Berikut adalah hasil sinyal *input* dengan menerapkan fungsi aktivasi sigmoid biner yang didapatkan dari 9 data latih yang telah dinormalisasi pada Tabel 4.8 Hasil Perhitungan Sinyal *Input* dengan Fungsi Sigmoid Biner sebagai berikut:

Tabel 4.8 Hasil Perhitungan Sinyal *Input* dengan Fungsi Sigmoid biner

Z_1	Z_2	Z_3	Z_4
0.4816	0.4770	0.4599	0.4771
0.4872	0.5072	0.5418	0.5230
0.4199	0.5118	0.5661	0.4929
0.3873	0.3555	0.3218	0.3622
0.3284	0.3780	0.3900	0.3548
0.4450	0.5162	0.4905	0.4676
0.4847	0.4687	0.5055	0.5111
0.3679	0.4885	0.5345	0.4406
0.5123	0.4951	0.4722	0.4982

d. Bobot dari *Hidden Layer* ke *Output Layer*

Untuk data ke-1, bobot dari *hidden layer* ke *output layer* dibangkitkan secara *random* dengan *range* (0,1). Sedangkan, data ke-2 sampai data ke-9 ditentukan berdasarkan perubahan bobot data sebelumnya yang diperlihatkan secara lengkap pada Lampiran D. Berikut adalah bobot dari *hidden layer* ke *output layer* untuk data ke-1 pada Tabel 4.9 Bobot *Hidden Layer* ke *Output Layer* sebagai berikut:

Tabel 4.9 Bobot *Hidden Layer* ke *Output Layer*

W11	W21	W31	W41	W12	W22	W32	W42
0.5716	0.4962	0.1278	0.1989	0.9383	0.6603	0.7164	0.7722
W13	**W23**	**W33**	**W43**	**W14**	**W24**	**W34**	**W44**
0.3699	0.7822	0.1866	0.7479	0.5031	0.3135	0.8484	0.3829

e. Menjumlahkan Setiap Bobot Sinyal *Output*

Langkah selanjutnya adalah menjumlahkan setiap bobot sinyal *output*. Contoh manualisasi perhitungan bobot sinyal *output* dari data ke-1 perlakuan Premium sebagai berikut:

$$Y_in_1 = (0.4816 * 0.5716) + (0.4770 * 0.4962) + (0.4599 * 0.1278) + (0.4771 * 0.1989) = 0.6657$$

$$Y_in_2 = (0.4816 * 0.9383) + (0.4770 * 0.6603) + (0.4599 * 0.7164) + (0.4771 * 0.7722) = 1.4648$$

$$Y_in_3 = (0.4816 * 0.3699) + (0.4770 * 0.7822) + (0.4599 * 0.1866) + (0.4771 * 0.5031) = 0.9939$$

$$Y_in_4 = (0.4816 * 0.5031) + (0.4770 * 0.3135) + (0.4599 * 0.8484) + (0.4771 * 0.3829) = 0.9647$$

Berikut adalah hasil menjumlahkan bobot sinyal *output* yang didapatkan dari 9 data latih yang telah dinormalisasi ketika iterasi pertama yang diperlihatkan pada Tabel 4.10 Jumlah Y_in_k Bobot Sinyal *Hidden Layer* ke *Output Layer* sebagai berikut:

Tabel 4.10 Jumlah Y_in_k Bobot Sinyal *Hidden Layer* ke *Output Layer*

Y_in_1	Y_in_2	Y_in_3	Y_in_4
0.6657	1.4648	0.9939	0.9647
0.7299	1.5528	1.0889	1.0791
0.7045	1.4512	1.0617	1.0745
0.4878	1.0667	0.7652	0.7180
0.4621	1.0352	0.7731	0.7567
0.6523	1.3592	1.0541	1.0037
0.6131	1.3381	0.9938	0.9638
0.4871	1.2038	0.9090	0.8901
0.4971	1.3107	0.9831	0.9057

f. Menerapkan Fungsi Aktivasi Menghitung Sinyal *Output*

Fungsi aktivasi yang digunakan adalah fungsi aktivasi sigmoid biner. Berikut adalah contoh manualisasi proses perhitungan sinyal *output* pada data ke-1:

$$Y_1 = \frac{1}{(1 + 2.71828^{-(0.6657)})} = 0.6605$$

$$Y_2 = \frac{1}{(1 + 2.71828^{-(1.4648)})} = 0.8123$$

$$Y_3 = \frac{1}{(1 + 2.71828^{-(0.9939)})} = 0.7299$$

$$Y_4 = \frac{1}{(1 + 2.71828^{-(0.9647)})} = 0.7241$$

Berikut adalah hasil perhitungan sinyal *output* dengan menerapkan fungsi aktivasi sigmoid biner pada Tabel 4.11 Hasil Perhitungan Sinyal *Output* dengan Fungsi Sigmoid Biner sebagai berikut:

Tabel 4.11 Hasil Perhitungan Sinyal *Output* dengan Fungsi Sigmoid biner

Y_1	Y_2	Y_3	Y_4
0.6605	0.8123	0.7299	0.7241
0.6748	0.8253	0.7482	0.7463
0.6692	0.8102	0.7430	0.7454
0.6196	0.7440	0.6825	0.6722
0.6135	0.7379	0.6842	0.6806
0.6575	0.7956	0.7416	0.7318
0.6486	0.7922	0.7298	0.7239
0.6194	0.7692	0.7128	0.7089
0.6218	0.7876	0.7277	0.7121

1.2 Perhitungan *Backpropagation*

Proses selanjutnya adalah perhitungan *backpropagation*. Tahapan-tahapan pada proses *backpropagation* sebagai berikut:

a. Menghitung *Error Output*

Pada proses ini, unit *output* berusaha untuk mencapai nilai target yang sesuai dengan pola *input* pelatihan. Setelah itu, menghitung *error output* yang dihasilkan. Berikut adalah target dari data yang telah dinormalisasikan dari data ke-1 pada Tabel 4.12 Target Data ke-1 Sesudah Normalisasi sebagai berikut:

Tabel 4.12 Target Data ke-1 Sesudah Normalisasi

No	Nama Perlakuan	*Output*			
		Protein	Lemak	Laktosa	*Density*
1	Premium	0.8500	0.4835	0.8900	0.8447

Contoh manualisasi perhitungan *error output* dari target data ke-1 sesudah normalisasi sebagai berikut:

$$\delta_1 = (0.85 - 0.6605) * 0.6605 * (1 - 0.6605) = 0.0425$$

$$\delta_2 = (0.4835 - 0.8123) * 0.8123 * (1 - 0.8123) = -0.0501$$

$$\delta_3 = (0.89 - 0.7299) * 0.7299 * (1 - 0.7299) = 0.0316$$

$$\delta_4 = (0.8447 - 0.7241) * 0.7241 * (1 - 0.7241) = 0.0241$$

Berikut adalah hasil *error* antara *output* dengan target yang diberikan sebagai data latih pada Tabel 4.13 Hasil Perhitungan *Error Output* ke Target:

Tabel 4.13 Hasil Perhitungan *Error Output* ke Target

δ_1	δ_2	δ_3	δ_4
0.0425	-0.0501	0.0316	0.0241
0.0220	-0.0558	0.0192	0.0291
-0.1149	0.0138	-0.0216	-0.0533
-0.0282	-0.1015	0.0146	0.0197

0.0679	-0.0253	0.0466	0.0323
-0.0805	-0.1131	-0.1230	-0.1240
-0.1079	-0.0291	-0.0197	-0.0374
-0.1224	-0.0310	-0.0231	-0.0410
-0.1109	-0.0826	-0.0233	-0.0231

b. Menghitung Perubahan Bobot pada *Output Layer*

Langkah selanjutnya adalah proses menghitung perubahan bobot. Proses ini menggunakan nilai *learning rate* = 0.64. Proses menghitung perubahan bobot merupakan proses yang sekaligus mengirimkan informasi *error* yang didapatkan ke lapisan *output*. Berikut adalah contoh manualisasi perhitungan perubahan bobot terhadap *output layer* sebagai berikut:

$$\Delta W_{11} = 0.64 * 0.0425 * 0.4816 = 0.0131$$

$$\Delta W_{12} = 0.64 * -0.0501 * 0.4816 = -0.0155$$

$$\Delta W_{13} = 0.64 * 0.0316 * 0.4816 = 0.0097$$

$$\Delta W_{14} = 0.64 * 0.0241 * 0.4816 = 0.0074$$

Berikut adalah hasil perhitungan perubahan bobot yang didapatkan pada data latih ke-1 di Tabel 4.13 Hasil Perhitungan Perubahan Bobot *Output Layer*:

Tabel 4.14 Hasil Perhitungan Perubahan Bobot *Output Layer*

ΔW11	**ΔW21**	**ΔW31**	**ΔW41**	**ΔW12**	**ΔW22**	**ΔW32**	**ΔW42**
0.0131	0.0130	0.0125	0.0130	-0.0155	-0.0153	-0.0148	-0.0153
ΔW13	**ΔW23**	**ΔW33**	**ΔW43**	**ΔW14**	**ΔW24**	**ΔW34**	**ΔW44**
0.0097	0.0096	0.0093	0.0096	0.0074	0.0074	0.0071	0.0074

Sedangkan, hasil perhitungan perubahan bobot *output layer* untuk semua data pada iterasi pertama diperlihatkan secara lengkap pada Lampiran D.

c. Menjumlahkan Delta *Input*

Setelah itu, setiap unit *hidden layer* melakukan proses penjumlahan setiap delta *input*-nya pada unit-unit yang berada pada *output layer*. Berikut adalah contoh manualisasi perhitungan menjumlahkan delta *input* sebagai berikut:

$$\delta_in_1 = (0.0425 * 0.5716) + (-0.0501 * 0.9383) + (0.0316 * 0.3699) + (0.0241 * 0.5031) = 0.001$$

$$\delta_in_2 = (0.0425 * 0.4962) + (-0.0501 * 0.6603) + (0.0316 * 0.7822) + (0.0241 * 0.3135) = 0.0202$$

$$\delta_in_3 = (0.0425 * 0.1278) + (-0.0501 * 0.7164) + (0.0316 * 0.1866) + (0.0241 * 0.8484) = -0.0042$$

$$\delta_in_4 = (0.0425 * 0.1989) + (-0.0501 * 0.7722) + (0.0316 * 0.7479) + (0.0241 * 0.3829) = 0.0026$$

Berikut adalah hasil menjumlahkan setiap unit *hidden* ke *output* terhadap semua data latih yang telah dinormalisasi pada Tabel 4.15 Hasil Menjumlahkan Delta Unit *Hidden* ke *Output* sebagai berikut:

Tabel 4.15 Hasil Menjumlahkan Delta Unit *Hidden* ke *Output*

δ_in1	δ_in2	δ_in3	δ_in4
0.0010	0.0202	-0.0042	0.0026
-0.0165	-0.0003	-0.0074	-0.0117
-0.0915	-0.0855	-0.0581	-0.0528
-0.0925	-0.0598	-0.0532	-0.0619
0.0496	0.0640	0.0266	0.0418
-0.2576	-0.2468	-0.2163	-0.2399
-0.1085	-0.0920	-0.0621	-0.0657
-0.1160	-0.0985	-0.0629	-0.0679
-0.1409	-0.1121	-0.0736	-0.0900

d. Menghitung Informasi *Error*

Berikut adalah contoh manualisasi perhitungan informasi *error* yang didapatkan dari *hidden layer* dengan menggunakan data latih ke-1 pada iterasi pertama sebagai berikut:

$$\delta_1 = 0.001 * 0.4816 * (1 - 0.4816) = 0.0003$$

$$\delta_2 = 0.0202 * 0.477 * (1 - 0.477) = 0.005$$

$$\delta_3 = -0.0042 * 0.4599 * (1 - 0.4599) = -0.001$$

$$\delta_4 = 0.0026 * 0.4771 * (1 - 0.4771) = 0.0006$$

Berikut adalah hasil perhitungan informasi *error* pada *hidden layer* untuk semua data latih pada iterasi pertama pada Tabel 4.16 Hasil Menghitung Informasi *Error* sebagai berikut:

Tabel 4.16 Hasil Menghitung Informasi *Error*

δ1	δ2	δ3	δ4
0.0003	0.0050	-0.0010	0.0006
-0.0041	-0.0001	-0.0018	-0.0029
-0.0223	-0.0214	-0.0143	-0.0132
-0.0220	-0.0137	-0.0116	-0.0143
0.0109	0.0150	0.0063	0.0096
-0.0636	-0.0616	-0.0541	-0.0597
-0.0271	-0.0229	-0.0155	-0.0164
-0.0270	-0.0246	-0.0157	-0.0167
-0.0352	-0.0280	-0.0184	-0.0225

e. Perubahan Bobot pada *Hidden Layer*

Tahapan terakhir pada proses *backpropagation* adalah proses menghitung koreksi perubahan bobot pada *hidden layer*. Berikut adalah contoh manualisasi perhitungan perubahan bobot pada *hidden layer* sebagai berikut:

$$\Delta V_{11} = 0.64 * 0.0003 * 0.6867 = 0.0001$$

$$\Delta V_{21} = 0.64 * 0.0003 * 0.4126 = 0.0001$$

$$\Delta V_{31} = 0.64 * 0.0003 * 0.3963 = 0.0001$$

$$\Delta V_{41} = 0.64 * 0.0003 * 0.3551 = 0.0001$$

Berikut adalah perubahan bobot pada *hidden layer* yang didapatkan pada data latih ke-1 iterasi pertama pada Tabel 4.17 Hasil Perubahan Bobot *Hidden Layer* sebagai berikut:

Tabel 4.17 Hasil Perubahan Bobot *Hidden Layer*

ΔV11	**ΔV21**	**ΔV31**	**ΔV41**	**ΔV12**	**ΔV22**	**ΔV32**	**ΔV42**
0.0001	0.0001	0.0001	0.0001	0.0022	0.0013	0.0013	0.0011
ΔV13	**ΔV23**	**ΔV33**	**ΔV43**	**ΔV14**	**ΔV24**	**ΔV34**	**ΔV44**
-0.0005	-0.0003	-0.0003	-0.0002	0.0003	0.0002	0.0002	0.0001

Sedangkan, hasil perhitungan perubahan bobot *hidden layer* untuk semua data pada iterasi pertama diperlihatkan secara lengkap pada Lampiran D.

f. Hitung MSE

Berikut adalah contoh manualisasi perhitungan mencari nilai *error* dan nilai MSE. Nilai MSE yang didapatkan untuk data ke-1 dan rata-rata *error* yang dihasilkan oleh data latih pada iterasi pertama diperlihatkan pada Tabel 4.18 Hasil Hitung *Error* pada Iterasi Pertama sebagai berikut:

$$error = \frac{1}{4} * ((0.85 - 0.6605) + (0.4835 - 0.8123) + (0.89 - 0.7299) + (0.8447 - 0.7241)) = 0.0354$$

Tabel 4.18 Hasil Hitung *Error* pada Iterasi Pertama

Error
0.0354
-0.0079
-0.2058
-0.1238
0.1300
-0.5816
-0.2343
-0.2513
-0.2991

Nilai MSE yang didapatkan untuk 9 data latih pada iterasi pertama adalah:

$$MSE = \frac{1}{9} * \sum (error)^2 = 0.06908$$

1.3 Perhitungan Perbaikan Bobot

Pada perhitungan tahap ini akan dicari bobot terbaru pada setiap unit *hidden* maupun unit *output*. Berikut adalah langkah-langkah proses manualisasinya:

1. Perubahan Bobot pada Unit *Hidden*

Contoh proses manualisasi perhitungan bobot pada unit *hidden* untuk data latih ke-2 sebagai berikut:

$$V_{11} = 0.7105 + 0.0001 = 0.7106$$

$$V_{21} = -0.5918 + 0.0001 = -0.5917$$

$$V_{31} = -0.6764 + 0.0001 = -0.6763$$

$$V_{41} = -0.1384 + 0.0001 = -0.1383$$

Berikut adalah hasil dari perubahan bobot pada unit *hidden* untuk data latih ke-2 iterasi pertama pada Tabel 4.19 Bobot Baru pada Unit *Hidden*:

Tabel 4.19 Bobot Baru pada Unit *Hidden*

V11	V21	V31	V41	V12	V22	V32	V42
0.7106	-0.5917	-0.6763	-0.1383	0.2547	-0.0922	0.1394	-0.7920
V13	**V23**	**V33**	**V43**	**V14**	**V24**	**V34**	**V44**
-0.0669	-0.3611	0.7023	-0.6890	0.4020	-0.5563	-0.0506	-0.3313

2. Perubahan Bobot pada Unit *Output*

Contoh proses manualisasi perhitungan bobot pada unit *output* untuk data latih ke-2 sebagai berikut:

$$W_{11} = 0.5716 + 0.0131 = 0.5847$$

$$W_{21} = 0.4962 + 0.0130 = 0.5092$$

$$W_{31} = 0.1278 + 0.0125 = 0.1403$$

$$W_{41} = 0.1989 + 0.0130 = 0.2119$$

Berikut adalah hasil dari perubahan bobot pada unit *output* untuk data latih ke-2 iterasi pertama pada Tabel 4.20 sebagai berikut:

Tabel 4.20 Bobot Baru pada Unit *Output*

W11	W21	W31	W41	W12	W22	W32	W42
0.5847	0.5092	0.1403	0.2119	0.9228	0.6450	0.7016	0.7569
W13	**W23**	**W33**	**W43**	**W14**	**W24**	**W34**	**W44**
0.3796	0.7918	0.1959	0.7575	0.5105	0.3209	0.8555	0.3903

2. Iterasi Kedua

Untuk proses iterasi kedua, bobot yang digunakan oleh *hidden layer* dan *output layer* pada data ke-1 adalah bobot data terakhir yang didapatkan pada proses pelatihan ANN iterasi pertama. Iterasi dilanjutkan kembali dimulai dari data ke-1 data latih pada Tabel 4.3 Data Latih Sesudah Normalisasi. Proses pada iterasi kedua sama dengan proses pada iterasi pertama sebagai berikut:

1.1 Perhitungan *Feedforward*

a. Proses *Input Layer* ke *Hidden Layer*

Bobot yang digunakan untuk proses *input layer* ke *hidden layer* adalah bobot (V_{ij}) yang didapatkan dari data terakhir pada proses pelatihan ANN iterasi pertama. Berikut adalah bobot baru untuk data latih ke-1 pada iterasi kedua yang diperlihatkan pada Tabel 4.21 Bobot *Input Layer* ke *Hidden Layer* Iterasi Kedua sebagai berikut:

Tabel 4.21 Bobot *Input Layer* ke *Hidden Layer* Iterasi Kedua

V11	V21	V31	V41	V12	V22	V32	V42
0.6403	-0.6368	-0.7366	-0.1681	0.1943	-0.1285	0.0879	-0.8127
V13	**V23**	**V33**	**V43**	**V14**	**V24**	**V34**	**V44**
-0.1158	-0.3935	0.6616	-0.7075	0.3486	-0.5912	-0.0938	-0.3509

Bobot untuk data ke-2 sampai ke-9 didapatkan pada proses perubahan bobot pada tahap sebelumnya yang diperlihatkan secara lengkap pada Lampiran D. Hanya data ke-1 pada iterasi kedua saja yang bobotnya didapatkan dari bobot terakhir pada iterasi sebelumnya.

b. Menjumlahkan Bobot Sinyal *Input*

Langkah selanjutnya adalah menjumlahkan bobot sinyal *input*. Contoh manualisasi perhitungan bobot sinyal *input* dari data latih ke-1 iterasi kedua sebagai berikut:

$$Z_in_1 = (0.6867 * 0.6403) + (0.4126 * -0.6368) + (0.3963 * -0.7366) + (0.3551 * -0.1681)$$
$$= -0.1747$$

$$Z_in_2 = (0.6867 * 0.1943) + (0.4126 * -0.1285) + (0.3963 * 0.0879) + (0.3551 * -0.8127)$$
$$= -0.1734$$

$$Z_in_3 = (0.6867 * -0.1158) + (0.4126 * -0.3935) + (0.3963 * 0.6616) + (0.3551 * -0.7075)$$
$$= -0.2310$$

$$Z_in_4 = (0.6867 * 0.3486) + (0.4126 * -0.5912) + (0.3963 * -0.0938) + (0.3551 * -0.3509)$$
$$= -0.1664$$

Berikut adalah hasil perhitungan penjumlahan bobot sinyal *input* yang didapatkan dari 9 data latih iterasi kedua yang telah dinormalisasi pada Tabel 4.22 Jumlah Z_in_j Bobot Sinyal *Input* Iterasi Kedua sebagai berikut:

Tabel 4.22 Jumlah Z_in_j Bobot Sinyal *Input* Iterasi Kedua

Z_in_1	Z_in_2	Z_in_3	Z_in_4
-0.1747	-0.1734	-0.2310	-0.1664
-0.1665	-0.0650	0.0875	0.0072
-0.4461	-0.0525	0.1802	-0.1191
-0.5842	-0.6917	-0.8323	-0.6577
-0.7847	-0.5495	-0.4942	-0.6476
-0.3236	-0.0167	-0.1096	-0.2058
-0.1662	-0.2076	-0.0483	-0.0307
-0.5906	-0.0842	0.1052	-0.2736
0.0157	-0.0454	-0.1334	-0.0315

c. Menerapkan Fungsi Aktivasi Menghitung Sinyal *Input*

Berikut adalah contoh manualisasi proses perhitungan sinyal *input* pada iterasi kedua:

e = bilangan *eural*, yaitu 2.71828

$$Z_1 = \frac{1}{(1+2.71828^{-(-0.1747)})} = 0.4564$$

$$Z_2 = \frac{1}{(1+2.71828^{-(-0.1734)})} = 0.4568$$

$$Z_3 = \frac{1}{(1+2.71828^{-(--0.2310)})} = 0.4425$$

$$Z_4 = \frac{1}{(1+2.71828^{-(-0.1664)})} = 0.4585$$

Berikut adalah hasil sinyal *input* dengan menerapkan fungsi aktivasi sigmoid biner pada Tabel 4.23 Hasil Perhitungan Sinyal *Input* pada Iterasi Kedua sebagai berikut:

Tabel 4.23 Hasil Perhitungan Sinyal *Input* pada Iterasi Kedua

Z_1	Z_2	Z_3	Z_4
0.4564	0.4568	0.4425	0.4585
0.4585	0.4838	0.5219	0.5018
0.3903	0.4869	0.5449	0.4703
0.3580	0.3337	0.3032	0.3413
0.3133	0.3660	0.3789	0.3435
0.4198	0.4958	0.4726	0.4487
0.4585	0.4483	0.4879	0.4923
0.3565	0.4790	0.5263	0.4320
0.5039	0.4887	0.4667	0.4921

d. Bobot dari *Hidden Layer* ke *Output Layer*

Untuk data ke-1, bobot dari *hidden layer* ke *output layer* diperoleh dari bobot data terakhir pada iterasi sebelumnya. Sedangkan, data ke-2 sampai data ke-9 ditentukan berdasarkan perubahan bobot data sebelumnya yang diperlihatkan secara lengkap pada Lampiran D. Berikut adalah bobot dari *hidden layer* ke *output layer* untuk data ke-1 proses iterasi kedua yang diperlihatkan pada Tabel 4.24 Bobot *Hidden Layer* ke *Output Layer* Iterasi Kedua sebagai berikut:

Tabel 4.24 Bobot *Hidden Layer* ke *Output Layer* Iterasi Kedua

W11	W21	W31	W41	W12	W22	W32	W42
0.4827	0.3914	0.0154	0.0979	0.8302	0.5464	0.6046	0.6612
W13	**W23**	**W33**	**W43**	**W14**	**W24**	**W34**	**W44**
0.3467	0.7519	0.1565	0.7214	0.4604	0.2601	0.7933	0.3342

e. Menjumlahkan Setiap Bobot Sinyal *Output*

Langkah selanjutnya adalah menjumlahkan setiap bobot sinyal *output*. Contoh manualisasi perhitungan bobot sinyal *output* dari data ke-1 proses iterasi kedua sebagai berikut:

$$Y_in_1 = (0.4564 * 0.4827) + (0.4568 * 0.3914) + (0.4425 * 0.0154) + (0.4585 * 0.0979)$$
$$= 0.4508$$

$$Y_in_2 = (0.4564 * 0.8302) + (0.4568 * 0.5464) + (0.4425 * 0.6046) + (0.4585 * 0.6612)$$
$$= 1.1992$$

$$Y_in_3 = (0.4816 * 0.3467) + (0.4770 * 0.7519) + (0.4599 * 0.1565) + (0.4771 * 0.7214)$$
$$= 0.9017$$

$$Y_in_4 = (0.4816 * 0.4604) + (0.4770 * 0.2601) + (0.4599 * 0.7933) + (0.4771 * 0.3342)$$
$$= 0.8332$$

Berikut adalah hasil menjumlahkan bobot sinyal *output* yang didapatkan dari 9 data latih proses iterasi kedua pada Tabel 4.25 Jumlah Y_in_k Bobot Sinyal *Hidden Layer* ke *Output Layer* Iterasi Kedua sebagai berikut:

Tabel 4.25 Jumlah Y_in_k Bobot Sinyal *Hidden Layer* ke *Output Layer* Iterasi Kedua

Y_in_1	Y_in_2	Y_in_3	Y_in_4
0.4508	1.1992	0.9017	0.8332
0.5002	1.2633	0.9873	0.9364
0.4860	1.1675	0.9604	0.9349
0.3341	0.8541	0.6854	0.6130
0.3196	0.8498	0.7141	0.6695
0.4615	1.1065	0.9673	0.8819
0.4254	1.0806	0.9111	0.8454

0.3277	0.9905	0.8524	0.7990
0.3362	1.0923	0.9314	0.8173

f. Menerapkan Fungsi Aktivasi Menghitung Sinyal *Output*

Berikut adalah contoh manualisasi proses perhitungan sinyal *output* pada data ke-1 proses iterasi kedua sebagai berikut:

$$Y_1 = \frac{1}{(1+2.71828^{-(0.4508)})} = 0.6108$$

$$Y_2 = \frac{1}{(1+2.71828^{-(1.1992)})} = 0.7684$$

$$Y_3 = \frac{1}{(1+2.71828^{-(0.9017)})} = 0.7113$$

$$Y_4 = \frac{1}{(1+2.71828^{-(0.8332)})} = 0.6970$$

Berikut adalah hasil perhitungan sinyal *output* dengan menerapkan fungsi aktivasi sigmoid biner pada Tabel 4.26 Hasil Perhitungan Sinyal *Output* pada Iterasi Kedua sebagai berikut:

Tabel 4.26 Hasil Perhitungan Sinyal *Output* pada Iterasi Kedua

Y_1	Y_2	Y_3	Y_4
0.6108	0.7684	0.7113	0.6970
0.6225	0.7796	0.7285	0.7184
0.6192	0.7627	0.7232	0.7181
0.5828	0.7014	0.6649	0.6486
0.5792	0.7005	0.6713	0.6614
0.6134	0.7515	0.7246	0.7072
0.6048	0.7466	0.7132	0.6996
0.5812	0.7292	0.7011	0.6898
0.5833	0.7488	0.7174	0.6937

1.2 Perhitungan *Backpropagation*

Proses selanjutnya adalah perhitungan *backpropagation*. Tahapan-tahapan pada proses *backpropagation* sebagai berikut:

a. Menghitung *Error Output*

Proses selanjutnya adalah menghitung *error output* yang dihasilkan. Target dari data yang telah dinormalisasikan dari data ke-1 terletak pada Tabel 4.5. Contoh manualisasi perhitungan *error output* dari target data ke-1 proses iterasi kedua sebagai berikut:

$$\delta_1 = (0.85 - 0.6108) * 0.6108 * (1 - 0.6108) = 0.0569$$

$$\delta_2 = (0.4835 - 0.7684) * 0.7684 * (1 - 0.7684) = -0.0507$$

$$\delta_3 = (0.89 - 0.7113) * 0.7113 * (1 - 0.7113) = 0.0367$$

$$\delta_4 = (0.8447 - 0.6970) * 0.6970 * (1 - 0.6970) = 0.0312$$

Berikut adalah hasil *error* antara *output* dengan target yang diberikan sebagai data latih pada Tabel 4.27 Hasil Perhitungan *Error Output* ke Target Iterasi Kedua sebagai berikut:

Tabel 4.27 Hasil Perhitungan *Error Output* ke Target Iterasi Kedua

δ1	δ2	δ3	δ4
0.0569	-0.0507	0.0367	0.0312
0.0358	-0.0587	0.0240	0.0367
-0.1106	0.0248	-0.0187	-0.0513
-0.0201	-0.1026	0.0189	0.0258
0.0782	-0.0196	0.0505	0.0375
-0.0743	-0.1217	-0.1246	-0.1257
-0.1027	-0.0248	-0.0170	-0.0342
-0.1171	-0.0265	-0.0212	-0.0384
-0.1053	-0.0856	-0.0218	-0.0201

b. Menghitung Perubahan Bobot

Langkah selanjutnya adalah proses menghitung perubahan bobot. Proses ini menggunakan nilai *learning rate* sebesar 0.64. Berikut adalah contoh manualisasi perhitungan perubahan bobot terhadap *output layer* pada iterasi kedua sebagai berikut:

$$\Delta W_{11} = 0.64 * 0.0569 * 0.4564 = 0.0166$$

$$\Delta W_{12} = 0.64 * -0.0507 * 0.4564 = -0.0148$$

$$\Delta W_{13} = 0.64 * 0.0367 * 0.4564 = 0.0107$$

$$\Delta W_{14} = 0.64 * 0.0312 * 0.4564 = 0.0091$$

Berikut adalah hasil perhitungan perubahan bobot yang didapatkan pada data latih ke-1 pada iterasi kedua pada Tabel 4.28 Hasil Perhitungan Perubahan Bobot sebagai berikut:

Tabel 4.28 Hasil Perhitungan Perubahan Bobot

ΔW11	ΔW21	ΔW31	ΔW41	ΔW12	ΔW22	ΔW32	ΔW42
0.0166	0.0166	0.0161	0.0167	-0.0148	-0.0148	-0.0144	-0.0149
ΔW13	**ΔW23**	**ΔW33**	**ΔW43**	**ΔW14**	**ΔW24**	**ΔW34**	**ΔW44**
0.0107	0.0107	0.0104	0.0108	0.0091	0.0091	0.0088	0.0091

Sedangkan, hasil perhitungan perubahan bobot untuk semua data pada iterasi kedua diperlihatkan secara lengkap pada Lampiran D.

c. Menjumlahkan Delta *Input*

Setelah itu, setiap unit *hidden layer* melakukan proses penjumlahan setiap delta *input*-nya pada unit-unit yang berada pada *output layer*. Berikut adalah contoh manualisasi perhitungan menjumlahkan delta *input* pada iterasi kedua sebagai berikut:

$$\delta_in_1 = (0.0569 * 0.4827) + (-0.0507 * 0.8302) + (0.0367 * 0.3467) + (0.0312 * 0.4604) = 0.0124$$

$$\delta_in_2 = (0.0569 * 0.3914) + (-0.0507 * 0.5464) + (0.0367 * 0.7519) + (0.0312 * 0.2601) = 0.0303$$

$$\delta_in_3 = (0.0569 * 0.0514) + (-0.0507 * 0.6046) + (0.0367 * 0.1565) + (0.0312 * 0.7933) = 0.0007$$

$$\delta_in_4 = (0.0569 * 0.0979) + (-0.0507 * 0.6612) + (0.0367 * 0.7214) + (0.0312 * 0.3342) = 0.0089$$

Berikut adalah hasil menjumlahkan setiap unit *hidden* ke *output* pada Tabel 4.29 Hasil Menjumlahkan Delta Unit *Hidden* ke *Output* Iterasi Kedua sebagai berikut:

Tabel 4.29 Hasil Menjumlahkan Delta Unit *Hidden* ke *Output* Iterasi Kedua

δ_in1	δ_in2	δ_in3	δ_in4
0.0124	0.0303	0.0007	0.0089
-0.0041	0.0116	0.0000	-0.0036
-0.0680	-0.0624	-0.0357	-0.0304
-0.0734	-0.0399	-0.0358	-0.0444
0.0582	0.0689	0.0279	0.0451
-0.2383	-0.2221	-0.1939	-0.2197
-0.0882	-0.0706	-0.0415	-0.0459
-0.0954	-0.0768	-0.0417	-0.0479
-0.1222	-0.0906	-0.0530	-0.0711

d. Menghitung Informasi *Error*

Berikut adalah contoh manualisasi perhitungan informasi *error* yang didapatkan dari *hidden layer* dengan menggunakan data latih ke-1 pada iterasi kedua sebagai berikut:

$$\delta_1 = 0.0124 * 0.4564 * (1 - 0.4564) = 0.0031$$

$$\delta_2 = 0.0303 * 0.4568 * (1 - 0.4568) = 0.0075$$

$$\delta_3 = 0.0007 * 0.4425 * (1 - 0.4425) = 0.0002$$

$$\delta_1 = 0.0089 * 0.4585 * (1 - 0.4585) = 0.0022$$

Berikut adalah hasil perhitungan informasi *error* pada *hidden layer* untuk semua data latih pada iterasi kedua pada Tabel 4.30 Hasil Menghitung Informasi *Error* Iterasi Kedua sebagai berikut:

Tabel 4.30 Hasil Menghitung Informasi *Error* Iterasi Kedua

δ_1	δ_2	δ_3	δ_4
0.0031	0.0075	0.0002	0.0022
-0.0010	0.0029	0.0000	-0.0009
-0.0162	-0.0156	-0.0089	-0.0076
-0.0169	-0.0089	-0.0076	-0.0100
0.0125	0.0160	0.0066	0.0102
-0.0580	-0.0555	-0.0483	-0.0544
-0.0219	-0.0175	-0.0104	-0.0115
-0.0219	-0.0192	-0.0104	-0.0117
-0.0305	-0.0226	-0.0132	-0.0178

e. Perubahan Bobot pada *Hidden Layer*

Berikut adalah contoh manualisasi perhitungan perubahan bobot pada *hidden layer* pada iterasi kedua sebagai berikut:

$$\Delta V_{11} = 0.64 * 0.0031 * 0.6867 = 0.0014$$

$$\Delta V_{21} = 0.64 * 0.0031 * 0.4126 = 0.0008$$

$$\Delta V_{31} = 0.64 * 0.0031 * 0.3963 = 0.0008$$

$$\Delta V_{41} = 0.64 * 0.0031 * 0.3551 = 0.0007$$

Berikut adalah perubahan bobot pada *hidden layer* yang didapatkan pada data latih ke-1 iterasi kedua pada Tabel 4.31 Hasil Perubahan Bobot *Hidden Layer* Iterasi Kedua sebagai berikut:

Tabel 4.31 Hasil Perubahan Bobot *Hidden Layer* Iterasi Kedua

ΔV11	**ΔV21**	**ΔV31**	**ΔV41**	**ΔV12**	**ΔV22**	**ΔV32**	**ΔV42**
0.0014	0.0008	0.0008	0.0007	0.0033	0.0020	0.0019	0.0017
ΔV13	**ΔV23**	**ΔV33**	**ΔV43**	**ΔV14**	**ΔV24**	**ΔV34**	**ΔV44**
0.0001	0.0000	0.0000	0.0000	0.0010	0.0006	0.0006	0.0005

Sedangkan, hasil perhitungan perubahan bobot *hidden layer* untuk semua data pada iterasi kedua diperlihatkan secara lengkap pada Lampiran D.

f. Hitung MSE

Berikut adalah contoh manualisasi perhitungan mencari nilai *error* dan nilai MSE. Nilai MSE yang didapatkan untuk data ke-1 dan rata-rata *error* yang dihasilkan oleh data latih pada iterasi kedua diperlihatkan pada Tabel 4.32 Hasil Hitung *Error* pada Iterasi Kedua sebagai berikut:

$$error = \frac{1}{4} * ((0.85 - 0.6108) + (0.4835 - 0.7684) + (0.89 - 0.7113) + (0.8447 - 0.6970)) = 0.0702$$

Tabel 4.32 Hasil Hitung *Error* pada Iterasi Kedua

Error
0.0702
0.0285
-0.1697
-0.0937
0.1560
-0.5492
-0.2018
-0.2240
-0.2725

Sedangkan, nilai MSE yang didapatkan untuk 9 data latih pada iterasi kedua adalah:

$$MSE = \frac{1}{9} * \sum (error)^2 = 0.05937$$

1.3 Perhitungan Perbaikan Bobot

Pada perhitungan tahap ini akan dicari bobot terbaru pada setiap unit *hidden* maupun unit *output*. Berikut adalah langkah-langkah proses manualisasinya:

1. Perubahan Bobot pada Unit *Hidden Layer*

Contoh proses manualisasi perhitungan bobot pada unit *hidden layer* untuk data latih ke-2 sebagai berikut:

$$V_{11} = 0.6403 + 0.0014 = 0.6416$$

$$V_{21} = -0.6368 + 0.0008 = -0.6360$$

$$V_{31} = -0.7366 + 0.0008 = -0.7358$$

$$V_{41} = -0.1681 + 0.0007 = -0.1674$$

Berikut adalah hasil dari perubahan bobot pada unit *hidden layer* untuk data latih ke-2 iterasi kedua pada Tabel 4.33 Bobot Baru pada Unit *Hidden Layer* Iterasi Kedua sebagai berikut:

Tabel 4.33 Bobot Baru pada Unit *Hidden Layer* Iterasi Kedua

V11	**V21**	**V31**	**V41**	**V12**	**V22**	**V32**	**V42**
0.6416	-0.6360	-0.7358	-0.1674	0.1976	-0.1265	0.0898	-0.8110
V13	**V23**	**V33**	**V43**	**V14**	**V24**	**V34**	**V44**
-0.1157	-0.3935	0.6616	-0.7074	0.3496	-0.5907	-0.0932	-0.3504

2. Perubahan Bobot pada Unit *Output*

Contoh proses manualisasi perhitungan bobot pada unit *output layer* untuk data latih ke-2 sebagai berikut:

$$W_{11} = 0.4827 + 0.0166 = 0.4993$$

$$W_{21} = 0.3914 + 0.0166 = 0.4080$$

$$W_{31} = 0.0154 + 0.0161 = 0.0315$$

$$W_{41} = 0.09797 + 0.0167 = 0.1146$$

Berikut adalah hasil dari perubahan bobot pada unit *output layer* untuk data latih ke-2 iterasi kedua pada Tabel 4.34 Bobot Baru pada Unit *Output Layer* Iterasi Kedua sebagai berikut:

Tabel 4.34 Bobot Baru pada Unit *Output Layer* Iterasi Kedua

W11	W21	W31	W41	W12	W22	W32	W42
0.4993	0.4080	0.0315	0.1146	0.8154	0.5316	0.5902	0.6463
W13	**W23**	**W33**	**W43**	**W14**	**W24**	**W34**	**W44**
0.3574	0.7626	0.1669	0.7322	0.4695	0.2692	0.8021	0.3433

4.3.1.3 Perhitungan Pengujian Data Uji ANN

Pada manualisasi tahap ini, dilakukan iterasi sebanyak satu kali. Bobot terakhir yang diperoleh dari data ke-9 pada iterasi kedua dijadikan sebagai bobot terbaik yang akan digunakan pada proses pengujian data uji ANN. Proses pengujian dilakukan hanya pada tahapan *feedforward* saja. Data uji yang diproses sudah dinormalisasi terlebih dahulu sesuai pada Tabel 4.4. Berikut adalah proses pengujian data uji ANN:

a. Proses *Input Layer* ke *Hidden Layer*

Bobot yang digunakan untuk proses *input layer* ke *hidden layer* adalah bobot (V_{ij}) yang didapatkan dari data terakhir pada iterasi kedua. Berikut adalah bobot baru untuk data uji ke-1 yang diperlihatkan pada Tabel 4.35 Bobot *Input Layer* ke *Hidden Layer* Data Uji sebagai berikut:

Tabel 4.35 Bobot *Input Layer* ke *Hidden Layer* Data Uji

V11	V21	V31	V41	V12	V22	V32	V42
0.5842	-0.6723	-0.7834	-0.1894	0.1501	-0.1543	0.0508	-0.8243
V13	**V23**	**V33**	**V43**	**V14**	**V24**	**V34**	**V44**
-0.1533	-0.4185	0.6319	-0.7196	0.3077	-0.6180	-0.1253	-0.3635

b. Menjumlahkan Bobot Sinyal *Input*

Langkah selanjutnya adalah menjumlahkan bobot sinyal *input*. Contoh manualisasi perhitungan bobot sinyal *input* dari data uji ke-1 sebagai berikut:

$$Z_in_1 = (0.1 * 0.5842) + (0.9 * -0.6723) + (0.4586 * -0.7834) + (0.9 * -0.1894)$$
$$= -1.0764$$

$$Z_in_2 = (0.1 * 0.1501) + (0.9 * -0.1543) + (0.4586 * 0.0508) + (0.9 * -0.8243)$$
$$= -0.8424$$

$$Z_in_3 = (0.1 * -0.1533) + (0.9 * -0.4185) + (0.4586 * 0.6319) + (0.9 * -0.7196)$$
$$= -0.7497$$
$$Z_in_4 = (0.1 * 0.3077) + (0.9 * -0.6180) + (0.4586 * -0.1253) + (0.9 * -0.3635)$$
$$= -0.91$$

Berikut adalah hasil perhitungan penjumlahan bobot sinyal *input* yang didapatkan dari 6 data uji yang telah dinormalisasi pada Tabel 4.36 Jumlah Z_in_j Bobot Sinyal Data Uji sebagai berikut:

Tabel 4.36 Jumlah Z_in_j Bobot Sinyal *Input* Data Uji

Z_in1	Z_in2	Z_in3	Z_in4
-1.0764	-0.8424	-0.7497	-0.9100
-0.6507	-0.5702	-0.4294	-0.5146
0.0597	-0.4260	-0.6837	-0.2077
-0.4987	0.0279	0.2216	-0.1262
-0.0122	-0.5946	-0.6076	-0.1921
-0.4771	-0.1757	-0.2499	-0.3614

c. Menerapkan Fungsi Aktivasi Menghitung Sinyal *Input*

Fungsi aktivasi yang digunakan adalah fungsi aktivasi sigmoid biner. Berikut adalah contoh manualisasi proses perhitungan sinyal *input*:

e = bilangan *eural*, yaitu 2.71828

$$Z_1 = \frac{1}{(1 + 2.71828^{-(-1.0764)})} = 0.2542$$

$$Z_2 = \frac{1}{(1 + 2.71828^{-(-0.8424)})} = 0.3010$$

$$Z_3 = \frac{1}{(1 + 2.71828^{-(-0.7497)})} = 0.3209$$

$$Z_4 = \frac{1}{(1 + 2.71828^{-(-0.91)})} = 0.2870$$

Berikut adalah hasil sinyal *input* dengan menerapkan fungsi aktivasi sigmoid biner pada Tabel 4.37 Hasil Perhitungan Sinyal *Input* Data Uji sebagai berikut:

Tabel 4.37 Hasil Perhitungan Sinyal *Input* Data Uji

Z1	Z2	Z3	Z4
0.2542	0.3010	0.3209	0.2870
0.3428	0.3612	0.3943	0.3741
0.5149	0.3951	0.3354	0.4483
0.3779	0.5070	0.5552	0.4685
0.4970	0.3556	0.3526	0.4521
0.3829	0.4562	0.4379	0.4106

d. Bobot dari *Hidden Layer* ke *Output Layer*

Berikut adalah bobot dari *hidden layer* ke *output layer* untuk proses pengujian yang diperlihatkan pada Tabel 4.38 Bobot dari *Hidden Layer* ke *Output Layer* Data Uji sebagai berikut:

Tabel 4.38 Bobot dari *Hidden Layer* ke *Output Layer* Data Uji

W11	W21	W31	W41	W12	W22	W32	W42
0.4164	0.3097	-0.0741	0.0195	0.7309	0.4408	0.5011	0.5582
W13	**W23**	**W33**	**W43**	**W14**	**W24**	**W34**	**W44**
0.3309	0.7291	0.1342	0.7025	0.4283	0.2174	0.7490	0.2960

e. Menjumlahkan Setiap Bobot Sinyal *Output*

Langkah selanjutnya adalah menjumlahkan setiap bobot sinyal *output*. Contoh manualisasi perhitungan bobot sinyal *output* sebagai berikut:

$$Y_in_1 = (0.2542 * 0.4164) + (0.3010 * 0.3097) + (0.3209 * -0.0741) + (0.2870 * 0.0195)$$
$$= 0.1809$$

$$Y_in_2 = (0.2542 * 0.7309) + (0.3010 * 0.4408) + (0.3209 * 0.5011) + (0.2870 * 0.5582)$$
$$= 0.6395$$

$$Y_in_3 = (0.2542 * 0.3309) + (0.3010 * 0.7291) + (0.3209 * 0.1342) + (0.2870 * 0.7025)$$
$$= 0.5483$$

$$Y_in_4 = (0.2542 * 0.4283) + (0.3010 * 0.2174) + (0.3209 * 0.7490) + (0.2870 * 0.2960)$$
$$= 0.4996$$

Berikut adalah hasil menjumlahkan bobot sinyal *output* yang didapatkan dari 6 data uji pada Tabel 4.39 Jumlah Y_in_k Bobot Sinyal *Hidden Layer* ke *Output Layer* Data Uji sebagai berikut:

Tabel 4.39 Jumlah Y_in_k Bobot Sinyal *Hidden Layer* ke *Output Layer* Data Uji

Y_in_1	Y_in_2	Y_in_3	Y_in_4
0.1809	0.6395	0.5483	0.4996
0.2327	0.8162	0.6925	0.6314
0.3207	0.9688	0.8184	0.6903
0.2824	1.0394	0.8983	0.8265
0.2998	0.9490	0.7886	0.6881
0.2763	0.9296	0.8066	0.7127

f. Menerapkan Fungsi Aktivasi Menghitung Sinyal *Output*

Fungsi aktivasi yang digunakan adalah fungsi aktivasi sigmoid biner. Berikut adalah contoh manualisasi proses perhitungan sinyal *output* sebagai berikut:

$$Y_1 = \frac{1}{(1+2.71828^{-(0.1809)})} = 0.5451$$

$$Y_2 = \frac{1}{(1+2.71828^{-(0.6395)})} = 0.6546$$

$$Y_3 = \frac{1}{(1+2.71828^{-(0.5483)})} = 0.6337$$

$$Y_4 = \frac{1}{(1+2.71828^{-(0.4996)})} = 0.6224$$

Berikut adalah hasil perhitungan sinyal *output* dengan menerapkan fungsi aktivasi sigmoid biner pada Tabel 4.40 Hasil Perhitungan Sinyal *Output* Data Uji sebagai berikut:

Tabel 4.40 Hasil Perhitungan Sinyal *Output* Data Uji

Y_1	Y_2	Y_3	Y_4
0.5451	0.6546	0.6337	0.6224
0.5579	0.6934	0.6665	0.6528
0.5795	0.7249	0.6939	0.6660
0.5701	0.7387	0.7106	0.6956
0.5744	0.7209	0.6875	0.6655
0.5686	0.7170	0.6914	0.6710

4.3.1.4 Proses Denormalisasi

Proses denormalisasi adalah proses yang digunakan untuk mengubah nilai kandungan gizi susu ke dalam *range* nilai aslinya. Y_1 adalah *output* dari kandungan gizi susu berupa protein, Y_2 adalah *output* dari kandungan gizi susu berupa lemak, Y_3 adalah *output* dari kandungan gizi susu berupa laktosa dan Y_4 adalah *output* dari kandungan gizi susu berupa *density*. Berikut adalah proses perhitungan denormalisasi data yang diperlihatkan pada Tabel 4.41 Hasil Denormalisasi Data Uji sebagai berikut:

$$Y_1'' = \frac{(4.13-3.81)*(0.5451-0.1)}{0.8} + 3.81 = 3.9880$$

$$Y_2'' = \frac{(8.19-6.25)*(0.6546-0.1)}{0.8} + 6.25 = 7.5950$$

$$Y_3'' = \frac{(3.94-3.14)*(0.6337-0.1)}{0.8} + 3.14 = 3.6737$$

$$Y_4'' = \frac{(28.95-22.3)*(0.6224-0.1)}{0.8} + 22.3 = 26.6422$$

Tabel 4.41 Hasil Denormalisasi Data Uji

Y_1"	Y_2"	Y_3"	Y_4"
3.9880	7.5950	3.6737	26.6422
3.9932	7.6891	3.7065	26.8952
4.0018	7.7653	3.7339	27.0052
3.9981	7.7989	3.7506	27.2511
3.9998	7.7557	3.7275	27.0010
3.9975	7.7462	3.7314	27.0464

4.3.2 Perhitungan Manual *Genetic Algorithm* (GA)

Pada proses optimasi menggunakan algoritma GA, kandungan gizi susu yang dioptimalkan dan diminimalkan pada perhitungan manual ini adalah *output* dari Y_1 dan Y_2 setelah di denormalisasi dengan bobot badan kambing PE sebesar 32 kg. Y_1 adalah kandungan gizi susu yang berupa protein dan Y_2 adalah kandungan gizi susu yang berupa lemak. Proses perhitungan manualisasi untuk optimasi menggunakan GA sebagai berikut:

4.3.2.1 Menentukan Parameter GA

Penentuan nilai parameter GA yang akan digunakan pada proses perhitungan manualisasi sebagai berikut:

a. Jumlah populasi (*popsize*) = 5
b. *Crossover rate* (*cr*) = 0.4
c. *Mutation rate* (*mr*) = 0.2
d. Jumlah generasi = 1

4.3.2.2 Membangkitkan Populasi Awal

Proses ini dilakukan untuk membangkitkan populasi awal secara acak sebanyak *popsize* yang telah ditentukan dengan panjang kromosom sebesar banyaknya jumlah kandungan nutrisi yang ada pada pakan yang digunakan. Ditentukan *popsize* sebesar 5 populasi dengan *stringLen* sebanyak 4, yaitu Abu, PK, LK dan SK. Berikut adalah hasil inisialisasi populasi awal yang diperlihatkan pada Tabel 4.42 dimana P_1 sebagai *parent* ke-1, P_2 sebagai *parent* ke-2, P_3 sebagai *parent* ke-3, P_4 sebagai *parent* ke-4 dan P_5 sebagai *parent* ke-5:

Tabel 4.42 Inisialisasi Populasi Awal

No	Individu	*Chromosome*			
		Abu	PK	LK	SK
1	P_1	0.658	0.471	0.119	0.862
2	P_2	0.646	0.502	0.133	0.946
3	P_3	0.603	0.479	0.117	0.839
4	P_4	0.711	0.453	0.127	0.819
5	P_5	0.682	0.449	0.131	0.837

4.3.2.3 Proses Reproduksi Crossover

Proses reproduksi *crossover* dilakukan untuk mendapatkan individu baru (*offspring*) hasil reproduksi dari dua *parent* yang telah dipilih secara acak kemudian disilangkan antara *parent* yang pertama dan kedua. Metode *crossover* yang digunakan pada perhitungan manualisasi GA adalah *extended intermediate crossover*. Pada Persamaan 2-17 dan 2-18 telah dijelaskan bahwa proses *extended intermediate crossover* membutuhkan nilai alpha (α) dalam proses perhitungannya. Nilai alpha (α) dipilih secara acak pada interval [-0.25, 1.25]. Berikut adalah nilai alpha (α) yang telah ditetapkan di awal secara acak yang diperlihatkan pada Tabel 4.43 Penentuan Nilai Alpha (α) sebagai berikut:

Tabel 4.43 Penentuan Nilai Alpha (α)

Alpha 1 (α_1)	Alpha 2 (α_2)	Alpha 3 (α_3)	Alpha 4 (α_4)
0.665	-0.167	1.194	-0.232

Misalkan yang terpilih sebagai induk adalah P_1 dan P_4 dengan nilai alpha (α) yang telah diperlihatkan pada Tabel 4.43, maka akan dihasilkan dua *offspring*, yaitu C_1 dan C_2. Berikut adalah contoh manualisasi perhitungan proses reproduksi *crossover* menggunakan metode *extended intermediate crossover* sebagai berikut:

C_1 :
$$x_1 = 0.658 + (0.665 * (0.711 - 0.658)) = 0.6932$$
$$x_2 = 0.471 + (-0.167 * (0.453 - 0.471)) = 0.474$$
$$x_3 = 0.119 + (1.194 * (0.127 - 0.119)) = 0.1286$$
$$x_4 = 0.862 + (-0.232 * (0.819 - 0.862)) = 0.872$$

C_2 :
$$x_1 = 0.711 + (0.665 * (0.658 - 0.711)) = 0.6758$$
$$x_2 = 0.453 + (-0.167 * (0.471 - 0.453)) = 0.45$$
$$x_3 = 0.127 + (1.194 * (0.119 - 0.127)) = 0.1174$$
$$x_4 = 0.819 + (-0.232 * (0.862 - 0.819)) = 0.809$$

Berikut adalah hasil proses reproduksi *crossover* menggunakan metode *extended intermediate crossover* yang diperlihatkan pada Tabel 4.44 Hasil Perhitungan Manual Proses *Crossover* sebagai berikut:

Tabel 4.44 Hasil Perhitungan Manual Proses *Crossover*

Individu	*Chromosome*			
	Abu	PK	LK	SK
P_1	0.658	0.471	0.119	0.862
P_4	0.711	0.453	0.127	0.819
C_1	0.6932	0.4740	0.1286	0.8720
C_2	0.6758	0.4500	0.1174	0.8090

4.3.2.4 Proses Reproduksi Mutasi

Proses reproduksi mutasi dilakukan untuk mendapatkan individu baru (*offspring*) yang diperoleh dari salah satu gen yang mengalami penyimpangan

dari kromosom induknya. Metode mutasi yang digunakan pada perhitungan manualisasi GA adalah metode *random mutation*. Pada persamaan 2-19 telah dijelaskan bahwa proses *random mutation* membutuhkan nilai *r*, max_j dan min_j dalam proses perhitungannya. Nilai *r* dipilih secara acak pada suatu interval tertentu. Misalkan *range r* dipilih secara acak pada interval [-0.1, 0.1] dan nilai *r* yang dipilih adalah 0.075, max_j = 7.3, min_j = 0 serta individu yang terpilih sebagai induk adalah P_3 dengan gen yang terpilih nomer 2 (x_2), maka contoh manualisasi perhitungan proses reproduksi mutasi menggunakan metode *random mutation* sebagai berikut:

C_3 : $x_1 = 0.603(tetap)$

$x_2 = 0.479 + (0.075 * (7.3 - 0)) = 1.0265$

$x_3 = 0.117(tetap)$

$x_4 = 0.839(tetap)$

Berikut adalah hasil proses reproduksi mutasi menggunakan metode *random mutation* yang diperlihatkan pada Tabel 4.45 Hasil Perhitungan Manual Proses Mutasi sebagai berikut:

Tabel 4.45 Hasil Perhitungan Manual Proses Mutasi

Individu	*Chromosome*			
	Abu	PK	LK	SK
P_3	0.603	0.479	0.117	0.839
C_3	0.603	1.0265	0.117	0.839

4.3.2.5 Perhitungan Nilai Fitness

Proses perhitungan nilai *fitness* dilakukan untuk mendapatkan hasil evaluasi berdasarkan suatu nilai fungsi obektif tertentu. Nilai *fitness* tersebut didapatkan dari proses pelatihan GA terhadap proses pengujian ANN dengan menggunakan Persamaan 2-15, yaitu rumus denormalisasi data. Misalkan, pengguna ingin mengoptimalkan kandungan gizi susu protein dan meminimalkan kandungan gizi lemak, maka rumus fitness yang digunakan sebagai berikut:

$$Fitness = denormalisasi_protein + \frac{1}{denormalisasi_lemak}$$

Bobot terakhir yang didapatkan dari proses pengujian ANN akan digunakan sebagai bobot terbaru untuk proses pelatihan GA. Proses pelatihan GA terhadap ANN hanya dilakukan pada tahapan *feedforward* saja hingga didapatkan hasil dari proses denormalisasinya. Berikut adalah contoh manualisasi perhitungan denormalisasi protein dan lemak yang akan digunakan untuk mencari nilai *fitness* sebagai berikut:

$$Denormalisasi\,\mathrm{Pr}otein_P_1 = \frac{(4.13 - 3.81) * (0.5691 - 0.1)}{0.8} + 3.81 = 3.9977$$

$$DenormalisasiLemak_P_1 = \frac{(8.19 - 6.25) * (0.7010 - 0.1)}{0.8} + 6.25 = 7.7074$$

$$Denormalisasi\ \Pr otein_P_2 = \frac{(4.13-3.81)*(0.5667-0.1)}{0.8} + 3.81 = 3.9967$$

$$DenormalisasiLemak_P_2 = \frac{(8.19-6.25)*(0.6944-0.1)}{0.8} + 6.25 = 7.6915$$

$$Denormalisasi\ \Pr otein_P_3 = \frac{(4.13-3.81)*(0.5684-0.1)}{0.8} + 3.81 = 3.9974$$

$$DenormalisasiLemak_P_3 = \frac{(8.19-6.25)*(0.7001-0.1)}{0.8} + 6.25 = 7.7052$$

$$Denormalisasi\ \Pr otein_P_4 = \frac{(4.13-3.81)*(0.5710-0.1)}{0.8} + 3.81 = 3.9984$$

$$DenormalisasiLemak_P_4 = \frac{(8.19-6.25)*(0.7056-0.1)}{0.8} + 6.25 = 7.7186$$

$$Denormalisasi\ \Pr otein_P_5 = \frac{(4.13-3.81)*(0.5701-0.1)}{0.8} + 3.81 = 3.9981$$

$$DenormalisasiLemak_P_5 = \frac{(8.19-6.25)*(0.7040-0.1)}{0.8} + 6.25 = 7.7146$$

$$Denormalisasi\ \Pr otein_C_1 = \frac{(4.13-3.81)*(0.5694-0.1)}{0.8} + 3.81 = 3.9977$$

$$DenormalisasiLemak_C_1 = \frac{(8.19-6.25)*(0.7012-0.1)}{0.8} + 6.25 = 7.7080$$

$$Denormalisasi\ \Pr otein_C_2 = \frac{(4.13-3.81)*(0.5708-0.1)}{0.8} + 3.81 = 3.9983$$

$$DenormalisasiLemak_C_2 = \frac{(8.19-6.25)*(0.7054-0.1)}{0.8} + 6.25 = 7.7181$$

$$Denormalisasi\ \Pr otein_C_3 = \frac{(4.13-3.81)*(0.5584-0.1)}{0.8} + 3.81 = 3.9934$$

$$DenormalisasiLemak_C_3 = \frac{(8.19-6.25)*(0.6701-0.1)}{0.8} + 6.25 = 7.6325$$

Berdasarkan nilai yang diperoleh dari proses perhitungan denormalisasi protein dan lemak untuk semua individu, maka langkah selanjutnya adalah melakukan proses perhitungan nilai *fitness.* Berikut adalah contoh manualisasi perhitungan nilai *fitness* untuk semua individu sebagai berikut:

$$Fitness_P_1 = 3.9977 + \frac{1}{7.7074} = 4.1274$$

$$Fitness_P_2 = 3.9967 + \frac{1}{7.6915} = 4.1267$$

$$Fitness_P_3 = 3.9974 + \frac{1}{7.7052} = 4.1271$$

$$Fitness_P_4 = 3.9984 + \frac{1}{7.7186} = 4.1280$$

$$Fitness_P_5 = 3.9981 + \frac{1}{7.7146} = 4.1277$$

$$Fitness_C_1 = 3.9977 + \frac{1}{7.7080} = 4.1275$$

$$Fitness_C_2 = 3.9983 + \frac{1}{7.7181} = 4.1279$$

$$Fitness_C_3 = 3.9934 + \frac{1}{7.6325} = 4.1244$$

Kandungan gizi susu yang ingin dioptimasi adalah protein dan kandungan gizi susu yang diminimalkan pada perhitungan manualisasi GA adalah lemak. Berikut adalah hasil perhitungan nilai *fitness* yang diperlihatkan pada Tabel 4.46 Hasil Perhitungan Nilai *Fitness* sebagai berikut:

Tabel 4.46 Hasil Perhitungan Nilai *Fitness*

Individu	*Chromosome*				*Fitness*
	Abu	PK	LK	SK	
P1	0.658	0.471	0.119	0.862	4.1274
P2	0.646	0.502	0.133	0.946	4.1267
P3	0.603	0.479	0.117	0.839	4.1271
P4	0.711	0.453	0.127	0.819	4.1280
P5	0.682	0.449	0.131	0.837	4.1277
C1	0.6932	0.4740	0.1286	0.8720	4.1275
C2	0.6758	0.4500	0.1174	0.8090	4.1279
C3	0.6030	1.0265	0.1170	0.8390	4.1244

4.3.2.6 Proses Evaluasi dan Seleksi

Proses seleksi merupakan proses untuk memilih individu dari himpunan populasi gabungan yang dipertahankan pada generasi selanjutnya. Metode seleksi yang digunakan pada perhitungan manualisasi GA adalah *elitism selection*. Metode ini dilakukan untuk menggabungkan semua individu, yaitu *parent* dan *offspring*, kemudian diseleksi dengan cara mengurutkan dari nilai *fitness* yang tertinggi ke terendah. Berikut adalah hasil proses evaluasi dan seleksi yang diperlihatkan pada Tabel 4.47 Hasil Proses Evaluasi dan Seleksi sebagai berikut:

Tabel 4.47 Hasil Proses Evaluasi dan Seleksi

Individu	*Chromosome*				*Fitness*
	Abu	PK	LK	SK	
P4	0.711	0.453	0.127	0.819	4.1280
C2	0.676	0.45	0.117	0.809	4.1279
P5	0.682	0.449	0.131	0.837	4.1277
C1	0.693	0.474	0.129	0.872	4.1275

P1	0.658	0.471	0.119	0.862	4.1274
P3	0.603	0.479	0.117	0.839	4.1271
P2	0.646	0.502	0.133	0.946	4.1267
C3	0.603	1.0265	0.117	0.839	4.1244

Berdasarkan hasil dari Tabel 4.47, jika iterasi maksimal yang dimasukan adalah 1, maka individu terbaik yang dijadikan sebagai solusi dari permasalahan pemodelan komposisi pakan kambing PE untuk optimasi kandungan gizi susu menggunakan ANN dan GA adalah P_4 dengan nilai *fitness* sebesar 4.1280. Diketahui bahwa perbandingan banyaknya rumput odot dan rumput raja yang digunakan dari total pemberian pakan adalah 70%: 30% dengan bobot kambing PE sebesar 32 kg. Total kandungan nutrisi dari rumput odot sebesar 0.606 per 1 kg dan total kandungan nutrisi dari rumput raja sebesar 0.604 per 1 kg berdasarkan penjelasan yang ada di sub bab 2.5 dengan contoh perhitungan sebagai berikut:

$$TotalKandungan_Odot = 0.1697 + 0.1703 + 0.0176 + 0.2484 = 0.606$$
$$TotalKandungan_Raja = 0.186 + 0.135 + 0.035 + 0.2484 = 0.604$$

Total kandungan nutrisi dari rumput odot dan rumput raja adalah 1.2104 per 2 kg dengan contoh perhitungan sebagai berikut:

$$TotalKandunganNutrisi_Odot - Raja = 0.606 + 0.604 = 1.2104$$

Sedangkan, total kandungan nutrisi dari rumput odot dan rumput raja yang diperoleh dari proses ANN dan GA dengan *fitness* terbesar dari P_4 adalah 2.11 per 2 kg dengan contoh perhitungan total kandungan nutrisi rumput odot dan rumput raja sebagai berikut:

$$TotalKandunganNutrisi_ANN - GA = 0.711 + 0.453 + 0.127 + 0.819 = 2.11$$

Hasil yang diperoleh pada proses manualisasi ANN dan GA adalah untuk mendapatkan kandungan gizi protein optimal dan lemak yang diminimalkan dengan hasil 4.1280%, maka pakan hijauan berupa rumput odot sebaiknya diberikan sebanyak 1.2203 kg dan rumput raja sebaiknya diberikan sebanyak 0.523 kg untuk diberikan pada seekor kambing PE dengan BB sebesar 32 kg dengan contoh perhitungan manualisasinya sebagai berikut:

$$TotalPemberianPakan_Odot - Raja \ \ (kg) = 2.11/1.2104 = 1.7432$$

$$PemberianPakan_Odot \ \ (kg) = \frac{70}{100} * 1.7432 = 1.2203$$

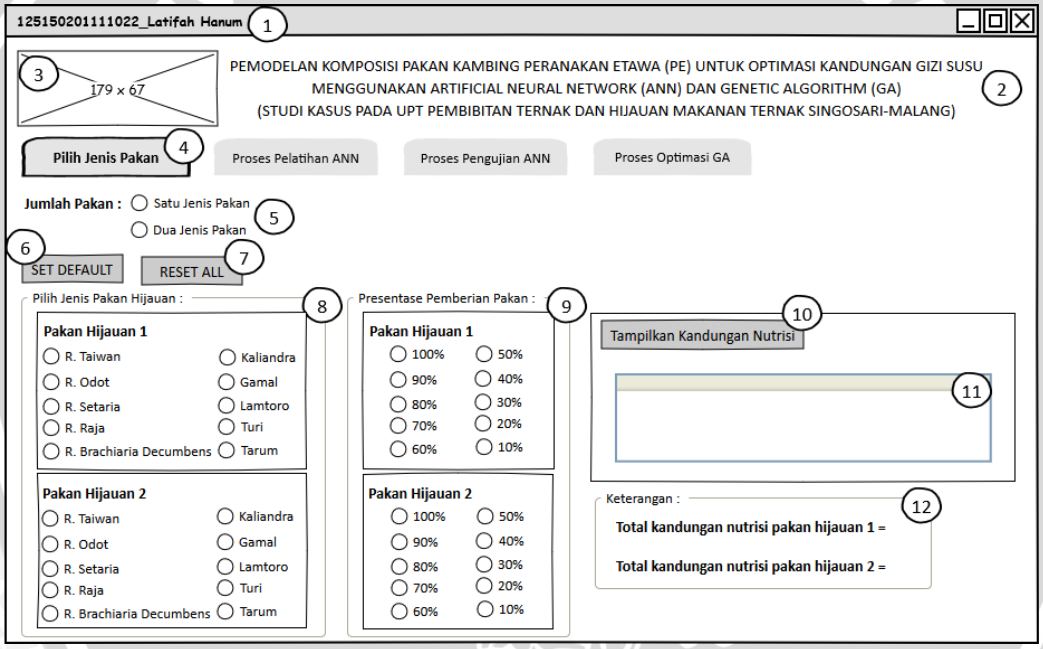
$$PemberianPakan_Raja \ \ (kg) = \frac{30}{100} * 1.7432 = 0.523$$

4.4 Perancangan Antarmuka

Pada sub bab ini menjelaskan tentang perancangan antarmuka untuk menggambarkan keadaan sebenarnya dari implementasi sistem yang akan dibangun. Terdapat 4 halaman yang akan dibuat pada perancangan antarmuka pemodelan komposisi pakan kambing PE untuk optimasi kandungan gizi susu menggunakan ANN dan GA, yaitu proses pemilihan jenis pakan hijauan, proses pelatihan ANN, proses pengujian ANN dan optimasi metode GA. Proses rancangan antarmuka ini akan dijelaskan lebih detail pada sub bab 4.4.1 – 4.4.4.

4.4.1 Proses Pemilihan Jenis Pakan Hijauan

Proses pemilihan jenis pakan hijauan digunakan untuk memasukkan jenis pakan hijauan yang digunakan oleh pengguna beserta presentase pemberian pakan setiap pakan hijauannya. Berikut adalah gambar rancangan antarmuka proses pemilihan jenis pakan hijauan yang diperlihatkan pada Gambar 4.16 Rancangan Antarmuka Proses Pemilihan Jenis Pakan Hijauan sebagai berikut.



Gambar 4.16 Rancangan Antarmuka Proses Pemilihan Jenis Pakan Hijauan

Berikut adalah penjelasan dari Gambar 4.16 Rancangan Antarmuka Proses Pemilihan Jenis Pakan Hijauan:

1. Judul w*indow*, yaitu berisikan NIM dan nama pembuat program.
2. Judul program yang telah dibuat.
3. *Image* yang akan diisikan dengan logo dari Fakultas Ilmu Komputer UB.
4. *Tab* untuk melakukan proses pemilihan jenis pakan hijauan.
5. *Radio button* untuk memilih jumlah pakan yang akan digunakan.
6. Tombol *SET DEFAULT* untuk mengatur jumlah pakan, jenis pakan dan presentase pakan hijauan secara otomatis.

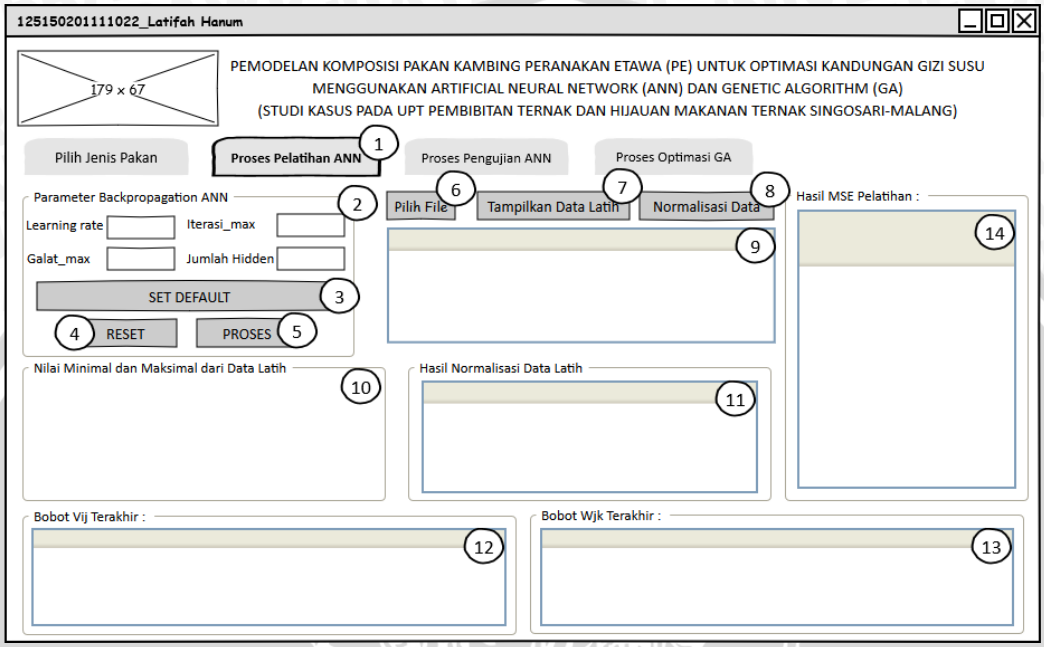
7. Tombol *RESET* untuk me-*reset* jumlah pakan, jenis pakan dan presentase pemberian pakan yang telah dimasukan sebelumnya.
8. *Group box* untuk menampilkan daftar dari jenis pakan hijauan yang digunakan.
9. *Group box* untuk menampilkan daftar presentase pemberian pakan yang dapat diberikan.
10. Tombol TAMPILKAN KANDUNGAN NUTRISI untuk menampilkan kandungan nutrisi pakan hijauan yang telah dipilih sebelumnya dalam satu kilogram.
11. Tabel untuk menampilkan nama pakan hijauan, kandungan abu, PK, LK dan SK dari pakan hijauan yang telah dipilih sebelumnya oleh pengguna.
12. *Group box* untuk menampilkan total kandungan nutrisi dari pakan hijauan yang telah dipilih sebelumnya oleh pengguna.

4.4.2 Proses Pelatihan ANN

Proses pelatihan ANN digunakan untuk memperoleh target dan bobot yang memiliki nilai MSE terkecil. Pada rancangan antarmuka proses pelatihan ANN, pengguna dapat memilih data yang digunakan sebagai data latih serta memasukkan nilai parameter ANN. Berikut adalah penjelasan dari rancangan antarmuka proses pelatihan ANN yang diperlihatkan pada Gambar 4.17 Rancangan Antarmuka Proses Pelatihan ANN:

1. *Tab* untuk melakukan proses pelatihan ANN *backpropagation*.
2. *Group box* untuk memasukkan nilai parameter ANN pada proses pelatihan ANN yang meliputi:
   a. Nilai *learning rate* (α)
   b. Galat maksimal
   c. Jumlah iterasi maksimal
   d. Jumlah *hidden layer*
3. Tombol *SET DEFAULT* untuk mengatur nilai parameter ANN secara otomatis sesuai dengan hasil pengujian yang didapatkan.
4. Tombol *RESET* untuk me-*reset* nilai yang telah dimasukan sebelumnya.
5. Tombol PROSES untuk melakukan perhitungan proses pelatihan ANN.
6. Tombol Pilih File untuk memilih data latih yang ingin digunakan.
7. Tombol Tampilkan Data Latih untuk mengambil data latih yang digunakan dan menampilkan data latih pada tabel.
8. Tombol Normalisasi Data untuk menampilkan hasil dari proses normalisasi data dan diperlihatkan pada tabel.
9. Tabel untuk menampilkan data latih yang telah di *load*.

10. *Group box* untuk menampilkan nilai minimal dan maksimal dari data latih.
11. Tabel untuk menampilkan hasil normalisasi data latih.
12. Tabel untuk menampilkan hasil bobot V_{ij} terakhir dari proses pelatihan ANN.
13. Tabel untuk menampilkan hasil bobot W_{jk} terakhir proses pelatihan ANN.
14. Tabel untuk menampilkan hasil MSE dari proses pelatihan ANN.



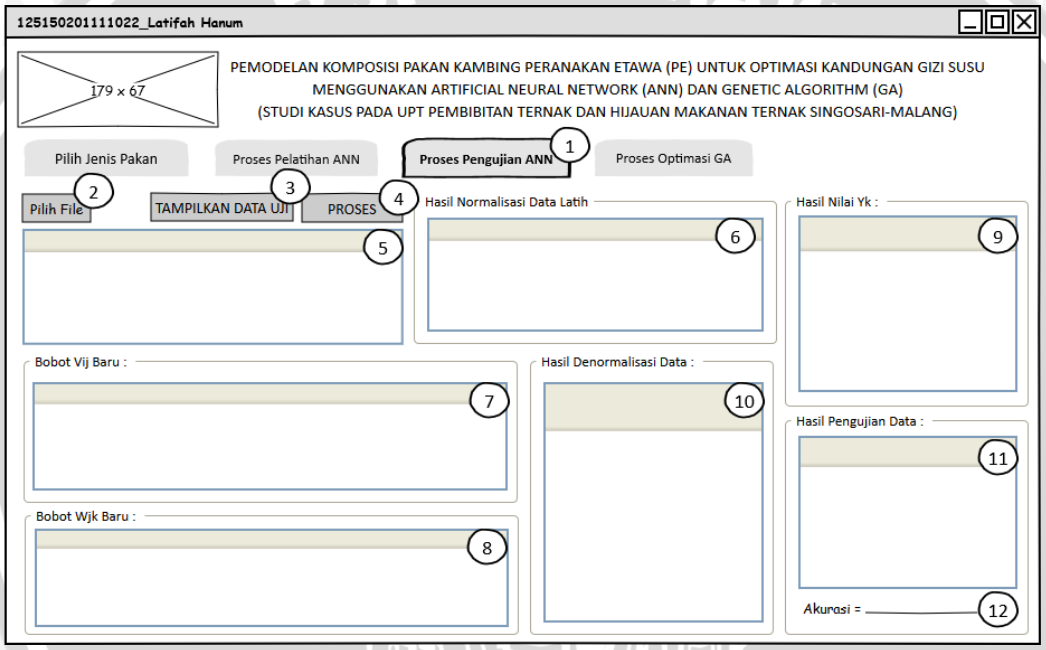
Gambar 4.17 Rancangan Antarmuka Proses Pelatihan ANN

4.4.3 Proses Pengujian ANN

Proses selanjutnya adalah proses pengujian ANN terhadap data uji yang dilakukan untuk menghitung tingkat akurasi agar dapat mengetahui bobot yang dihasilkan dari proses pelatihan ANN sudah bagus atau belum. Proses pengujian ANN hanya dilakukan pada tahapan *feedforward* saja. Rancangan antarmuka ini akan menampilkan data uji, hasil normalisasi dari data uji serta proses yang diperoleh dari *feedforward* berdasarkan jenis pakan hijauan beserta kandungan nutrisi dari masing-masing jenis pakan hijauan yang telah dipilih pada proses pelatihan ANN pada *Tab* yang sebelumnya. Berikut adalah penjelasan dari rancangan antarmuka proses pengujian ANN yang diperlihatkan pada Gambar 4.18 Rancangan Antarmuka Proses Pengujian ANN:

1. *Tab* untuk melakukan proses pengujian ANN *backpropagation*.
2. Tombol Pilih File untuk memilih data uji yang ingin digunakan.
3. Tombol TAMPILKAN DATA UJI untuk mengambil data uji yang digunakan dan menampilkan data uji pada tabel.
4. Tombol PROSES untuk melakukan perhitungan proses pengujian ANN.
5. Tabel untuk menampilkan data uji yang telah di *load*.

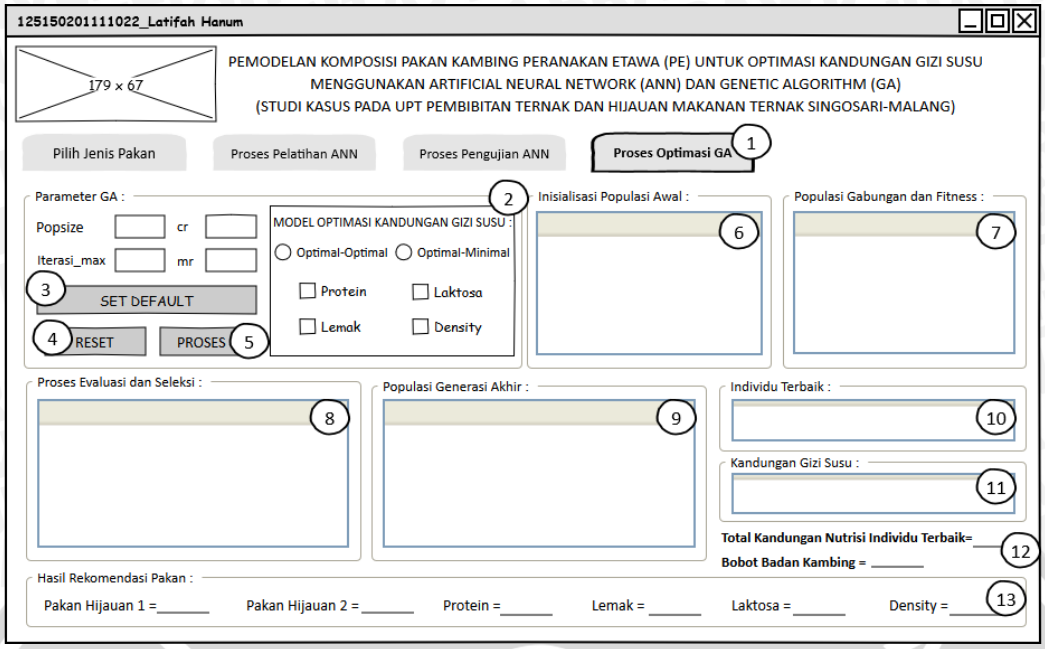
6. Tabel untuk menampilkan hasil normalisasi data uji.
7. Tabel untuk menampilkan hasil bobot V_{ij} baru yang didapatkan dari proses pelatihan ANN.
8. Tabel untuk menampilkan hasil bobot W_{jk} baru yang didapatkan dari proses pelatihan ANN.
9. Tabel untuk menampilkan hasil nilai Y_k pada program yang didapat dari proses pengujian ANN.
10. Tabel untuk menampilkan hasil denormalisasi data dari data uji.
11. Tabel untuk menampilkan hasil pengujian datanya.
12. Label untuk menampilkan hasil akurasi yang diperoleh pada proses pengujian ANN terhadap data uji.



Gambar 4.18 Rancangan Antarmuka Proses Pengujian ANN

4.4.4 Optimasi Metode GA

Pada halaman ini, pengguna akan memasukkan nilai parameter GA yang meliputi jumlah populasi (*popsize*), jumlah generasi (iterasi *max*), *crossover rate* (*cr*) dan *mutation rate* (*mr*). selain itu, pengguna juga diharuskan memilih model optimasi yang digunakan, yaitu optimal-optimal atau optimal-minimal dan kandungan gizi susu yang ingin dioptimasi atau diminimalisasi untuk menghasilkan nilai yang terbaik. Halaman ini akan menampilkan kombinasi kandungan nutrisi dari jenis pakan hijauan yang telah dipilih untuk mendapatkan hasil yang optimal dari kandungan gizi susu yang telah dipilih sebelumnya. Berikut adalah gambaran rancangan antarmuka proses optimasi metode GA yang diperlihatkan pada Gambar 4.19 Rancangan Antarmuka Proses Optimasi GA.



Gambar 4.19 Rancangan Antarmuka Proses Optimasi GA

Berikut adalah penjelasan dari Gambar 4.19 Rancangan Antarmuka Proses Optimasi GA:

1. *Tab* untuk melakukan proses optimasi menggunakan *Genetic Algorithm* (GA).
2. *Group box* untuk memasukkan nilai parameter GA yang diisikan pada proses optimasi GA yang meliputi:
   a. Jumlah populasi (*popsize*)
   b. Jumlah generasi (iterasi *max*)
   c. *Crossover rate* (*cr*)
   d. *Mutation rate* (*mr*)

   Serta memasukkan model optimasi dan kandungan gizi susu yang ingin dioptimalkan atau diminimalkan oleh pengguna.
3. Tombol *SET DEFAULT* untuk mengatur agar nilai parameter GA diisikan secara otomatis berdasarkan hasil pengujian yang telah dilakukan.
4. Tombol *RESET* untuk me-*reset* nilai parameter GA serta kandungan gizi susu yang dioptimalkan dan diminimalkan.
5. Tombol PROSES untuk melakukan perhitungan proses optimasi GA.
6. Tabel untuk menampilkan hasil dari proses inisialisasi populasi awal pada iterasi pertama.
7. Tabel untuk menampilkan populasi gabungan beserta nilai *fitness* setiap individu pada iterasi pertama.
8. Tabel untuk menampilkan hasil dari proses evaluasi dan seleksi setiap individu pada iterasi pertama.

9. Tabel untuk menampilkan populasi generasi akhir pada iterasi terakhir yang akan dijadikan sebagai solusi nantinya.
10. Tabel untuk menampilkan individu terbaik dengan nilai *fitness* terbesar yang dijadikan sebagai solusi akhir dari program.
11. Tabel untuk menampilkan hasil denormalisasi dari individu terbaik terhadap kandungan gizi susu yang berupa protein, lemak, laktosa dan *density*.
12. *Label* untuk menampilkan total kandungan nutrisi yang didapatkan dari individu terbaik beserta bobot badan kambingnya.
13. *Group box* untuk menampilkan hasil luaran akhir yang akan didapatkan pengguna, antara lain banyaknya pemberian pakan hijauan pertama, pakan hijauan kedua, presentase kenaikan protein, penurunan lemak, kenaikan laktosa dan kenaikan *density* yang didapatkan setelah melalui proses ANN dan GA.

4.5 Perancangan Pengujian Algoritma

Pada sub bab ini, hal yang akan dijelaskan adalah membuat rancangan pengujian algoritma. Rancangan pengujian algoritma dilakukan untuk mengetahui prediksi yang dihasilkan proses pengujian ANN dan hasil optimasi yang diberikan oleh GA. Rancangan pengujian algoritma ini dilakukan secara bertahap dan sekuensial, yaitu hasil yang diperoleh dari pengujian yang sebelumnya akan digunakan pada pengujian yang selanjutnya sebagai nilai *default*.

4.5.1 Pengujian Jumlah Iterasi

Pengujian ini dilakukan dengan memberikan jumlah iterasi yang berbeda-beda guna memperoleh jumlah iterasi berapa yang menghasilkan bobot terbaik dengan nilai MSE terendah. Jumlah iterasi yang akan diujikan merupakan jumlah iterasi kelipatan 100 dengan *range* antara 100 sampai 1000 iterasi. Pengujian jumlah iterasi dilakukan sebanyak 10 kali percobaan. Berikut adalah rancangan pengujian jumlah iterasi pada Tabel 4.48 Rancangan Pengujian Jumlah Iterasi.

Tabel 4.48 Rancangan Pengujian Jumlah Iterasi

Jumlah Iterasi	Nilai MSE Percobaan ke-*i*										Rata-rata MSE
	1	2	3	4	5	6	7	8	9	10	
100											
200											
300											
400											
500											
600											
700											
800											
900											
1000											

4.5.2 Pengujian Jumlah *Hidden Layer*

Pengujian jumlah *hidden layer* dilakukan untuk mengenali target dan *input*. Pada pengujian ini akan dipilih jumlah *hidden layer* yang menghasilkan nilai MSE terendah. Variasi jumlah *hidden layer* yang digunakan pada pengujian ini adalah 1 sampai 10 *hidden layer*. Pengujian jumlah *hidden layer* dilakukan sebanyak 10 kali percobaan. Berikut adalah rancangan pengujian jumlah *hidden layer* pada Tabel 4.49 Rancangan Pengujian Jumlah *Hidden Layer*.

Tabel 4.49 Rancangan Pengujian Jumlah *Hidden Layer*

Hidden Layer	Nilai MSE Percobaan ke-*i*										Rata-rata MSE
	1	2	3	4	5	6	7	8	9	10	
1											
2											
3											
4											
5											
6											
7											
8											
9											
10											

4.5.3 Pengujian Nilai *Learning Rate*

Pengujian nilai *learning rate* dilakukan untuk mengetahui pengaruh variasi nilai *learning rate* terhadap MSE yang dihasilkan. Nilai *learning rate* berkisar antara 0.1 sampai dengan 0.9. Pada pengujian ini akan dipilih nilai *learning rate* yang menghasilkan nilai MSE terendah. Pengujian nilai *learning rate* dilakukan sebanyak 10 kali percobaan. Proses ini dilakukan untuk mendapatkan nilai yang mendekati target sehingga hasil pengujiannya akan lebih baik. Berikut adalah rancangan pengujian nilai *learning rate* pada Tabel 4.50 Rancangan Pengujian Nilai *Learning Rate*.

Tabel 4.50 Rancangan Pengujian Nilai *Learning Rate*

Learning Rate	Nilai MSE Percobaan ke-*i*										Rata-rata MSE
	1	2	3	4	5	6	7	8	9	10	
0.9											
0.8											
0.7											
0.6											
0.5											
0.4											
0.3											
0.2											
0.1											

4.5.4 Pengujian Ukuran Populasi

Pengujian ukuran populasi dilakukan untuk mengetahui ukuran populasi terbaik yang akan digunakan hingga didapatkan hasil ukuran populasi yang konvergensi, yaitu nilai *fitness* yang dihasilkan cenderung sama dan tidak lagi ditemukan nilai *fitness* yang memiliki perbedaan yang signifikan. Ukuran populasi yang akan diuji adalah kelipatan 10 dengan *range* antara 10 sampai 100. Pengujian ukuran populasi dilakukan sebanyak 10 kali perocbaan. Berikut adalah rancangan pengujian ukuran populasi pada Tabel 4.51 Rancangan Pengujian Ukuran Populasi.

Tabel 4.51 Rancangan Pengujian Ukuran Populasi

Banyak Populasi	Nilai *Fitness* Percobaan ke-*i*										Rata-rata *Fitness*
	1	2	3	4	5	6	7	8	9	10	
10											
20											
30											
40											
50											
60											
70											
80											
90											
100											

4.5.5 Pengujian Jumlah Generasi

Pengujian jumlah generasi dilakukan untuk mengetahui banyaknya generasi yang digunakan untuk menghasilkan komposisi pakan dengan kandungan gizi susu yang optimal. Jumlah generasi yang akan digunakan adalah jumlah generasi yang menghasilkan nilai *fitness* tertinggi. Jumlah generasi yang akan diuji adalah kelipatan 100 dengan *range* antara 100 sampai 1000. Pengujian jumlah generasi dilakukan sebanyak 10 kali percobaan. Berikut adalah rancangan pengujian jumlah generasi pada Tabel 4.52 Rancangan Pengujian Jumlah Generasi.

Tabel 4.52 Rancangan Pengujian Jumlah Generasi

Jumlah Generasi	Nilai *Fitness* Percobaan ke-*i*										Rata-rata *Fitness*
	1	2	3	4	5	6	7	8	9	10	
100											
200											
300											
400											
500											
600											
700											
800											
900											
1000											

4.5.6 Pengujian Kombinasi *Crossover Rate* dan *Mutation Rate*

Pengujian kombinasi *crossover rate* (*cr*) dan *mutation rate* (*mr*) dilakukan untuk mengetahui apakah kombinasi *cr* dan *mr* akan berpengaruh terhadap hasil luaran pada proses optimasi menggunakan *Genetic Algorithm* (GA), sehingga diperoleh kombinasi *cr* dan *mr* dengan rata-rata nilai *fitness* yang terbaik. Pengujian kombinasi *cr* dan *mr* dilakukan sebanyak 10 kali percobaan. Berikut adalah rancangan pengujian kombinasi *cr* dan *mr* pada Tabel 4.53 Rancangan Pengujian Kombinasi *cr* dan *mr*.

Tabel 4.53 Rancangan Pengujian Kombinasi *cr* dan *mr*

cr	*mr*	Nilai *Fitness* Percobaan ke-*i*										Rata-rata *Fitness*
		1	2	3	4	5	6	7	8	9	10	
1	0											
0.9	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	0.9											

4.5.7 Pengujian Kombinasi Variasi *Output* yang Dioptimalkan

Pengujian kombinasi variasi *output* yang dioptimalkan dilakukan untuk mengetahui grafik nilai *fitness* yang dihasilkan oleh kombinasi variasi *output* dengan kombinasi berupa kandungan gizi susu protein yang dioptimalkan dan lemak yang diminimalkan. Pengujian kombinasi variasi *output* dilakukan sebanyak 10 kali percobaan dengan menggunakan parameter ANN dan GA yang sudah optimal. Berikut adalah rancangan pengujian kombinasi variasi *output* pada Tabel 4.54 Rancangan Pengujian Kombinasi Variasi *Output* yang Dioptimalkan dan rancangan pengujian kenaikan protein dan penurunan lemak pada Tabel 4.55 Rancangan Pengujian Kenaikan Protein dan Penurunan Lemak.

Tabel 4.54 Rancangan Pengujian Kombinasi Variasi *Output* yang Dioptimalkan

Kandungan Gizi Susu		Nilai *Fitness* Percobaan ke-*i*										Rata-rata *Fitness*
Optimalkan	Minimalkan	1	2	3	4	5	6	7	8	9	10	
Protein	Lemak											

Tabel 4.55 Rancangan Pengujian Kenaikan Protein dan Penurunan Lemak

Kasus Optimasi (Protein Optimal dan Lemak Minimal)	Nilai Presentase Percobaan ke-*i*										Rata-rata Presentase
	1	2	3	4	5	6	7	8	9	10	
Protein Dioptimalkan											
Lemak Diminimalkan											

BAB 5 IMPLEMENTASI

Pada bab ini akan menjelaskan tentang implementasi sistem berdasarkan analisis kebutuhan dan proses perancangan sistem yang telah dibuat sebelumnya. Pembahasan pada bab ini terdiri dari penjelasan implementasi algoritma *Artificial Neural Network* (ANN) yang berupa proses pelatihan ANN dan pengujian ANN, implementasi algoritma *Genetic Algorithm* (GA) untuk proses optimasi serta implementasi antarmuka sistem.

5.1 Implementasi Algoritma *Artificial Neural Network* (ANN)

Implementasi algoritma *Artificial Neural Network* (ANN) pada sistem terdiri dari dua bagian utama, yaitu proses pelatihan *Artificial Neural Network* (ANN) dan proses pengujian *Artificial Neural Network* (ANN).

5.1.1 Proses Pelatihan *Artificial Neural Network* (ANN)

Proses pelatihan ANN dimulai dari proses normalisasi data, proses *feedforward*, proses *backpropagation* dan proses perbaikan bobot hingga diperoleh nilai MSE setiap iterasinya.

5.1.1.1 Proses Normalisasi Data

Normalisasi data merupakan suatu proses yang digunakan untuk menormalisasikan data latih dan data uji dimana nilai atribut setiap datanya diskalakan dengan rentang nilai antara 0.1 sampai 0.9. Langkah yang dilakukan adalah mencari nilai minimal dan maksimal dari masing-masing data menggunakan fungsi `Nilai_Min` dan `Nilai_Max`, kemudian menampilkan hasil normalisasi datanya menggunakan fungsi `Normalisasi_Data`, sesuai Gambar 5.1 *Source Code* Normalisasi Data.

```
1  public double Nilai_Max(double[] data) {
2      double max = data[0];
3
4      for (int i = 1; i < data.length; i++) {
5          if (data[i] > max) {
6              max = data[i];
7          }
8      }
9
10     return max;
11 }
12
13 public double Nilai_Min(double[] data) {
14     double min = data[0];
15
16     for (int i = 1; i < data.length; i++) {
17         if (data[i] < min) {
18             min = data[i];
19         }
20     }
21
22     return min;
```

```
23  }
24
25  public double[][] Normalisasi_Data(double[][] data, double[]
26  minmax) {
27      double[][]        hasil_normalisasi        =        new
28      double[data.length][4];
29
30      for (int i = 0; i < data.length; i++) {
31          for (int j = 0; j < 4; j++) {
32              hasil_normalisasi[i][j] = (0.8 * (data[i][j] -
33              minmax[j]) / (minmax[j + 4] - minmax[j])) +
34              0.1;
35          }
36      }
37
38      return hasil_normalisasi;
39  }
```

Gambar 5.1 *Source Code* Proses Normalisasi Data

Penjelasan Gambar 5.1 *Source Code* Proses Normalisasi Data sebagai berikut:

1. Baris 5-7 adalah proses pencarian nilai maksimal dari data yang telah dimasukkan dan dipilih sebelumnya.
2. Baris 17-19 adalah proses pencarian nilai minimal dari data yang telah dimasukkan dan dipilih sebelumnya.
3. Baris 32-34 adalah proses untuk melakukan normalisasi data dengan pengulangan sebanyak jumlah data dan panjang setiap datanya, yaitu 4.

5.1.1.2 Proses Feedforward

Pada implementasi ini, sistem akan meneruskan sinyal masukan ke *hidden layer* dengan menggunakan fungsi aktivasi *sigmoid biner*. Langkah pertama adalah membangkitkan bobot dari *input layer* ke *hidden layer* dan *hidden layer* ke *output layer*. Fungsi untuk membangkitkan bobot ini dijelaskan pada fungsi `Bobot_Vij` dan `Bobot_Wjk`. Langkah kedua adalah meneruskan bobot-bobot tersebut untuk menjumlahkan sinyal masukan pada *input* dan *output* yang dijelaskan pada fungsi `Z_in` dan `Y_in`. Langkah ketiga adalah menghitung fungsi aktivasi untuk masing-masing sinyal *input* dan *output* menggunakan fungsi `Fungsi_Sigmoid_Z` dan `Fungsi_Sigmoid_Y`.

```
 1  public double[][] Bobot_Vij(int iterasi, int batas_min, int
 2  batas_max,  int  jum_hidden_layer,  int  data,  double[][]
 3  bobot_sebelum, double[][] delta_bobot) {
 4      int input = 4;
 5      double          bobot_Vij[][]          =          new
 6      double[input][jum_hidden_layer];
 7
 8      if (iterasi == 0) {
 9          if (data == 0) {
10              for (int j = 0; j < jum_hidden_layer; j++) {
11                  for (int i = 0; i < input; i++) {
12                      bobot_Vij[i][j]   =   (Math.random()   *
13                      (batas_max - batas_min)) + batas_min;
14                  }
15              }
16          } else {
```

```
            for (int j = 0; j < jum_hidden_layer; j++) {
                double hasil = 0;
                for (int i = 0; i < input; i++) {
                    hasil     =     (bobot_sebelum[i][j]     +
                    delta_bobot[i][j]);
                    bobot_Vij[i][j] = hasil;
                }
            }
        }
    } else {
        if (data == 0) {
            for (int j = 0; j < jum_hidden_layer; j++) {
                for (int i = 0; i < input; i++) {
                    bobot_Vij[i][j] = bobot_sebelum[i][j];
                }
            }
        } else {
            for (int j = 0; j < jum_hidden_layer; j++) {
                double hasil = 0;
                for (int i = 0; i < input; i++) {
                    hasil     =     (bobot_sebelum[i][j]     +
                    delta_bobot[i][j]);
                    bobot_Vij[i][j] = hasil;
                }
            }
        }
    }

    return bobot_Vij;
}

public double[][] Bobot_Wjk(int iterasi, int batas_min, int
batas_max,  int  jum_hidden_layer,  int  data,  double[][]
bobot_sebelum, double[][] delta_bobot) {
    int output = 4;
    double          bobot_Wjk[][]          =          new
    double[jum_hidden_layer][output];

    if (iterasi == 0) {
        if (data == 0) {
            for (int j = 0; j < jum_hidden_layer; j++) {
                for (int k = 0; k < output; k++) {
                    bobot_Wjk[j][k]    =    (Math.random()    *
                    (batas_max - batas_min)) + batas_min;
                }
            }
        } else {
            for (int j = 0; j < jum_hidden_layer; j++) {
                double hasil = 0;
                for (int k = 0; k < output; k++) {
                    hasil     =     (bobot_sebelum[j][k]     +
                    delta_bobot[j][k]);
                    bobot_Wjk[j][k] = hasil;
                }
            }
        }
    } else {
        if (data == 0) {
            for (int j = 0; j < jum_hidden_layer; j++) {
```

```
                for (int k = 0; k < output; k++) {
                    bobot_Wjk[j][k] = bobot_sebelum[j][k];
                }
            }
        } else {
            for (int j = 0; j < jum_hidden_layer; j++) {
                double hasil = 0;
                for (int k = 0; k < output; k++) {
                    hasil     =     (bobot_sebelum[j][k]     +
                    delta_bobot[j][k]);
                    bobot_Wjk[j][k] = hasil;
                }
            }
        }
    }

    return bobot_Wjk;
}

public   double[]   Z_in(double[][]   Input_X,   double[][]
Bobot_Vij, int jum_hidden_layer, int data) {
    int jum_input_Xi = 4;
    double[] Z_in = new double[jum_hidden_layer];

    for (int j = 0; j < jum_hidden_layer; j++) {
        double hasil = 0;
        for (int i = 0; i < jum_input_Xi; i++) {
            hasil  =  (Input_X[data][i]  *  Bobot_Vij[i][j])  +
             hasil;
            Z_in[j] = hasil;
        }
    }

    return Z_in;
}

public double[] Y_in(double[] Z, double[][] Bobot_Wjk, int
jum_hidden_layer) {
    int jum_output = 4;
    double[] Y_in = new double[output];

    for (int k = 0; k < jum_output; k++) {
        double hasil = 0;
        for (int j = 0; j < jum_hidden_layer; j++) {
            hasil = (Z[j] * Bobot_Wjk[j][k]) + hasil;
            Y_in[k] = hasil;
        }
    }

    return Y_in;
}

public   double[]   Fungsi_Sigmoid_Z(double[]   Z_in,   int
jum_hidden_layer) {
    double[] fungsi_sigmoid = new double[jum_hidden_layer];

    for (int j = 0; j < jum_hidden_layer; j++) {
        fungsi_sigmoid[j]  =  1  /  (1  +  Math.pow(2.71828,  -
        (Z_in[j])));
```

```
135         }
136
137         return fungsi_sigmoid;
138     }
139
140     public    double[]    Fungsi_Sigmoid_Y(double[]    Y_in,    int
141     jum_hidden_layer) {
142         int jum_output = 4;
143         double[] fungsi_sigmoid = new double[output];
144
145         for (int k = 0; k < jum_output; k++) {
146             fungsi_sigmoid[k]  =  1  /  (1  +  Math.pow(2.71828,  -
147             (Y_in[k])));
148         }
149
150         return fungsi_sigmoid;
151     }
```

Gambar 5.2 *Source Code* Proses *Feedforward*

Penjelasan Gambar 5.2 *Source Code* Proses *Feedforward* sebagai berikut:

1. Baris 12-13 adalah proses pembangkitan bobot V_{ij} dari *input layer* ke *hidden layer* untuk iterasi pertama dan data ke-0 yang dibangkitkan secara acak.
2. Baris 20-22 adalah proses pembangkitan bobot V_{ij} dari *input layer* ke *hidden layer* untuk iterasi pertama dan data yang tidak ke-0 dengan cara menjumlahkan bobot sebelumnya dengan perubahan bobot V_{ij}.
3. Baris 30 adalah proses pembangkitan bobot V_{ij} dari *input layer* ke *hidden layer* untuk iterasi selanjutnya dan data ke-0 yang diambil dari bobot sebelumnya.
4. Baris 37-39 adalah proses pembangkitan bobot V_{ij} dari *input layer* ke *hidden layer* untuk iterasi selanjutnya dan data yang tidak ke-0 dengan cara menjumlahkan bobot sebelumnya dengan perubahan bobot V_{ij}.
5. Baris 59-60 adalah proses pembangkitan bobot W_{jk} dari *hidden layer* ke *output layer* untuk iterasi pertama dan data ke-0 yang dibangkitkan secara acak.
6. Baris 67-69 adalah proses pembangkitan bobot W_{jk} dari *hidden layer* ke *output layer* untuk iterasi pertama dan data yang tidak ke-0 dengan cara menjumlahkan bobot sebelumnya dengan perubahan bobot W_{jk}.
7. Baris 77 adalah proses pembangkitan bobot W_{jk} dari *hidden layer* ke *output layer* untuk iterasi selanjutnya dan data ke-0 yang diambil dari bobot sebelumnya.
8. Baris 84-86 adalah proses pembangkitan bobot W_{jk} dari *hidden layer* ke *output layer* untuk iterasi selanjutnya dan data yang tidak ke-0 dengan cara menjumlahkan bobot sebelumnya dengan perubahan bobot W_{jk}.
9. Baris 103-105 adalah proses untuk menjumlahkan bobot sinyal *input*.
10. Baris 120-121 adalah proses untuk menjumlahkan bobot sinyal *output*.

11. Baris 133-134 adalah proses untuk menghitung fungsi aktivasi sinyal *input*.

12. Baris 146-147 adalah proses untuk menghitung fungsi aktivasi sinyal *output*.

5.1.1.3 Proses Backpropagation

Pada proses *backpropagation*, sistem akan menghitung hasil dari *output layer* dan memeriksa semua kesalahan ke *hidden layer*, yaitu memeriksa kesalahan yang dihasilkan oleh *output* terhadap target *output* yang sebenarnya. Langkah pertama adalah menghitung *error output* dengan menggunakan fungsi `Error_Output`. Langkah kedua adalah menghitung perubahan bobot pada *output layer* dengan menggunakan fungsi `Delta_Wjk`. Langkah ketiga adalah menjumlahkan delta *input* dengan menggunakan fungsi `Jumlah_Delta_Input`. Langkah keempat adalah menghitung informasi *error* dengan menggunakan fungsi `Informasi_Error`. Langkah kelima adalah menghitung perubahan bobot pada *hidden layer* dengan menggunakan fungsi `Delta_Vij`. Proses terakhir adalah menghitung nilai *error* setiap data dan menampilkan hasil MSE setiap *epoch*-nya dengan menggunakan fungsi `Hitung_Error` dan `Hitung_MSE`.

```
1   public double[] Error_Output(double[][] target, double[] Y,
2   int data) {
3       int jum_output = 4;
4       double error_output[] = new double[output];
5
6       for (int k = 0; k < jum_output; k ++) {
7           error_output[k] = (target[data][k] - Y[k]) * Y[k] *
8           (1 - Y[k]);
9       }
10
11      return error_output;
12  }
13
14  public double[][] Delta_Wjk(double[] error_output, double[]
15  Z, int jum_hidden_layer, double learning_rate) {
16      int jum_output = 4;
17      double          delta_Wjk[][]          =          new
18      double[jum_hidden_layer][jum_output];
19
20      for (int j = 0; j < jum_hidden_layer; j++) {
21          for (int k = 0; k < jum_output; k++) {
22              delta_Wjk[j][k]      =      learning_rate      *
23              error_output[k] * Z[j];
24          }
25      }
26
27      return delta_Wjk;
28  }
29
30  public  double[]  Jumlah_Delta_Input(double[]  error_output,
31  double[][] bobot_Wjk, int jum_hidden_layer) {
32      int jum_output = 4;
33      double          jum_delta_input[]          =          new
34      double[jum_hidden_layer];
35
36      for (int j = 0; j < jum_hidden_layer; j++) {
37          double hasil = 0;
```

```
38          for (int k = 0; k < jum_output; k ++) {
39              hasil = (error_output[k] * bobot_Wjk[j][k]) +
40              hasil;
41          }
42          jum_delta_input[j] = hasil;
43      }
44
45      return jum_delta_input;
46  }
47
48  public  double[]  Informasi_Error(double[]  jum_delta_input,
49  double[] Z, int jum_hidden_layer) {
50      double          informasi_error[]          =          new
51      double[jum_hidden_layer];
52
53      for (int j = 0; j < jum_hidden_layer; j++) {
54          informasi_error[j] = jum_delta_input[j] * Z[j] * (1
55          - Z[j]);
56      }
57
58      return informasi_error;
59  }
60
61  public    double[][]    Delta_Vij(double[]    informasi_error,
62  double[][]    input_X,    int    jum_hidden_layer,    double
63  learning_rate, int data) {
64      int jum_input = 4;
65      double[][]          delta_Vij          =          new
66      double[jum_input][jum_hidden_layer];
67
68      for (int j = 0; j < jum_hidden_layer; j++) {
69          for (int i = 0; i < jum_input; i++) {
70              delta_Vij[i][j]     =     learning_rate      *
71              informasi_error[j] * input_X[data][i];
72          }
73      }
74
75      return delta_Vij;
76  }
77
78  public  double  Hitung_Error(double[][]  target,  double[]  Y,
79  int data) {
80      double error = 0;
81      int output = 4;
82      double jum_output = 4;
83      double hasil = 0;
84
85      for (int j = 0; j < output; j++) {
86          hasil = (target[data][j] - Y[j]) + hasil;
87      }
88
89      error = (1 / jum_output) * hasil;
90
91      return error;
92  }
93
94  public double Hitung_MSE(double[] error) {
95      double MSE = 0;
96
```

```
97          for (int i = 0; i < error.length; i++) {
98              MSE += (Math.pow(error[i], 2) / error.length);
99          }
100
101         return MSE;
102     }
```

Gambar 5.3 *Source Code* Proses *Backpropagation*

Penjelasan Gambar 5.3 *Source Code* Proses *Backpropagation* sebagai berikut:

1. Baris 7-8 adalah proses menghitung nilai *error output* antara nilai *output* yang didapatkan dengan target *output* yang seharusnya.
2. Baris 22-23 adalah proses menghitung perubahan bobot W_{jk} pada *output layer*.
3. Baris 39-40 adalah proses menjumlahkan delta *input layer.*
4. Baris 42 adalah proses untuk menyimpan hasil perhitungan penjumlahan delta *input* ke dalam array satu dimensi.
5. Baris 54-55 adalah proses menghitung informasi *error* dari *input layer* ke *hidden layer*.
6. Baris 70-71 adalah proses menghitung perubahan bobot pada V_{ij} pada *hidden layer*.
7. Baris 89 adalah proses menghitung nilai *error* untuk setiap data.
8. Baris 98 adalah proses menghitung nilai MSE untuk setiap *epoch*.

5.1.2 Proses Pengujian *Artificial Neural Network* (ANN)

Proses pengujian ANN dimulai dari proses normalisasi data dan hanya terdapat satu proses saja, yaitu proses *feedforward* hingga diperoleh hasil denormalisasi setiap datanya. Nilai bobot yang diberikan pada proses pengujian ANN adalah bobot terakhir yang didapat pada proses pelatihan dengan menggunakan fungsi `Bobot_Vij_Pelatihan` dan `Bobot_Wjk_Pelatihan`. Proses pengujian ANN dilakukan dengan cara memanggil fungsi-fungsi proses *feedforward* yang ada pada proses pelatihan ANN yang telah dijelaskan pada Gambar 5.2 *Source Code* Proses *Feedforward*. Setelah proses *feedforward* pada pengujian ANN selesai, maka langkah selanjutnya adalah melakukan proses denormalisasi data dengan menggunakan fungsi `Denormalisasi` dan `Simpan_Hasil_Denormalisasi` untuk menyimpan hasil denormalisasi ke dalam format .txt. Proses denormalisasi data dilakukan agar hasil *output* dari data uji dan *training* GA dapat diubah ke *range* nilai yang sebenarnya (*real*).

```
1   public      double[][]      Bobot_Vij_Pelatihan(double[][]
2   bobot_sebelum, int jum_hidden_layer) {
3       int jum_input = 4;
4       double          bobot_Vij[][]           =           new
5       double[jum_input][jum_hidden_layer];
6
7       for (int j = 0; j < jum_hidden_layer; j++) {
8           for (int i = 0; i < jum_input; i++) {
```

```
            bobot_Vij[i][j] = bobot_sebelum[i][j];
        }
    }

    return bobot_Vij;
}

public      double[][]      Bobot_Wjk_Pelatihan(double[][]
bobot_sebelum, int jum_hidden_layer) {
    int jum_output = 4;
    double          bobot_Wjk[][]          =          new
    double[jum_hidden_layer][jum_output];

    for (int j = 0; j < jum_hidden_layer; j++) {
        for (int k = 0; k < jum_output; k++) {
            bobot_Wjk[j][k] = bobot_sebelum[j][k];
        }
    }

    return bobot_Wjk;
}

public  double[]  Denormalisasi(double[]  Y,  String  path,
String jenis_data, int jum_data) throws IOException {
    int jum_output = 4;
    double[] denormalisasi = new double[4];
    double[][]                kandungan_gizi                =
    PREPROS.Kandungan_Gizi_Susu(path,          jenis_data,
    jum_data);

    double[]          proteinArray          =          new
    double[kandungan_gizi.length];
    double[]          lemakArray          =          new
    double[kandungan_gizi.length];
    double[]          laktosaArray          =          new
    double[kandungan_gizi.length];
    double[]          densityArray          =          new
    double[kandungan_gizi.length];

    for (int i = 0; i < kandungan_gizi.length; i++) {
        proteinArray[i] = kandungan_gizi[i][0];
        lemakArray[i] = kandungan_gizi[i][1];
        laktosaArray[i] = kandungan_gizi[i][2];
        densityArray[i] = kandungan_gizi[i][3];
    }

    double minProtein = ND.Nilai_Min(proteinArray);
    double minLemak = ND.Nilai_Min(lemakArray);
    double minLaktosa = ND.Nilai_Min(laktosaArray);
    double minDensity = ND.Nilai_Min(densityArray);

    double maxProtein = ND.Nilai_Max(proteinArray);
    double maxLemak = ND.Nilai_Max(lemakArray);
    double maxLaktosa = ND.Nilai_Max(laktosaArray);
    double maxDensity = ND.Nilai_Max(densityArray);

    double[] minData = {minProtein, minLemak, minLaktosa,
    minDensity};
    double[] maxData = {maxProtein, maxLemak, maxLaktosa,
```

```
68          maxDensity};
69
70          for (int k = 0; k < jum_output; k++) {
71              denormalisasi[k] = (((maxData[k] - minData[k]) *
72              (Y[k] - 0.1)) / 0.8) + minData[k];
73          }
74
75          return denormalisasi;
76      }
```

Gambar 5.4 *Source Code* Proses Pengujian ANN dan Denormalisasi Data

Penjelasan Gambar 5.4 *Source Code* Proses Pengujian ANN dan Denormalisasi Data sebagai berikut:

1. Baris 9 adalah proses untuk menampilkan bobot dari *input layer* ke *hidden layer* (bobot V_{ij}) yang terakhir pada proses pelatihan ANN.
2. Baris 24 adalah proses untuk menampilkan bobot dari *hidden layer* ke *output layer* (bobot W_{jk}) yang terakhir pada proses pelatihan ANN.
3. Baris 39-53 adalah membuat inisialisasi variabel baru, yaitu `proteinArray`, `lemakArray`, `laktosaArray` dan `densityArray` yang digunakan untuk menyimpan hasil pembacaan dari baris 49-52.
4. Baris 55-58 adalah proses untuk mencari nilai minimal dari `proteinArray`, `lemakArray`, `laktosaArray` dan `densityArray`.
5. Baris 60-63 adalah proses untuk mencari nilai maksimal dari `proteinArray`, `lemakArray`, `laktosaArray` dan `densityArray`.
6. Baris 65-66 adalah proses untuk menyimpan nilai minimal dari `proteinArray`, `lemakArray`, `laktosaArray` dan `densityArray` ke dalam array satu dimensi yang baru bernama `minData`.
7. Baris 67-68 adalah proses untuk menyimpan nilai maksimal dari `proteinArray`, `lemakArray`, `laktosaArray` dan `densityArray` ke dalam array satu dimensi yang baru bernama `maxData`.
8. Baris 71-72 adalah proses untuk menghitung hasil denormalisasi data.

5.2 Implementasi Algoritma *Genetic Algorithm* (GA)

Implementasi algoritma *Genetic Algorithm* (GA) pada sistem terdiri dari inisialisasi populasi awal sebanyak *popsize* yang telah ditentukan sebelumnya, proses *crossover* yang berupa *extended intermediate crossover*, proses mutasi yang berupa *random mutation*, proses *training* GA ke dalam ANN yang berupa proses *feedforward* saja, proses evaluasi dan seleksi menggunakan *elitism selection*, proses *random injection*, proses yang menampilkan individu yang lolos ke generasi berikutnya dan hasil luaran yang didapatkan dari individu terbaik.

5.2.1 Proses Inisialisasi Populasi Awal

Langkah pertama pada proses optimasi GA adalah inisialisasi atau membangkitkan populasi awal sebanyak *popsize* yang telah ditentukan

sebelumnya. Populasi yang dibangkitkan memiliki rentang nilai berupa batas bawah dan batas atas yang ditentukan berdasarkan jenis pakan hijauan yang telah dipilih oleh pengguna yang dijelaskan pada fungsi `Batas_Bawah` dan `Batas_Atas`. Populasi tersebut akan dibangkitkan secara acak berdasarkan batas bawah dan batas atas yang telah ditentukan sebelumnya. Proses inisialisasi populasi awal dijelaskan pada fungsi `Generate_Populasi`.

```
1  public double[] Batas_Bawah(double BB, int stringLen, int
2  jum_pakan, int pakan1, int pakan2) {
3       double[][] kandungan = KNP.Kandungan_Rumput();
4
5       double[] hitung_kandungan = new double[stringLen];
6       double[] batas_bawah = new double[stringLen];
7
8       if (jum_pakan == 1) {
9           for (int i = 0; i < stringLen; i++) {
10              hitung_kandungan[i] = BB * kandungan[pakan1 -
11              1][i] * 0.1;
12              batas_bawah[i]                                  =
13               Double.valueOf(df.format(hitung_kandungan[i]));
14          }
15      } else if (jum_pakan == 2) {
16          for (int i = 0; i < stringLen; i++) {
17              hitung_kandungan[i] = (BB * kandungan[pakan1 -
18              1][i] * 0.1) + (BB * kandungan[pakan2 - 1][i] *
19              0.1);
20              batas_bawah[i]                                  =
21               Double.valueOf(df.format(hitung_kandungan[i]));
22          }
23      } else {
24          System.out.println("Error !");
25      }
26
27      return batas_bawah;
28 }
29
30 public double[] Batas_Atas(double BB, int stringLen, int
31 jum_pakan, int pakan1, int pakan2) {
32      double[][] kandungan = KNP.Kandungan_Rumput();
33
34      double[] hitung_kandungan = new double[stringLen];
35      double[] batas_atas = new double[stringLen];
36
37      if (jum_pakan == 1) {
38          for (int i = 0; i < stringLen; i++) {
39              hitung_kandungan[i] = BB * kandungan[pakan1 -
40              1][i] * 0.12;
41              batas_atas[i]                                   =
42               Double.valueOf(df.format(hitung_kandungan[i]));
43          }
44      } else if (jum_pakan == 2) {
45          for (int i = 0; i < stringLen; i++) {
46              hitung_kandungan[i] = (BB * kandungan[pakan1 -
47              1][i] * 0.12) + (BB * kandungan[pakan2 - 1][i] *
48              0.12);
49              batas_atas[i]                                   =
50               Double.valueOf(df.format(hitung_kandungan[i]));
```

```
51              }
52          } else {
53              System.out.println("Error !");
54          }
55
56          return batas_atas;
57      }
58
59      public   double[][]   Generate_Populasi(double[]   batas_bawah,
60      double[] batas_atas, int popsize, int stringLen) {
61          double[][] gen_populasi = new double[popsize][stringLen];
62          double[] BatasBawahRand = new double[stringLen];
63          double[] BatasAtasRand = new double[stringLen];
64
65          for (int i = 0; i < stringLen; i++) {
66              BatasBawahRand[i] = batas_bawah[i];
67              BatasAtasRand[i] = batas_atas[i];
68          }
69
70          for (int i = 0; i < popsize; i++) {
71              for (int j = 0; j < stringLen; j++) {
72                  gen_populasi[i][j]    =    (BatasBawahRand[j]    +
73                   (Math.random()     *     (BatasAtasRand[j]     -
74                   BatasBawahRand[j])));
75              }
76          }
77
78          return gen_populasi;
79      }
```

Gambar 5.5 *Source Code* Proses Inisialisasi Populasi Awal

Penjelasan Gambar 5.5 *Source Code* Proses Inisialisasi Populasi Awal sebagai berikut:

1. Baris 10-11 adalah proses penghitungan kandungan nutrisi pakan jika jumlah pakan yang digunakan adalah satu untuk method `Batas_Bawah`.
2. Baris 12-13 adalah proses penyimpanan hasil dari perhitungan kandungan nutrisi pakan jika jumlah pakan yang digunakan adalah satu ke dalam array satu dimensi yang bernama `batas_bawah`.
3. Baris 17-19 adalah proses penghitungan kandungan nutrisi pakan jika jumlah pakan yang digunakan adalah dua untuk method `Batas_Bawah`.
4. Baris 20-21 adalah proses penyimpanan hasil dari perhitungan kandungan nutrisi pakan jika jumlah pakan yang digunakan adalah dua ke dalam array satu dimensi yang bernama `batas_bawah`.
5. Baris 39-40 adalah proses penghitungan kandungan nutrisi pakan jika jumlah pakan yang digunakan adalah satu untuk method `Batas_Atas`.
6. Baris 41-42 adalah proses penyimpanan hasil dari perhitungan kandungan nutrisi pakan jika jumlah pakan yang digunakan adalah satu ke dalam array satu dimensi yang bernama `batas_atas`.
7. Baris 46-48 adalah proses penghitungan kandungan nutrisi pakan jika jumlah pakan yang digunakan adalah dua untuk method `Batas_Atas`.

8. Baris 49-50 adalah proses penyimpanan hasil dari perhitungan kandungan nutrisi pakan jika jumlah pakan yang digunakan adalah dua ke dalam array satu dimensi yang bernama `batas_atas`.
9. Baris 65-68 adalah proses untuk menyimpan hasil batas bawah dan batas atas yang didapat dari method `Batas_Bawah` dan `Batas_Atas` dengan penggunaan parameter `batas_bawah` dan `batas_atas`.
10. Baris 72-74 adalah proses inisialisasi atau membangkitkan populasi awal secara acak dengan *range* antara batas bawah dan batas atas sebanyak *popsize* dan *stringlen* yang telah ditetapkan sebelumnya.

5.2.2 Proses *Crossover*

Salah satu metode *crossover* yang diimplementasikan ke dalam sistem adalah *extended intermediate crossover*. Hasil *offspring* yang dihasilkan dari satu kali proses *crossover* adalah dua buah *offspring*. Banyaknya hasil *offspring* yang dihasilkan bergantung pada nilai *crossover rate* (*cr*) yang telah ditentukan sebelumnya. Banyaknya *parent* yang digunakan untuk proses *crossover* adalah dua buah *parent* yang dipilih secara acak oleh sistem. Kemudian, melakukan proses *crossover* terhadap dua *parent* yang telah dipilih dan nilai *alpha* yang telah ditentukan sebelumnya secara acak oleh fungsi `Nilai_Alpha`. Proses *extended intermediate crossover* dijelaskan pada fungsi `Extended_Intermediate_Crossover`.

```
1  public double[][] Extended_Intermediate_Crossover(double[][]
2  gen_populasi, double[] alpha, int popsize, double Cr, int
3  stringLen) {
4       int index = 0;
5       int jum_crossover = (int) Math.ceil(Cr * popsize);
6
7       if (jum_crossover % 2 != 0) {
8             jum_crossover++;
9       }
10
11      //Proses crossover akan dilakukan sebanyak jum_crossover
12      int hasil_crossover = (int) Math.ceil(jum_crossover /
13      2);
14      double[][] offspringCrossover = new
15      double[jum_crossover][stringLen];
16
17      //Proses pengacakan pemilihan parent yang akan dilakukan
18      oleh crossover
19      for (int i = 0; i < hasil_crossover; i++) {
20          int parent1 = (int) (Math.random() * (popsize - 1));
21          int parent2 = parent1;
22
23          while (parent1 == parent2) {
24              parent2 = (int) (Math.random() * (popsize - 1));
25          }
26
27          //Proses extended intermediate dilakukan dengan
28          nilai alpha yang sudah diacak sebelumnya
29          for (int j = 0; j < stringLen; j++) {
30              offspringCrossover[index][j] =
```

```
31                  gen_populasi[parent1][j]     +     (alpha[j]     *
32                  (gen_populasi[parent2][j]                        -
33                  gen_populasi[parent1][j]));
34                  if        (offspringCrossover[index][j]          >
35                  BatasAtasRand[j]) {
36                      offspringCrossover[index][j]                 =
37                      BatasAtasRand[j];
38                  }   else   if   (offspringCrossover[index][j]    <
39                  BatasBawahRand[j]) {
40                      offspringCrossover[index][j]                 =
41                      BatasBawahRand[j];
42                  }
43
44                  offspringCrossover[index      +      1][j]       =
45                  gen_populasi[parent2][j]     +     (alpha[j]     *
46                  (gen_populasi[parent1][j]                        -
47                  gen_populasi[parent2][j]));
48                  if   (offspringCrossover[index     +    1][j]    >
49                  BatasAtasRand[j]) {
50                      offspringCrossover[index     +     1][j]     =
51                      BatasAtasRand[j];
52                  } else if (offspringCrossover[index  +  1][j]    <
53                  BatasBawahRand[j]) {
54                      offspringCrossover[index     +     1][j]     =
55                      BatasBawahRand[j];
56                  }
57              }
58              index++;
59              index++;
60          }
61
62          return offspringCrossover;
63      }
```

Gambar 5.6 *Source Code* Proses *Crossover*

Penjelasan Gambar 5.6 *Source Code* Proses *Crossover* sebagai berikut:

1. Baris 5 adalah variabel yang digunakan untuk menyimpan hasil perhitungan jumlah *offspring* yang seharusnya dihasilkan.
2. Baris 20-25 adalah proses pengacakan pemilihan dua *parent* sebanyak jumlah *crossover* yang telah ditentukan.
3. Baris 30-33 adalah proses *extended intermediate crossover* untuk *offspring* pertama dengan nilai *alpha* yang telah diacak sebelumnya sebanyak *stringlen* yang telah ditetapkan.
4. Baris 34-37 adalah jika hasil proses *extended intermediate crossover* untuk *offspring* pertama > `BatasAtasRand`, maka hasil proses *extended intermediate crossover* dibuat sama dengan nilai `BatasAtasRand`.
5. Baris 38-42 adalah jika hasil proses *extended intermediate crossover* untuk *offspring* pertama < `BatasBawahRand`, maka hasil proses *extended intermediate crossover* dibuat sama dengan nilai `BatasBawahRand`.
6. Baris 44-47 adalah proses *extended intermediate crossover* untuk *offspring* kedua dengan nilai *alpha* yang telah diacak sebelumnya sebanyak *stringlen* yang telah ditetapkan.

7. Baris 48-51 adalah jika hasil proses *extended intermediate crossover* untuk *offspring* kedua > `BatasAtasRand`, maka hasil proses *extended intermediate crossover* dibuat sama dengan nilai `BatasAtasRand`.
8. Baris 52-56 adalah jika hasil proses *extended intermediate crossover* untuk *offspring* kedua < `BatasBawahRand`, maka hasil proses *extended intermediate crossover* dibuat sama dengan nilai `BatasBawahRand`.

5.2.3 Proses Mutasi

Salah satu metode mutasi yang diimplementasikan ke dalam sistem adalah *random mutation*. Hasil *offspring* yang dihasilkan dari satu kali proses mutasi adalah satu buah *offspring*. Banyaknya hasil *offspring* yang dihasilkan bergantung pada nilai *mutation rate* (*mr*) yang telah ditentukan sebelumnya. Banyaknya *parent* yang digunakan untuk proses mutasi adalah satu buah *parent* yang dipilih secara acak oleh sistem. Kemudian, melakukan proses mutasi terhadap *parent* yang telah dipilih dan nilai *r* yang telah ditentukan sebelumnya secara acak oleh fungsi `Nilai_r` dengan nilai Max_i = 7.3 dan nilai Min_j = 0. Proses *random mutation* dijelaskan pada fungsi `Random_Mutation`.

```
1  public double[][] Random_Mutation(double[][] gen_populasi,
2  double r, int popsize, int stringLen, double Mr) {
3      Random random = new Random();
4      int hasil_mutasi = (int) Math.ceil(Mr * popsize);
5      int jum_mutasi = hasil_mutasi;
6
7      double[][] offspringMutasiFinal = new
8      double[jum_mutasi][stringLen];
9      double[][] P1 = new double[1][stringLen];
10     double[][] offspringMutasi = new double[1][stringLen];
11
12     for (int i = 0; i < jum_mutasi; i++) {
13         // Menentukan Posisi Point Random secara random
14         int BatasBawahRandom = 1, BatasAtasRandom =
15         stringLen;
16         int PosisiPointRandom =
17         random.nextInt(BatasAtasRandom - BatasBawahRandom +
18         1) + BatasBawahRandom;
19
20         // Memilih 1 parent secara acak
21         BatasBawahRandom = 0;
22         BatasAtasRandom = popsize - 1;
23
24         int IndexP1 = random.nextInt(BatasAtasRandom -
25         BatasBawahRandom + 1) + BatasBawahRandom;
26
27         for (int m = 0; m < stringLen; m++) {
28             P1[0][m] = gen_populasi[IndexP1][m];
29             offspringMutasi[0][m] =
30             gen_populasi[IndexP1][m];
31         }
32
33         //Proses random mutation dilakukan berdasarkan nilai
34         r yang sudah diacak sebelumnya
35         for (int j = 0; j < stringLen; j++) {
36             if (PosisiPointRandom ==
```

```
37                  offspringMutasi[0][PosisiPointRandom - 1]) {
38                      offspringMutasi[0][PosisiPointRandom - 1] =
39                      gen_populasi[0][PosisiPointRandom - 1];
40                  } else {
41                      offspringMutasi[0][PosisiPointRandom - 1] =
42                      gen_populasi[IndexP1][PosisiPointRandom - 1]
43                      + (r * (7.3 - 0));
44                      if  (offspringMutasi[0][PosisiPointRandom -
45                      1] > BatasAtasRand[j]) {
46                          offspringMutasi[0][PosisiPointRandom -
47                          1] = BatasAtasRand[j];
48                      }                   else                if
49                      (offspringMutasi[0][PosisiPointRandom - 1] <
50                      BatasBawahRand[j]) {
51                          offspringMutasi[0][PosisiPointRandom -
52                          1] = BatasBawahRand[j];
53                      }
54                  }
55                  offspringMutasiFinal[i][j]                  =
56                  offspringMutasi[0][j];
57              }
58          }
59
60          return offspringMutasiFinal;
61      }
```

Gambar 5.7 *Source Code* Proses Mutasi

Penjelasan Gambar 5.7 *Source Code* Proses Mutasi sebagai berikut:

1. Baris 16-18 adalah proses untuk menentukan satu `PosisiPointRandom` secara acak berdasarkan batas bawah dan batas atas yang telah ditentukan.
2. Baris 24-25 adalah proses untuk memilih satu *parent* secara acak.
3. Baris 38-39 adalah proses *random mutation* dengan kondisi jika `PosisiPointRandom == offspringMutasi[0][PosisiPointRandom - 1]`, maka akan menghasilkan gen *offspring* yang sama dengan *parent*-nya.
4. Baris 41-43 adalah proses *random mutation* dengan nilai *r* yang telah diacak, nilai Max_i dan Min_j yang telah ditentukan sebelumnya dengan kondisi jika `PosisiPointRandom != offspringMutasi[0][PosisiPointRandom - 1]`, maka akan menghasilkan gen *offspring* yang telah dihitung menggunakan formula perhitungan *random mutation*.
9. Baris 46-47 adalah jika `offspringMutasi[0][PosisiPointRandom - 1] > BatasAtasRand[j]`, maka salah satu gen hasil proses *random mutation* dibuat sama dengan nilai `BatasAtasRand`.
5. Baris 51-52 adalah jika `offspringMutasi[0][PosisiPointRandom - 1] > BatasBawahRand[j]`, maka salah satu gen hasil proses *random mutation* dibuat sama dengan nilai `BatasBawahRand`.

5.2.4 Proses Evaluasi dan Seleksi

Langkah pertama yang dilakukan pada proses evaluasi dan seleksi adalah melakukan proses evaluasi yang berupa perhitungan nilai *fitness* yang dijelaskan pada fungsi `Hitung_Fitness`. Langkah kedua adalah melakukan proses seleksi

menggunakan metode *elitism selection*, yaitu proses memilih individu dari himpunan populasi dan *offspring* yang dipertahankan pada generasi berikutnya dengan mempertimbangkan nilai *fitness* setiap individu yang dijelaskan pada fungsi `Seleksi_Elitism`. Langkah ketiga adalah mengurutkan individu-individu yang memiliki nilai *fitness* dari yang tertinggi ke terendah untuk dipertahankan ke generasi selanjutnya yang dijelaskan pada fungsi `Seleksi_Elitism`.

```
1  public double[] Hitung_Fitness(double[][] populasi_gabungan,
2  double[][]    denormalisasi,    int    model_optimasi,    int
3  jum_optimalOptimal, int jum_optimalMinmal,int optimal1, int
4  optimal2, int optimal3, int minimal) {
5      double[]             fitness             =             new
6      double[populasi_gabungan.length];
7
8      for (int i = 0; i < 1; i++) {
9         for (int j = 0; j < populasi_gabungan.length; j++) {
10            //Rumus untuk menghitung fitnessnya
11            if (model_optimasi == 1) {
12                if (jum_optimalOptimal == 1) {
13                    if (optimal1 == 1) {
14                        fitness[j]                              =
15                         denormalisasi[j][(optimal1 - 1)];
16                    } else if (optimal2 == 3) {
17                        fitness[j]                              =
18                         denormalisasi[j][(optimal2 - 1)];
19                    } else {
20                        fitness[j]                              =
21                         denormalisasi[j][(optimal3 - 1)];
22                    }
23                } else if (jum_optimalOptimal == 2) {
24                    if (optimal1 == 1 && optimal2 == 3) {
25                        fitness[j]                              =
26                         denormalisasi[j][(optimal1  -  1)]  +
27                         denormalisasi[j][(optimal2 - 1)];
28                    } else if (optimal1 == 1 && optimal3 ==
29                    4) {
30                        fitness[j]                              =
31                         denormalisasi[j][(optimal1  -  1)]  +
32                         denormalisasi[j][(optimal3 - 1)];
33                    } else {
34                        fitness[j]                              =
35                         denormalisasi[j][(optimal2  -  1)]  +
36                         denormalisasi[j][(optimal3 - 1)];
37                    }
38                } else if (jum_optimalOptimal == 3) {
39                    fitness[j] = denormalisasi[j][(optimal1
40                    -  1)]  +  denormalisasi[j][(optimal2  -
41                    1)] + denormalisasi[j][(optimal3 - 1)];
42                }
43            } else {
44                if (jum_optimalMinmal == 2) {
45                    if (optimal1 == 1 && minimal == 2) {
46                        fitness[j]                              =
47                         denormalisasi[j][(optimal1  -  1)]  +
48                         (1  /  denormalisasi[j][(minimal  -
49                         1)]);
```

```
                    } else if (optimal2 == 3 && minimal ==
                    2) {
                        fitness[j]                          =
                        denormalisasi[j][(optimal2 - 1)] +
                        (1 / denormalisasi[j][(minimal -
                        1)]);
                    } else {
                        fitness[j]                          =
                        denormalisasi[j][(optimal3 - 1)] +
                        (1 / denormalisasi[j][(minimal -
                        1)]);
                    }
                } else if (jum_optimalMinmal == 3) {
                    if (optimal1 == 1 && optimal2 == 3 &&
                    minimal == 2) {
                        fitness[j]                          =
                        denormalisasi[j][(optimal1 - 1)] +
                        denormalisasi[j][(optimal2 - 1)] +
                        (1 / denormalisasi[j][(minimal -
                        1)]);
                    } else if (optimal1 == 1 && optimal3 ==
                    4 && minimal == 2) {
                        fitness[j]                          =
                        denormalisasi[j][(optimal1 - 1)] +
                        denormalisasi[j][(optimal3 - 1)] +
                        (1 / denormalisasi[j][(minimal -
                        1)]);
                    } else {
                        fitness[j]                          =
                        denormalisasi[j][(optimal2 - 1)] +
                        denormalisasi[j][(optimal3 - 1)] +
                        (1 / denormalisasi[j][(minimal -
                        1)]);
                    }
                } else {
                    fitness[j] = denormalisasi[j][(optimal1
                    - 1)] + denormalisasi[j][(optimal2 -
                    1)] + denormalisasi[j][(optimal3 - 1)]
                    + (1 / denormalisasi[j][(minimal -
                    1)]);
                }
            }
        }
    }

    return fitness;
}

public          double[][]          Seleksi_Elitism(double[][]
populasi_gabungan, double[] fitness, int popsize) {
    int stringLen_Chromosome = populasi_gabungan[0].length;
    int popsize_SemuaIndividu = populasi_gabungan.length;

    double[][]          Hasil          =          new
    double[popsize_SemuaIndividu][stringLen_Chromosome];
    int[]       IndexAwalIndividuGabungan       =       new
    int[popsize_SemuaIndividu];

    for (int i = 0; i < popsize_SemuaIndividu; i++) {
```

```
109             IndexAwalIndividuGabungan[i] = i;
110         }
111 
112         //Proses penghitungan fitness
113         double[]          Nilai_Fitness          =          new
114         double[popsize_SemuaIndividu];
115 
116         for (int i = 0; i < popsize_SemuaIndividu; i++) {
117             Nilai_Fitness[i] = fitness[i];
118         }
119 
120         //Sorting nilai fitness dari terbesar ke terkecil
121         for (int i = 0; i < popsize_SemuaIndividu; i++) {
122             double Temp_Fitness = Nilai_Fitness[i];
123             int          TempIndexAwalIndividuGabungan          =
124             IndexAwalIndividuGabungan[i];
125             double[][]          Temp_Hasil          =          new
126             double[1][stringLen_Chromosome];
127             for (int j = (i + 1); j < popsize_SemuaIndividu;
128             j++) {
129                 if (Nilai_Fitness[j] > Temp_Fitness) {
130                     Temp_Fitness = Nilai_Fitness[j];
131                     Nilai_Fitness[j] = Nilai_Fitness[i];
132                     Nilai_Fitness[i] = Temp_Fitness;
133                     TempIndexAwalIndividuGabungan                =
134                     IndexAwalIndividuGabungan[j];
135                     IndexAwalIndividuGabungan[j]                 =
136                     IndexAwalIndividuGabungan[i];
137                     IndexAwalIndividuGabungan[i]                 =
138                     TempIndexAwalIndividuGabungan;
139                     for (int k = 0; k < stringLen_Chromosome;
140                     k++) {
141                         Temp_Hasil[0][k]                         =
142                         populasi_gabungan[j][k];
143                     }
144 
145                     for (int k = 0; k < stringLen_Chromosome;
146                     k++) {
147                         populasi_gabungan[j][k]                  =
148                         populasi_gabungan[i][k];
149                     }
150 
151                     for (int k = 0; k < stringLen_Chromosome;
152                     k++) {
153                         populasi_gabungan[i][k]                  =
154                         Temp_Hasil[0][k];
155                     }
156                 }
157             }
158         }
159 
160         index_terbaik = IndexAwalIndividuGabungan[0];
161 
162         //Menampung    Hasil    populasi_gabungan    ter-sorting
163         sebanyak jumlah populasi gabungan
164         for (int i = 0; i < popsize_SemuaIndividu; i++) {
165             for (int j = 0; j < stringLen_Chromosome; j++) {
166                 Hasil[i][j] = populasi_gabungan[i][j];
167             }
```

```
168         }
169
170         return Hasil;
171     }
```

Gambar 5.8 *Source Code* Proses Evaluasi dan Seleksi

Penjelasan Gambar 5.8 *Source Code* Proses Evaluasi dan Seleksi sebagai berikut:

1. Baris 13-22 adalah proses menghitung nilai *fitness* dengan kondisi jika model optimasi yang digunakan adalah optimal-optimal dan kandungan gizi susu yang ingin dioptimalkan berjumlah satu, baik itu protein atau laktosa atau *density*.
2. Baris 25-27 adalah proses menghitung nilai *fitness* dengan kondisi jika model optimasi yang digunakan adalah optimal-optimal dan kandungan gizi susu yang ingin dioptimalkan berjumlah dua, yaitu protein dan laktosa.
3. Baris 30-32 adalah proses menghitung nilai *fitness* dengan kondisi jika model optimasi yang digunakan adalah optimal-optimal dan kandungan gizi susu yang ingin dioptimalkan berjumlah dua, yaitu protein dan *density*.
4. Baris 34-36 adalah proses menghitung nilai *fitness* dengan kondisi jika model optimasi yang digunakan adalah optimal-optimal dan kandungan gizi susu yang ingin dioptimalkan berjumlah dua, yaitu laktosa dan *density*.
5. Baris 39-41 adalah proses menghitung nilai *fitness* dengan kondisi jika model optimasi yang digunakan adalah optimal-optimal dan kandungan gizi susu yang ingin dioptimalkan berjumlah tiga, yaitu protein, laktosa dan *density*.
6. Baris 46-49 adalah proses menghitung nilai *fitness* dengan kondisi jika model optimasi yang digunakan adalah optimal-minimal dan kandungan gizi susu yang ingin dioptimalkan dan diminimalkan berjumlah dua, yaitu protein dan lemak.
7. Baris 52-55 adalah proses menghitung nilai *fitness* dengan kondisi jika model optimasi yang digunakan adalah optimal-minimal dan kandungan gizi susu yang ingin dioptimalkan dan diminimalkan berjumlah dua, yaitu laktosa dan lemak.
8. Baris 57-60 adalah proses menghitung nilai *fitness* dengan kondisi jika model optimasi yang digunakan adalah optimal-minimal dan kandungan gizi susu yang ingin dioptimalkan dan diminimalkan berjumlah dua, yaitu *density* dan lemak.
9. Baris 65-69 adalah proses menghitung nilai *fitness* dengan kondisi jika model optimasi yang digunakan adalah optimal-minimal dan kandungan gizi susu yang ingin dioptimalkan dan diminimalkan berjumlah tiga, yaitu protein, laktosa dan lemak.
10. Baris 72-76 adalah proses menghitung nilai *fitness* dengan kondisi jika model optimasi yang digunakan adalah optimal-minimal dan kandungan gizi susu yang ingin dioptimalkan dan diminimalkan berjumlah tiga, yaitu protein, *density* dan lemak.

11. Baris 78-82 adalah proses menghitung nilai *fitness* dengan kondisi jika model optimasi yang digunakan adalah optimal-minimal dan kandungan gizi susu yang ingin dioptimalkan dan diminimalkan berjumlah tiga, yaitu laktosa, *density* dan lemak.
12. Baris 85-89 adalah proses menghitung nilai *fitness* dengan kondisi jika model optimasi yang digunakan adalah optimal-minimal dan kandungan gizi susu yang ingin dioptimalkan dan diminimalkan berjumlah empat, yaitu protein, laktosa, *density* dan lemak.
13. Baris 117 adalah proses untuk menampilkan nilai *fitness* dari fungsi `Hitung_Fitness`.
14. Baris 129-156 adalah proses untuk *sorting* nilai *fitness* setiap individu dari nilai *fitness* yang tertinggi ke terendah
15. Baris 160 adalah proses untuk menyimpan *index* terbaik yang memiliki nilai *fitness* yang tertinggi.
16. Baris 166 adalah proses untuk menampung `Hasil` dari `populasi_gabungan` yang telah diurutkan berdasarkan nilai *fitness* tertinggi ke terendah sebanyak jumlah populasi gabungan.

5.2.5 Proses *Random Injection*

Setelah proses evaluasi dan seleksi telah dilakukan oleh sistem, maka langkah selanjutnya adalah melakukan proses *random injection* dengan tujuan untuk menghindari konvergensi dini. Cara kerja dari *random injection* ini adalah mencocokan nilai *fitness* pertama dengan kedua. Jika nilai *fitness* pertama dan kedua sama, maka akan dilakukan proses *random injection*.

```
1  public double[][] Random_Injection(double[]
2  tempHasilFitnessPerIterasi, double[][] gen_populasi, double[]
3  batas_atas, double[] batas_bawah, int stringLen) {
4      int jum_individu = tempHasilFitnessPerIterasi.length;
5      double[][] individu_baru = new
6      double[jum_individu][stringLen];
7
8      for (int i = 0; i < jum_individu; i++) {
9          for (int j = 0; j < stringLen; j++) {
10             if (i != jum_individu - 1) {
11                 if (tempHasilFitnessPerIterasi[i] ==
12                 tempHasilFitnessPerIterasi[(i + 1)]) {
13                     individu_baru[i][j] = (batas_bawah[j] +
14                      (Math.random() * (batas_atas[j] -
15                      batas_bawah[j])));
16                 } else {
17                     individu_baru[i][j] =
18                      gen_populasi[i][j];
19                 }
20             } else {
21                 individu_baru[i][j] = gen_populasi[i][j];
22             }
23         }
24     }
```

```
25
26          return individu_baru;
27  }
```

Gambar 5.9 *Source Code* Proses *Random Injection*

Penjelasan Gambar 5.9 *Source Code* Proses *Random Injection* sebagai berikut:

1. Baris 13-15 adalah melakukan proses *random injection* dengan ketentuan jika `i != jum_individu - 1` dan jika `tempHasilFitnessPerIterasi[i] == tempHasilFitnessPerIterasi[(i + 1)]`.
2. Baris 17-18 adalah sistem tidak melakukan proses *random injection* dikarenakan tidak memenuhi ketentuan yang telah disediakan pada percabangan tersebut, yaitu `tempHasilFitnessPerIterasi[i] != tempHasilFitnessPerIterasi[(i + 1)]`.
3. Baris 21 adalah sistem tidak melakukan proses *random injection* dikarenakan merupakan `jum_individu` yang terakhir.

5.2.6 Proses Pemilihan Individu yang Lolos ke Generasi Selanjutnya

Langkah selanjutnya adalah proses pemilihan individu yang lolos ke generasi selanjutnya. Banyaknya jumlah individu yang lolos ke generasi selanjutnya ditetapkan berdasarkan nilai *popsize* yang telah ditentukan sebelumnya.

```
1   public double[][] Next_Generation(double[][] Hasil, int
2   popsize, int stringLen) {
3       double[][] gen_populasiBaru = new
4       double[popsize][stringLen];
5
6       for (int i = 0; i < popsize; i++) {
7           for (int j = 0; j < stringLen; j++) {
8               gen_populasiBaru[i][j] = Hasil[i][j];
9           }
10      }
11
12      return gen_populasiBaru;
13  }
```

Gambar 5.10 *Source Code* Proses Individu yang Lolos ke Generasi Selanjutnya

Penjelasan Gambar 5.10 *Source Code* Proses Individu yang Lolos ke Generasi Selanjutnya sebagai berikut:

1. Baris 8 adalah proses untuk menyimpan hasil pemilihan individu yang lolos ke generasi selanjutnya pada variabel yang bernama `gen_populasiBaru` sebanyak *popsize* yang telah ditentukan sebelumnya.

5.2.7 Proses Pemilihan Individu Terbaik

Proses pemilihan individu terbaik merupakan proses untuk memilih satu individu yang memiliki nilai *fitness* yang tertinggi dan dijadikan sebagai solusi untuk memecahkan masalah yang telah diangkat sebelumnya.

```
1   public double Individu_Terbaik(double[][] Hasil, int
2   stringLen) {
3       double[][] individu_terbaik = new double[1][4];
```

```
4
5       for (int i = 0; i < stringLen; i++) {
6           individu_terbaik[0][i] = Hasil[0][i];
7       }
8
9       return individu_terbaik;
10  }
```

Gambar 5.11 *Source Code* Proses Pemilihan Individu Terbaik

Penjelasan Gambar 5.10 *Source Code* Proses Pemilihan Individu Terbaik sebagai berikut:

1. Baris 6 adalah proses untuk menyimpan hasil pemilihan individu terbaik pada variabel yang bernama `individu_terbaik`.

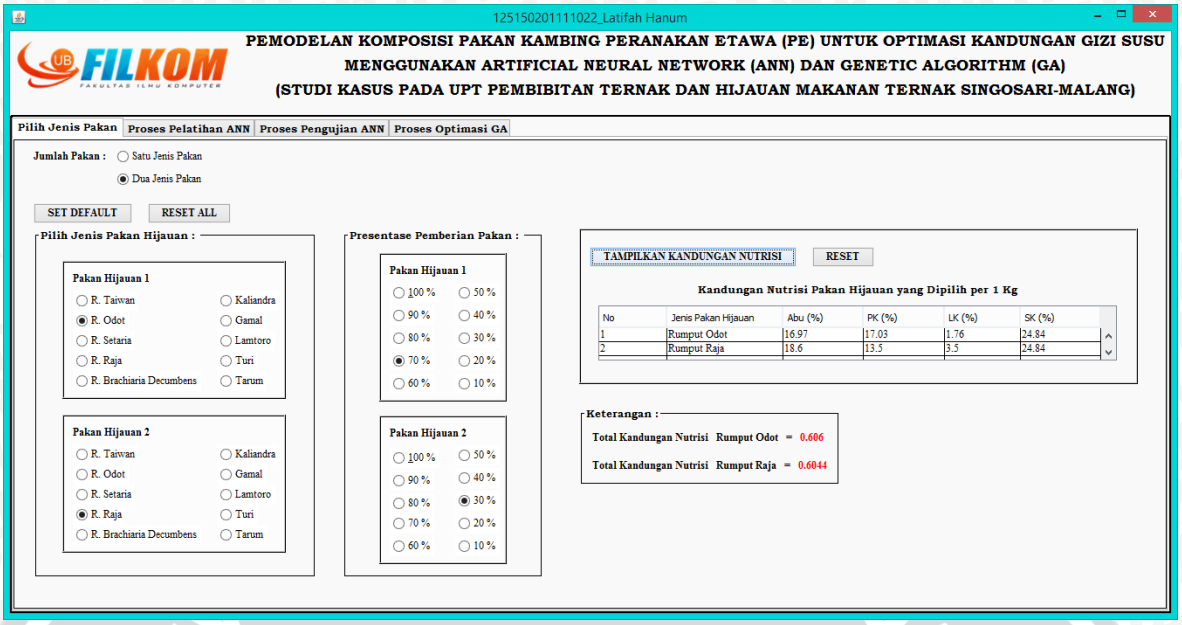
5.3 Implementasi Antarmuka Sistem

Implementasi antarmuka ini terdiri dari beberapa bagian, yaitu proses pemilihan jenis pakan hijauan yang terdiri dari pemilihan jumlah pakan yang digunakan beserta presentase pemberian pakannya, proses pelatihan ANN, proses pengujian ANN dan proses optimasi GA. Berikut adalah hasil implementasi program yang telah dibuat.

5.3.1 Antarmuka Hasil Pemilihan Jenis Pakan Hijauan

Pada antarmuka ini akan ditampilkan hasil dari jenis pakan hijauan yang digunakan beserta dengan kandungan nutrisi dari setiap jenis pakan hijauan yang telah dipilih sebelumnya. Terdapat 10 jenis pakan hijauan yang dapat dipilih oleh pengguna, yaitu rumput Taiwan, rumput odot, rumput setaria, rumput raja, rumput *brachiaria decumbens*, kaliandra, gamal, lamtoro, turi dan tarum. Sedangkan, presentase pemberian pakan hijauan berkisar antara 10% sampai 100% dengan rentang nilai sebesar 10%.

Pengguna akan melakukan beberapa aksi berupa memilih jumlah pakan yang digunakan, memilih jenis pakan hijauan sebanyak jumlah pakan yang dipilih serta presentase pemberian pakan setiap jenis pakan hijauan yang telah dipilih sebelumnya. Jumlah pakan yang digunakan terdiri dari satu jenis pakan hijauan dan dua jenis pakan hijauan. Jenis pakan hijauan dan presentase pemberian pakan yang diberikan tergantung pada jumlah pakan hijauan yang digunakan. Tombol TAMPILKAN KANDUNGAN NUTRISI pada program digunakan untuk menampilkan nama pakan yang telah dipilih berdasarkan jumlah pakan yang telah dipilih sebelumnya beserta kandungan nutrisi yang tedapat pada setiap pakan tersebut yang berupa abu, protein kasar (PK), lemak kasar (LK) dan serat kasar (SK) yang akan ditampilkan di dalam tabel yang sudah tersedia. Program juga akan menampilkan total kandungan nutrisi setiap pakan per satu kilogramnya. Tombol *SET DEFAULT* digunakan untuk menampilkan jumlah pakan hijauan, jenis pakan hijauan dan presentase pemberian pakan hijauan berdasarkan jenis pakan hijauan yang sering digunakan pada UPT Pembibitan Ternak dan Hijauan Makanan Ternak Singosari-Malang. Berikut adalah Gambar 5.12 Antarmuka Hasil Pemilihan Jenis Pakan Hijauan.

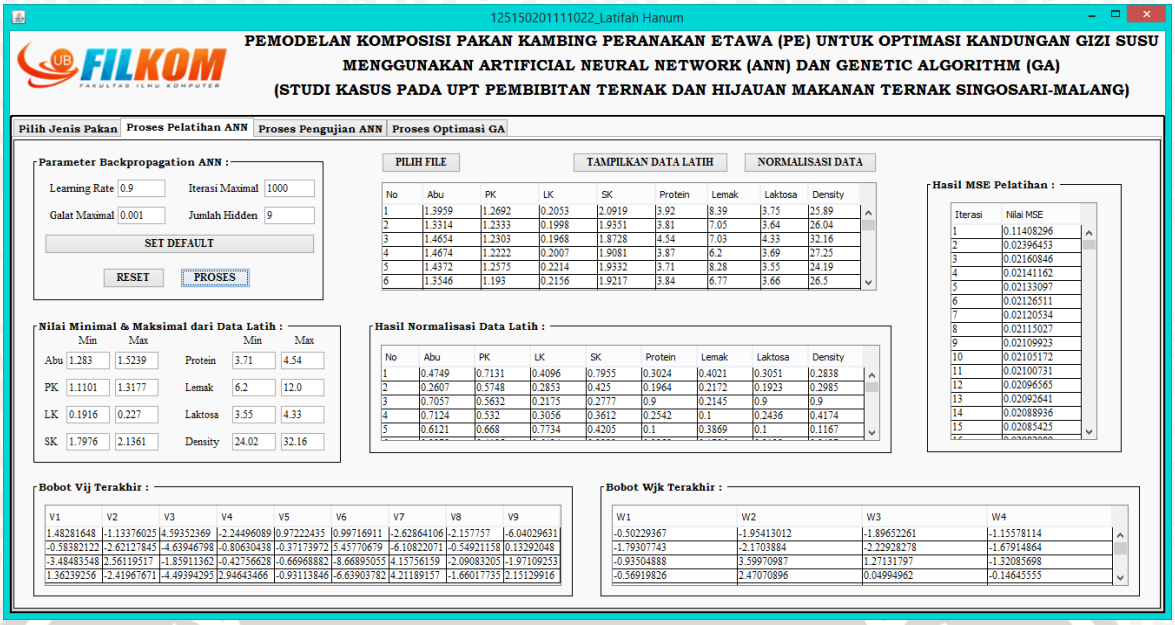


Gambar 5.12 Antarmuka Hasil Pemilihan Jenis Pakan Hijauan

5.3.2 Antarmuka Hasil Pelatihan ANN

Pada antarmuka ini akan ditampilkan hasil dari proses pelatihan ANN. Tombol *SET DEFAULT* digunakan untuk mengatur nilai parameter ANN secara otomatis agar sesuai dengan hasil pengujian yang telah dilakukan. Tombol PILIH FILE digunakan untuk memilih data latih yang ingin digunakan pada proses pelatihan ANN. Tombol TAMPILKAN DATA LATIH digunakan untuk menampilkan data latih yang berisikan nilai dari kandungan nutrisi bahan pakan yang berupa abu, protein kasar (PK), lemak kasar (LK) dan serat kasar (SK) serta nilai dari kandungan gizi susu kambing PE yang didapatkan dari hasil uji laboratorium yang berupa protein, lemak, laktosa dan *density*.

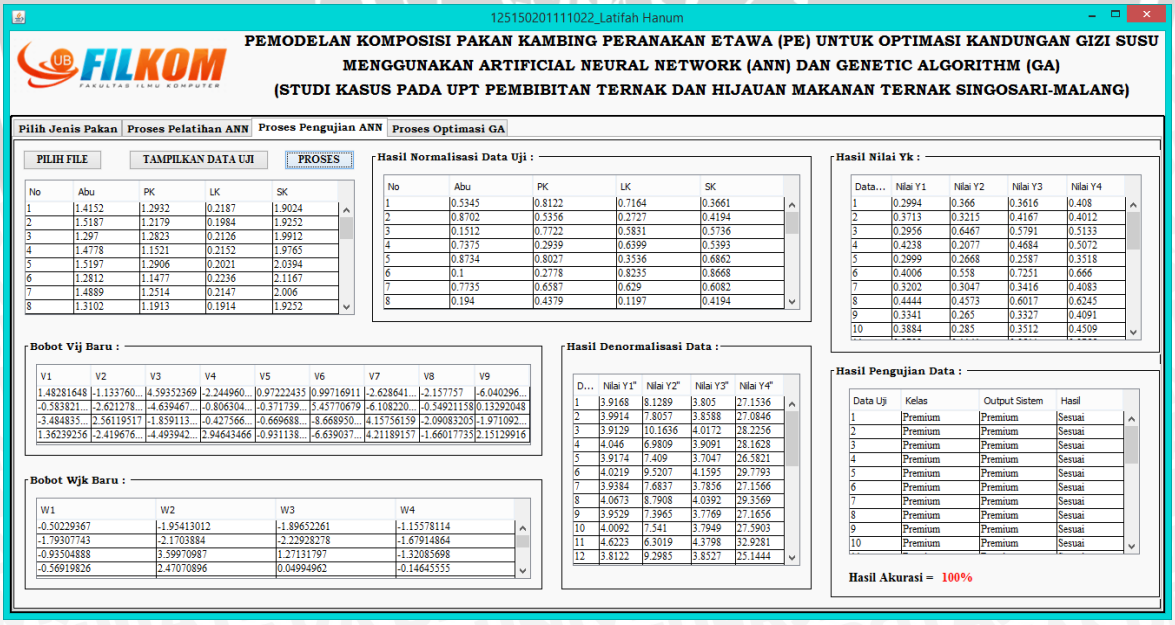
Tombol NORMALISASI DATA digunakan untuk menampilkan hasil dari proses normalisasi data latih dan hasil dari proses tersebut akan ditampilkan pada tabel yang sudah tersedia beserta hasil dari nilai minimal dan maksimal dari setiap kandungan nutrisi bahan pakan dan kandungan gizi susu kambing PE sehingga sistem akan menampilkan nilai minimal dan maksimal dari abu, PK, LK, SK, protein, lemak, laktosa dan *density.* Nilai minimal dan maksimal tersebut akan digunakan pada proses pelatihan ANN, khusunya pada proses normalisasi data. Tombol PROSES digunakan untuk melakukan proses perhitungan pelatihan ANN. Hasil dari proses perhitungan pelatihan ANN akan ditampilkan pada tabel yang telah disediakan sebelumnya, yaitu sistem akan menampilkan hasil MSE pelatihan sebanyak jumlah iterasi yang dimasukan, bobot V_{ij} terakhir dan bobot W_{jk} terakhir. Bobot V_{ij} terakhir dan bobot W_{jk} terakhir yang diperoleh pada proses pelatihan ANN akan digunakan pada proses pengujian ANN dan digunakan sebagai pengetahuan dasar untuk melakukan pengenalan pola yang baru. Bobot V_{ij} dan bobot W_{jk} pada proses pengujian ANN digunakan pada tahapan *feedforward* saja. Berikut adalah Gambar 5.13 Antarmuka Hasil Pelatihan ANN.



Gambar 5.13 Antarmuka Hasil Pelatihan ANN

5.3.3 Antarmuka Hasil Pengujian ANN

Pada antarmuka ini akan ditampilkan hasil dari proses pengujian ANN. Tombol PILIH FILE digunakan untuk memilih data uji. Tombol TAMPILKAN DATA UJI digunakan untuk menampilkan data uji dan normalisasi data uji yang hasilnya ditampilkan pada tabel yang sudah disediakan. Berikut adalah Gambar 5.14 Antarmuka Hasil Pengujian ANN.



Gambar 5.14 Antarmuka Hasil Pengujian ANN

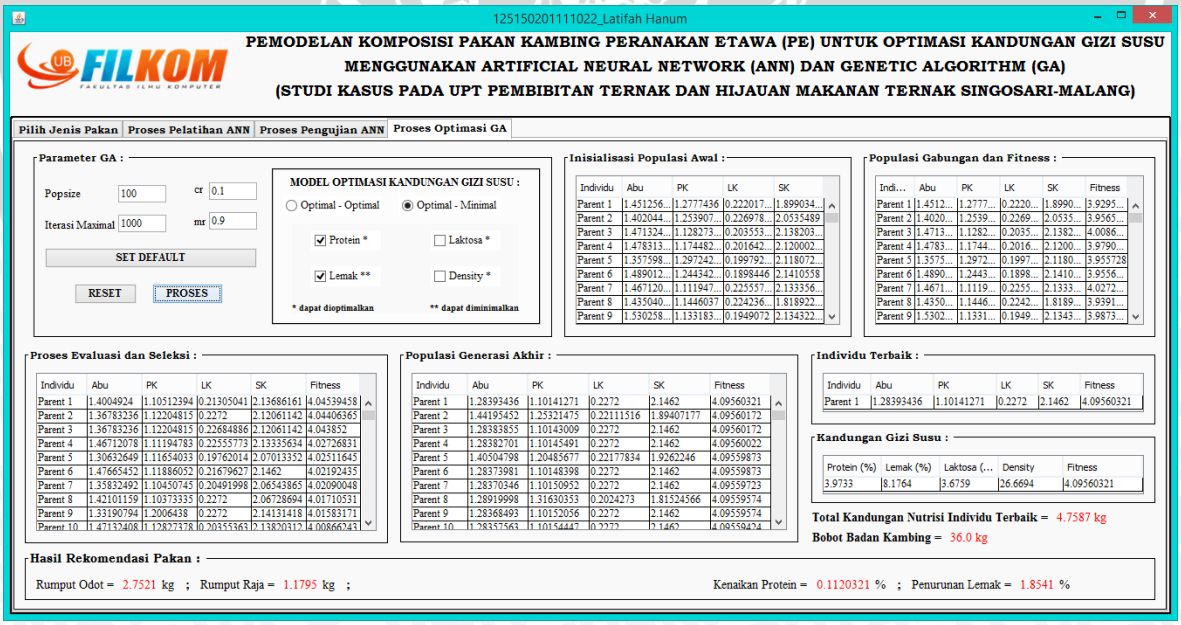
Tombol PROSES pada sistem digunakan untuk melakukan proses perhitungan pengujian ANN dan hasilnya akan ditampilkan pada tabel yang sudah disediakan dengan menampilkan bobot V_{ij} dan W_{jk} terakhir yang diperoleh dari proses pelatihan ANN, kemudian digunakan pada proses pengujian ANN sebagai bobot

V_{ij} dan W_{jk} yang baru. Program tersebut juga menampilkan hasil nilai Y_k, denormalisasi data dan pengujian data yang diperoleh sistem.

5.3.4 Antarmuka Hasil Optimasi GA

Pada antarmuka ini akan ditampilkan hasil dari proses optimasi menggunakan GA. Tombol *SET DEFAULT* digunakan untuk mengatur nilai parameter GA secara otomatis agar sesuai dengan hasil pengujian yang telah dilakukan. Sebelum pengguna menekan tombol PROSES, pengguna harus melakukan pemilihan terhadap model optimasi yang ingin digunakan, yaitu optimal dan optimal atau optimal dan minimal. Kemudian, pengguna dapat memilih kandungan gizi susu yang tersedia sesuai dengan model optimasi yang telah dipilih sebelumnya. Tombol PROSES ini digunakan untuk melakukan proses perhitungan optimasi GA dan hasilnya akan ditampilkan pada tabel yang sudah disediakan hingga diperoleh hasil rekomendasi pakannya.

Proses GA yang ditampilkan pada tabel yang terdapat di sistem adalah proses inisialisasi populasi awal pada iterasi pertama, populasi gabungan dan *fitness* yang diperoleh pada iterasi pertama, proses evaluasi dan seleksi pada iterasi pertama, populasi generasi akhir, individu terbaik dan kandungan gizi susu yang dihasilkan dari inidividu terbaik. Pada antarmuka ini, sistem juga akan menampilkan hasil rekomendasi pakan, bobot badan kambing PE yang digunakan serta hasil presentase kenaikan kandungan gizi susu ataupun penurunan kandungan gizi susu berdasarkan model optimasi yang dipilih sebelumnya. Berikut adalah Gambar 5.15 Antarmuka Hasil Optimasi GA.



Gambar 5.15 Antarmuka Hasil Optimasi GA

BAB 6 PENGUJIAN DAN ANALISIS

Pada bab ini akan menjelaskan hasil pengujian dan analisis dari sistem Pemodelan Komposisi Pakan Kambing Peranakan Etawa (PE) untuk Optimasi Kandungan Gizi Susu Menggunakan *Artificial Neural Network* (ANN) dan *Genetic Algorithm* (GA) yang telah diimplementasikan pada tahap sebelumnya. Proses pengujian dilakukan berdasarkan skenario yang telah dijelaskan pada Bab 4 sub bab 4.5, kemudian menganalisis hasil pengujian yang didapatkan.

6.1 Pengujian

Pada sub bab ini akan dijelaskan mengenai macam-macam pengujian yang akan dilakukan pada sistem. Pengujian tersebut dilakukan berdasarkan skenario pengujian yang telah dirancang pada Bab 4 sub bab 4.5 dan yang dilakukan secara sekuensial. Macam-macam pengujian tersebut sebagai berikut:

1. Pengujian terhadap jumlah iterasi
2. Pengujian terhadap jumlah *hidden layer*
3. Pengujian terhadap nilai *learning rate*
4. Pengujian terhadap ukuran populasi
5. Pengujian terhadap jumlah generasi
6. Pengujian terhadap kombinasi *crossover rate* dan *mutation rate*
7. Pengujian terhadap kombinasi variasi *output* yang dioptimalkan

6.2 Hasil Pengujian dan Analisis

Terdapat tujuh hasil pengujian dan analisis sesuai dengan skenario pengujian yang telah dibuat sebelumnya. Permasalahan studi kasus yang digunakan pada proses pengujian semua skenario ini adalah jumlah pakan yang dipakai ada 2, yaitu rumput odot dan rumput raja, presentase pemberian pakan = 70%:30%, BB kambing = 36 kg, model optimasi yang dipilih adalah optimal-minimal dengan kandungan gizi susu yang dioptimalkan adalah protein dan diminimalkan adalah lemak, data latih dan data uji yang digunakan sebesar 30 data yang sama untuk kualitas susu Premium dan 15 data yang sama untuk kualitas susu Baik, jumlah *hidden layer* = 4, nilai *learning rate* = 0.64, galat maksimal (epsilon) = 0.001, jumlah generasi = 250, *cr* = 0.5 dan *mr* = 0.1.

Hasil nilai pengujian terbaik yang diperoleh dari setiap poin pengujian akan digunakan pada pengujian selanjutnya secara sekuensial. Berikut adalah hasil dari tujuh pengujian yang dilakukan sebagai berikut:

6.2.1 Hasil Pengujian Terhadap Jumlah Iterasi

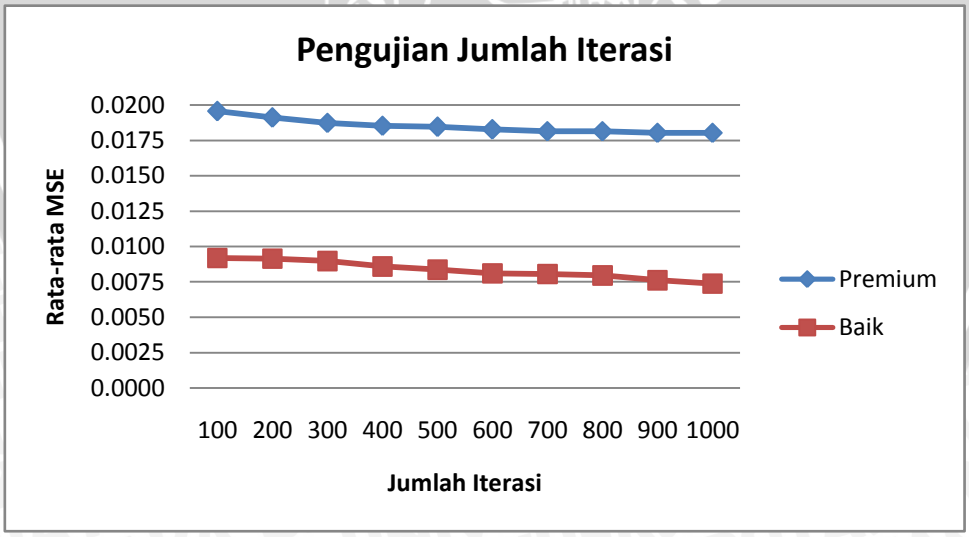
Pengujian jumlah iterasi adalah banyaknya jumlah pengulangan yang dilakukan untuk mendapatkan bobot terbaik sesuai dengan jumlah iterasi maksimal yang dimasukan. Pengujian ini dilakukan untuk mendapatkan jumlah

iterasi dengan nilai MSE yang terendah. Pengujian ini dilakukan sebanyak 10 kali percobaan untuk masing-masing variasi jumlah iterasi dengan *range* antara 100-1000. Data yang digunakan adalah 30 data latih dan 30 data uji untuk kualitas susu Premium dan 15 data latih dan 15 data uji untuk kualitas susu Baik, jumlah *hidden layer* = 4, nilai *learning rate* = 0.64 dan galat maksimal = 0.001. Hasil pengujian yang didapatkan pada variasi jumlah iterasi diperlihatkan pada Tabel 6.1 Hasil Pengujian Terhadap Jumlah Iterasi dan hasil pengujian secara lengkap untuk kualitas susu Premium dan Baik dapat dilihat pada Lampiran E dan F.

Tabel 6.1 Hasil Pengujian Terhadap Jumlah Iterasi

Jumlah Iterasi	Nilai MSE Percobaan ke-*i*										Rata-rata MSE
	1	2	3	4	5	6	7	8	9	10	
100	0.019376	0.019813	...	...	...	...	...	...	...	0.019466	0.019562
200	0.018857	0.019163	...	...	...	...	...	...	...	0.019089	0.019111
300	0.018518	0.018789	...	...	...	...	...	...	...	0.018886	0.018734
400	0.018365	0.018106	...	...	...	...	...	...	...	0.019327	0.018538
500	0.017918	0.018573	...	...	...	...	...	...	...	0.018724	0.018455
600	0.017813	0.018172	...	...	...	...	...	...	...	0.018498	0.018271
700	0.017989	0.017755	...	...	...	...	...	...	...	0.018666	0.018147
800	0.018032	0.017926	...	...	...	...	...	...	...	0.017703	0.018144
900	0.018214	0.017796	...	...	...	...	...	...	...	0.018307	0.018038
1000	0.017644	0.017848	...	...	...	...	...	...	...	0.018278	0.018028

Berdasarkan Tabel 6.1, maka dibuat grafik hasil pengujian seperti yang diperlihatkan pada Gambar 6.1 Grafik Hasil Pengujian Terhadap Jumlah Iterasi.



Gambar 6.1 Grafik Hasil Pengujian Terhadap Jumlah Iterasi

Jumlah iterasi merupakan salah satu parameter ANN yang mempengaruhi tingkat prediksi algoritma ANN. Dalam beberapa kasus, ada kalanya semakin besar jumlah iterasi yang digunakan, maka nilai MSE yang dihasilkan akan semakin kecil dan ada kalanya semakin besar jumlah iterasi yang digunakan,

maka nilai MSE yang dihasilkan semakin besar. Berdasarkan Gambar 6.1 Grafik Hasil Pengujian Terhadap Jumlah Iterasi, didapatkan bahwa semakin besar jumlah iterasi yang digunakan, maka nilai MSE yang didapatkan untuk kualitas susu yang Premium dan Baik juga semakin kecil. Hal tersebut membuktikan bahwa semakin besar jumlah iterasi yang digunakan, maka bobot yang dihasilkan juga akan semakin baik dan akurat. Jumlah iterasi terbaik dengan nilai rata-rata MSE terendah untuk kualitas susu Premium dan Baik memiliki jumlah iterasi terbaik yang sama, yaitu iterasi ke-1000. Jumlah iterasi terbaik untuk kualitas susu Premium diperoleh pada jumlah iterasi yang ke-1000 dengan rata-rata MSE sebesar 0.018028, sedangkan jumlah iterasi terbaik untuk kualitas susu Baik diperoleh pada jumlah iterasi yang ke-1000 dengan rata-rata MSE sebesar 0.007390.

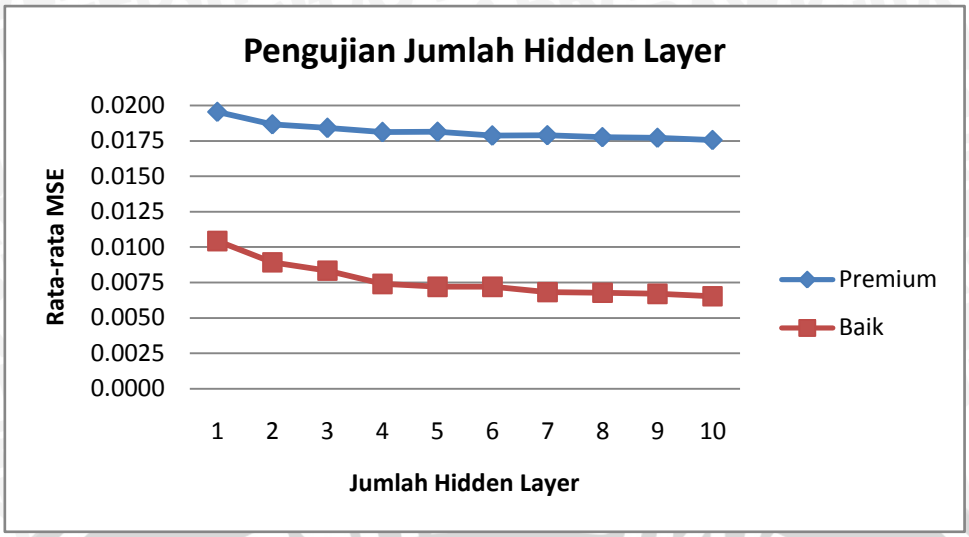
6.2.2 Hasil Pengujian Terhadap Jumlah *Hidden Layer*

Pengujian terhadap jumlah *hidden layer* dilakukan untuk mendapatkan banyaknya nilai bobot, sehingga sistem semakin teliti. Pengujian ini dilakukan sebanyak 10 kali untuk masing-masing variasi jumlah *hidden layer* dengan *range* variasi jumlah *hidden layer* yang digunakan adalah 1-10. Data yang digunakan adalah 30 data latih dan 30 data uji untuk kualitas susu Premium dan 15 data latih dan 15 data uji untuk kualitas susu Baik, jumlah iterasi = 1000 berdasarkan hasil dari pengujian jumlah iterasi, nilai *learning rate* = 0.64 dan galat maksimal = 0.001. Hasil pengujian yang didapatkan pada variasi jumlah *hidden layer* diperlihatkan pada Tabel 6.2 Hasil Pengujian Terhadap Jumlah *Hidden Layer* dan hasil pengujian secara lengkap untuk kualitas susu Premium dan Baik dapat dilihat pada Lampiran E dan F.

Tabel 6.2 Hasil Pengujian Terhadap Jumlah *Hidden Layer*

Hidden Layer	Nilai MSE Percobaan ke-*i*										Rata-rata MSE
	1	2	3	4	5	6	7	8	9	10	
1	0.019544	0.019540	...	...	...	...	...	...	...	0.019561	0.019545
2	0.019672	0.018961	...	...	...	...	...	...	...	0.017935	0.018664
3	0.018426	0.018836	...	...	...	...	...	...	...	0.018398	0.018407
4	0.017917	0.017629	...	...	...	...	...	...	...	0.018757	0.018113
5	0.018307	0.017975	...	...	...	...	...	...	...	0.017923	0.018156
6	0.017986	0.017851	...	...	...	...	...	...	...	0.018257	0.017869
7	0.017412	0.017923	...	...	...	...	...	...	...	0.017967	0.017896
8	0.017696	0.017348	...	...	...	...	...	...	...	0.017963	0.017766
9	0.018158	0.017507	...	...	...	...	...	...	...	0.017386	0.017723
10	0.017731	0.017567	...	...	...	...	...	...	...	0.017549	0.017563

Berdasarkan Tabel 6.2 Hasil Pengujian Terhadap Jumlah *Hidden Layer*, maka dibuat grafik hasil pengujian seperti yang diperlihatkan pada Gambar 6.2 Grafik Hasil Pengujian Terhadap Jumlah *Hidden Layer* sebagai berikut.



Gambar 6.2 Grafik Hasil Pengujian Terhadap Jumlah *Hidden Layer*

Banyaknya jumlah *hidden layer* yang digunakan mempengaruhi sistem dalam menghasilkan perubahan bobot yang semakin kecil. Berdasarkan Gambar 6.2 Grafik Hasil Pengujian Terhadap Jumlah *Hidden Layer*, didapatkan bahwa semakin besar jumlah *hidden layer* yang digunakan, maka nilai MSE yang didapatkan untuk kualitas susu yang Premium dan Baik juga semakin kecil dan semakin stabil. Jumlah *hidden layer* terbaik dengan nilai rata-rata MSE terendah untuk kualitas susu Premium dan Baik memiliki jumlah *hidden layer* terbaik yang sama, yaitu 10 *hidden layer*. Rata-rata MSE yang dihasilkan oleh jumlah *hidden layer* terbaik untuk kualitas susu Premium didapatkan sebesar 0.017563, sedangkan rata-rata MSE yang dihasilkan oleh jumlah *hidden layer* terbaik untuk kualitas susu Baik didapatkan sebesar 0.006536.

6.2.3 Hasil Pengujian Terhadap Variasi Nilai *Learning Rate*

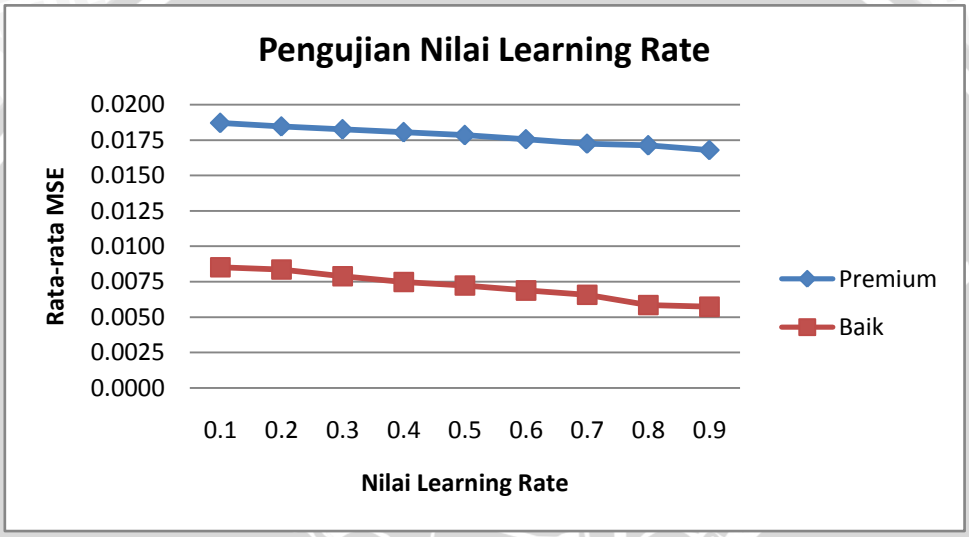
Pengujian terhadap variasi nilai *learning rate* dilakukan untuk mendapatkan nilai *learning rate* dengan MSE terendah. Pengujian ini dilakukan sebanyak 10 kali untuk masing-masing variasi nilai *learning rate* dengan r*ange* variasi nilai *learning rate* yang digunakan adalah 0.1-0.9. Data yang digunakan adalah 30 data latih dan 30 data uji untuk kualitas susu Premium dan 15 data latih dan 15 data uji untuk kualitas susu Baik, jumlah iterasi = 1000 berdasarkan hasil dari pengujian jumlah iterasi, jumlah *hidden layer* = 10 berdasarkan hasil dari pengujian jumlah *hidden layer* dan galat maksimal = 0.001. Hasil pengujian yang didapatkan pada variasi nilai *learning rate* diperlihatkan pada Tabel 6.3 Hasil Pengujian Terhadap Nilai *Learning Rate* dan hasil pengujian secara lengkap untuk kualitas susu Premium dan Baik dapat dilihat pada Lampiran E dan F.

Tabel 6.3 Hasil Pengujian Terhadap Nilai *Learning Rate*

Learning Rate	Nilai MSE Percobaan ke-*i*										Rata-rata MSE
	1	2	3	4	5	6	7	8	9	10	
0.9	0.017329	0.016435	...	...	...	...	...	...	...	0.017189	0.016794
0.8	0.017384	0.016642	...	...	...	...	...	...	...	0.017277	0.017124

0.7	0.017498	0.017452	...	...	...	...	...	...	...	0.016143	0.017250
0.6	0.017582	0.017403	...	...	...	...	...	...	...	0.017277	0.017560
0.5	0.017964	0.017637	...	...	...	...	...	...	...	0.017941	0.017845
0.4	0.018370	0.018120	...	...	...	...	...	...	...	0.017975	0.018045
0.3	0.018042	0.018205	...	...	...	...	...	...	...	0.018222	0.018252
0.2	0.018695	0.018235	...	...	...	...	...	...	...	0.018700	0.018466
0.1	0.018766	0.018675	...	...	...	...	...	...	...	0.018598	0.018706

Berdasarkan Tabel 6.3 Hasil Pengujian Terhadap Nilai *Learning Rate*, maka dibuat grafik hasil pengujian seperti yang diperlihatkan pada Gambar 6.3 Grafik Hasil Pengujian Terhadap Nilai *Learning Rate* sebagai berikut.



Gambar 6.3 Grafik Hasil Pengujian Terhadap Nilai *Learning Rate*

Berdasarkan Gambar 6.3 Grafik Hasil Pengujian Terhadap Nilai *Learning Rate*, didapatkan bahwa semakin besar nilai *learning rate* yang digunakan, maka nilai MSE yang didapatkan untuk kualitas susu yang Premium dan Baik juga semakin kecil. Sedangkan, jika semakin kecil nilai *learning rate* yang digunakan, maka nilai MSE yang didapatkan untuk kualitas susu yang Premium dan Baik akan semakin besar. Hal tersebut membuktikan bahwa nilai *learning rate* memiliki pengaruh yang besar dalam proses pelatihan ANN. Nilai *learning rate* terbaik dengan nilai rata-rata MSE terendah untuk kualitas susu Premium dan Baik memiliki nilai *learning rate* terbaik yang sama, yaitu 0.9. Rata-rata MSE yang dihasilkan oleh nilai *learning rate* terbaik untuk kualitas susu Premium didapatkan sebesar 0.016794, sedangkan rata-rata MSE yang dihasilkan oleh nilai *learning rate* terbaik untuk kualitas susu Baik didapatkan sebesar 0.005730.

6.2.4 Hasil Pengujian Terhadap Ukuran Populasi

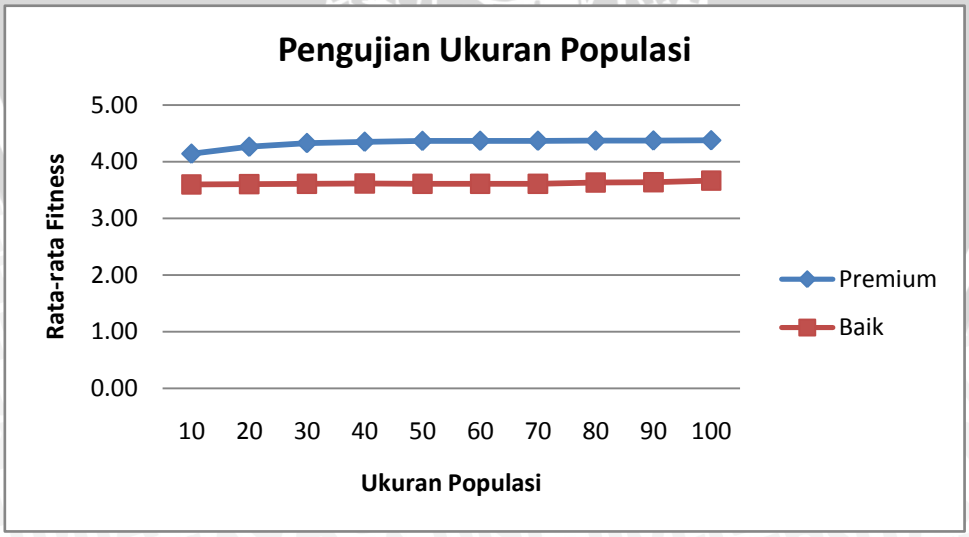
Pengujian terhadap ukuran populasi dilakukan untuk mendapatkan ukuran populasi terbaik yang memiliki nilai *fitness* tertinggi. Pengujian ini dilakukan sebanyak 10 kali untuk masing-masing ukuran populasi dengan *range* ukuran populasi yang digunakan adalah 10-100 dengan kelipatan sebesar 10. Data yang

digunakan adalah 30 data latih dan 30 data uji untuk kualitas susu Premium dan 15 data latih dan 15 data uji untuk kualitas susu Baik, jumlah iterasi = 1000 berdasarkan hasil dari pengujian jumlah iterasi, jumlah *hidden layer* = 10 berdasarkan hasil dari pengujian jumlah *hidden layer*, nilai *learning rate* = 0.9 berdasarkan hasil dari pengujian nilai *learning rate*, galat maksimal = 0.001, jumlah generasi = 250, *crossover rate* = 0.5 dan *mutation rate* = 0.1. Hasil pengujian pada ukuran populasi diperlihatkan pada Tabel 6.4 Hasil Pengujian Terhadap Ukuran Populasi dan hasil pengujian secara lengkap untuk kualitas susu Premium dan Baik dapat dilihat pada Lampiran E dan F.

Tabel 6.4 Hasil Pengujian Terhadap Ukuran Populasi

Ukuran Populasi	Nilai *Fitness* Percobaan ke-*i*										Rata-rata *Fitness*
	1	2	3	4	5	6	7	8	9	10	
10	4.220325	4.173376	...	...	...	...	...	...	...	4.5456618	4.142716
20	4.146335	4.033847	...	...	...	...	...	...	...	4.6017966	4.264439
30	4.131281	4.144516	...	...	...	...	...	...	...	4.3561902	4.331150
40	4.091596	4.326526	...	...	...	...	...	...	...	4.4506684	4.353579
50	3.893534	4.461145	...	...	...	...	...	...	...	4.4510487	4.368866
60	4.729255	4.168619	...	...	...	...	...	...	...	4.2898619	4.370491
70	4.169507	4.453774	...	...	...	...	...	...	...	4.5107143	4.371162
80	4.199015	4.501404	...	...	...	...	...	...	...	4.4432725	4.372364
90	4.755224	4.359783	...	...	...	...	...	...	...	4.3001735	4.375300
100	4.226962	4.231873	...	...	...	...	...	...	...	4.3778969	4.379729

Berdasarkan Tabel 6.4 Hasil Pengujian Terhadap Ukuran Populasi, maka dibuat grafik hasil pengujian seperti yang diperlihatkan pada Gambar 6.4.



Gambar 6.4 Grafik Hasil Pengujian Terhadap Ukuran Populasi

Besarnya ukuran populasi yang digunakan oleh sistem menunjukkan bahwa inidividu induk yang diciptakan oleh sistem sangat bervariasi. Hal tersebut memungkinkan sistem untuk mendapatkan solusi yang lebih beragam.

Berdasarkan Gambar 6.4 Grafik Hasil Pengujian Terhadap Ukuran Populasi, didapatkan bahwa semakin besar ukuran populasi yang digunakan, maka nilai *fitness* yang didapatkan untuk kualitas susu yang Premium dan Baik juga semakin stabil dan bernilai besar. Ukuran populasi terbaik dengan nilai rata-rata *fitness* tertinggi untuk kualitas susu Premium dan Baik memiliki ukuran populasi terbaik yang sama, yaitu 100. Rata-rata *fitness* yang dihasilkan oleh ukuran populasi terbaik untuk kualitas susu Premium didapatkan sebesar 4.379729, sedangkan rata-rata *fitness* yang dihasilkan oleh ukuran populasi terbaik untuk kualitas susu Baik didapatkan sebesar 3.667492.

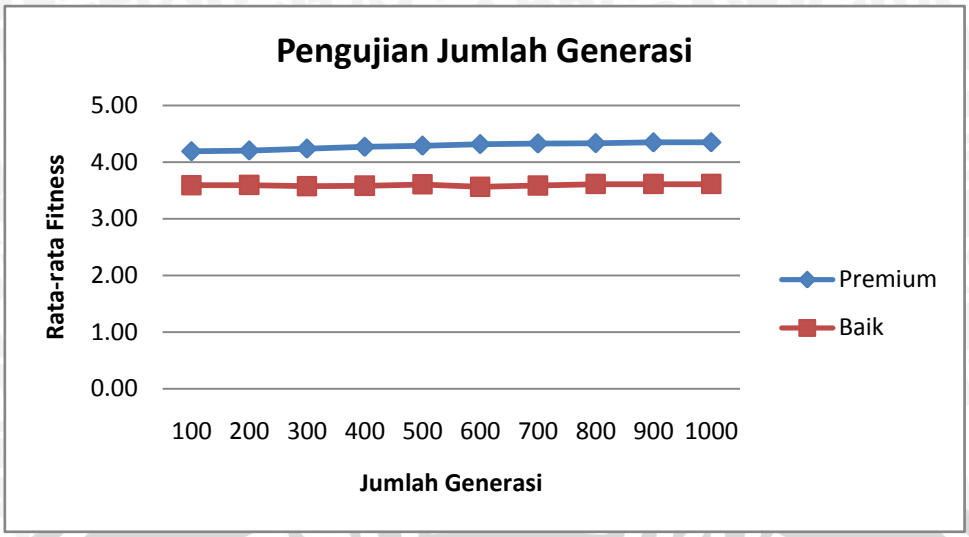
6.2.5 Hasil Pengujian Terhadap Jumlah Generasi

Pengujian terhadap jumlah generasi dilakukan untuk mendapatkan jumlah generasi terbaik yang memiliki nilai *fitness* tertinggi. Pengujian ini dilakukan sebanyak 10 kali untuk masing-masing jumlah generasi dengan *range* jumlah generasi yang digunakan adalah 100-1000 dengan kelipatan sebesar 100. Data yang digunakan adalah 30 data latih dan 30 data uji untuk kualitas susu Premium dan 15 data latih dan 15 data uji untuk kualitas susu Baik, jumlah iterasi = 1000 berdasarkan hasil dari pengujian jumlah iterasi, jumlah *hidden layer* = 10 berdasarkan hasil dari pengujian jumlah *hidden layer*, nilai *learning rate* = 0.9 berdasarkan hasil dari pengujian nilai *learning rate*, galat maksimal = 0.001, ukuran populasi = 100 berdasarkan hasil dari pengujian ukuran populasi, *crossover rate* = 0.5 dan *mutation rate* = 0.1. Hasil pengujian yang didapatkan pada ukuran populasi diperlihatkan pada Tabel 6.5 Hasil Pengujian Terhadap Jumlah Generasi dan hasil pengujian secara lengkap untuk kualitas susu Premium dan Baik dapat dilihat pada Lampiran E dan F.

Tabel 6.5 Hasil Pengujian Terhadap Jumlah Generasi

Jumlah Generasi	Nilai *Fitness* Percobaan ke-*i*										Rata-rata *Fitness*
	1	2	3	4	5	6	7	8	9	10	
100	4.034184	4.290529	...	...	...	...	...	...	...	4.237753	4.191343
200	4.498253	4.240378	...	...	...	...	...	...	...	4.012021	4.205530
300	4.097739	4.459041	...	...	...	...	...	...	...	4.309848	4.238523
400	4.563761	4.149908	...	...	...	...	...	...	...	4.261142	4.270270
500	4.510282	4.443940	...	...	...	...	...	...	...	4.280651	4.290704
600	4.157067	4.191511	...	...	...	...	...	...	...	4.726597	4.318531
700	4.322783	4.271945	...	...	...	...	...	...	...	4.544034	4.329780
800	4.473616	4.293715	...	...	...	...	...	...	...	4.677269	4.335209
900	4.267321	4.541325	...	...	...	...	...	...	...	4.607851	4.349221
1000	4.430069	4.790773	...	...	...	...	...	...	...	4.421393	4.351150

Berdasarkan Tabel 6.5 Hasil Pengujian Terhadap Jumlah Generasi, maka dibuat grafik hasil pengujian seperti yang diperlihatkan pada Gambar 6.5 Grafik Hasil Pengujian Terhadap Jumlah Generasi sebagai berikut.



Gambar 6.5 Grafik Hasil Pengujian Terhadap Jumlah Generasi

Jumlah generasi memiliki pengaruh terhadap nilai *fitness* yang dihasilkan. Kemampuan eksploitasi dalam GA akan semakin meningkat seiring bertambahnya jumlah generasi. Jumlah generasi yang banyak membuat ruang pencarian solusi menjadi lebih luas. Berdasarkan Gambar 6.5 Grafik Hasil Pengujian Terhadap Jumlah Generasi, didapatkan bahwa semakin besar jumlah generasi yang digunakan, maka nilai *fitness* yang didapatkan untuk kualitas susu yang Premium dan Baik juga semakin besar, sehingga kemungkinan besar akan mendapatkan solusi yang optimal. Jumlah generasi terbaik dengan nilai rata-rata *fitness* tertinggi untuk kualitas susu Premium dan Baik memiliki jumlah generasi terbaik yang sama, yaitu 1000. Rata-rata *fitness* yang dihasilkan oleh jumlah generasi terbaik untuk kualitas susu Premium didapatkan sebesar 4.351150, sedangkan rata-rata *fitness* yang dihasilkan oleh jumlah generasi terbaik untuk kualitas susu Baik didapatkan sebesar 3.6120187.

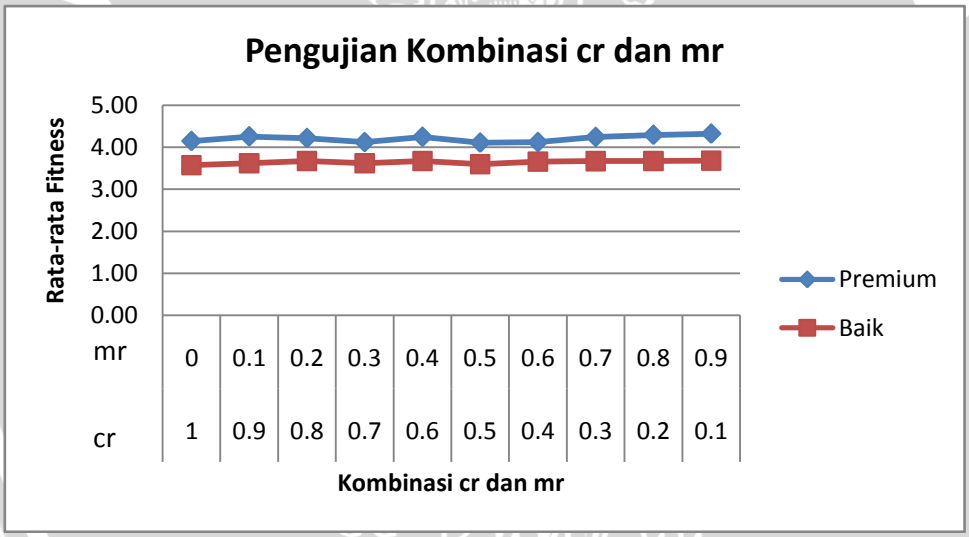
6.2.6 Hasil Pengujian Terhadap Kombinasi *Crossover Rate* dan *Mutation Rate*

Pengujian terhadap kombinasi *crossover rate* (*cr*) dan *mutation rate* (*mr*) dilakukan untuk mendapatkan kombinasi *cr* dan *mr* terbaik yang memiliki nilai *fitness* tertinggi. Pengujian ini dilakukan sebanyak 10 kali untuk masing-masing kombinasi *cr* dan *mr*. Data yang digunakan adalah 30 data latih dan 30 data uji, jumlah iterasi = 1000 berdasarkan hasil dari pengujian jumlah iterasi, jumlah *hidden layer* = 10 berdasarkan hasil dari pengujian jumlah *hidden layer*, nilai *learning rate* = 0.9 berdasarkan hasil dari pengujian nilai *learning rate*, galat maksimal = 0.001, ukuran populasi = 100 berdasarkan hasil dari pengujian ukuran populasi dan jumlah generasi = 1000 berdasarkan hasil dari pengujian jumlah generasi. Hasil pengujian yang didapatkan pada kombinasi *crossover rate* dan *mutation rate* diperlihatkan pada Tabel 6.6 Hasil Pengujian Terhadap Kombinasi *cr* dan *mr* dan hasil pengujian secara lengkap untuk kualitas susu Premium dan Baik dapat dilihat pada Lampiran E dan F.

Tabel 6.6 Hasil Pengujian Terhadap Kombinasi *cr* dan *mr*

cr	*mr*	Nilai *Fitness* Percobaan ke-*i*										Rata-rata *Fitness*
		1	2	3	4	5	6	7	8	9	10	
1	0	4.086611	3.984560	...	...	...	...	...	...	...	4.326498	4.149786
0.9	0.1	4.326807	4.186959	...	...	...	...	...	...	...	4.359140	4.259300
0.8	0.2	4.562964	3.974613	...	...	...	...	...	...	...	4.272020	4.219672
0.7	0.3	4.429833	4.430973	...	...	...	...	...	...	...	4.012381	4.122506
0.6	0.4	4.437079	4.038598	...	...	...	...	...	...	...	4.445610	4.248272
0.5	0.5	4.021439	4.455867	...	...	...	...	...	...	...	3.981198	4.110970
0.4	0.6	3.981198	3.981199	...	...	...	...	...	...	...	4.145922	4.123593
0.3	0.7	4.240663	4.182389	...	...	...	...	...	...	...	4.439150	4.244916
0.2	0.8	4.363322	4.468750	...	...	...	...	...	...	...	4.603837	4.295722
0.1	0.9	4.295721	4.032325	...	...	...	...	...	...	...	4.348067	4.324093

Berdasarkan Tabel 6.6 Hasil Pengujian Terhadap Kombinasi *cr* dan *mr*, maka dibuat grafik hasil pengujian seperti yang diperlihatkan pada Gambar 6.6 Grafik Hasil Pengujian Terhadap Kombinasi *cr* dan *mr* sebagai berikut.



Gambar 6.6 Grafik Hasil Pengujian Terhadap Kombinasi *cr* dan *mr*

Kombinasi *crossover rate* (*cr*) dan *mutation rate* (*mr*) memiliki pengaruh terhadap nilai *fitness* yang akan dihasilkan. Semakin besar nilai *crossover rate* dan semakin kecil nilai *mutation rate*, maka kemampuan *Genetic Algorithm* (GA) untuk memperlebar area pencarian akan semakin menurun. Sedangkan, jika semakin kecil nilai *crossover rate* dan semakin besar nilai *mutation rate*, maka kemampuan *Genetic Algorithm* (GA) untuk belajar dari generasi sebelumnya akan menurun dan tidak mampu untuk melakukan eksploitasi area dari *optimum local* secara efektif. Nilai *cr* dan *mr* yang optimal menunjukkan keseimbangan dalam hal eksplorasi dan eksploitasi. Berdasarkan Gambar 6.6 Grafik Hasil Pengujian Terhadap Kombinasi *cr* dan *mr*, didapatkan bahwa semakin besar nilai *mr* dan semakin kecil nilai *cr*, maka nilai *fitness* yang didapatkan untuk kualitas susu yang Premium dan Baik juga semakin besar. Kombinasi *cr* dan *mr* terbaik

dengan nilai rata-rata *fitness* tertinggi untuk kualitas susu Premium dan Baik memiliki kombinasi *cr* dan *mr* terbaik yang sama, yaitu 0.1 dan 0.9. Rata-rata *fitness* yang dihasilkan oleh kombinasi *cr* dan *mr* terbaik untuk kualitas susu Premium didapatkan sebesar 4.324093, sedangkan rata-rata *fitness* yang dihasilkan oleh kombinasi *cr* dan *mr* terbaik untuk kualitas susu Baik didapatkan sebesar 3.679464.

6.2.7 Hasil Pengujian Terhadap Kombinasi Variasi *Output* yang Dioptimalkan

Pengujian terhadap kombinasi variasi *output* yang dioptimalkan dilakukan untuk mengetahui nilai rata-rata *fitness* yang diperoleh dari kombinasi *output* yang berupa kandungan gizi susu protein yang dioptimalkan dan kandungan gizi susu lemak yang diminimalkan. Pengujian ini dilakukan sebanyak 10 kali untuk masing-masing kombinasi variasi *output*. Data yang digunakan adalah 30 data latih dan 30 data uji untuk kualitas susu Premium dan 15 data latih dan 15 data uji untuk kualitas susu Baik, jumlah iterasi = 1000 berdasarkan hasil dari pengujian jumlah iterasi, jumlah *hidden layer* = 10 berdasarkan hasil dari pengujian jumlah *hidden layer*, nilai *learning rate* = 0.9 berdasarkan hasil dari pengujian nilai *learning rate*, galat maksimal = 0.001, ukuran populasi = 100 berdasarkan hasil dari pengujian ukuran populasi, jumlah generasi = 1000 berdasarkan hasil dari pengujian jumlah generasi, *cr* = 0.1 dan *mr* = 0.9 berdasarkan hasil dari pengujian kombinasi *cr* dan *mr*.

Hasil pengujian yang didapatkan pada kombinasi variasi *output* yang dioptimalkan diperlihatkan pada Tabel 6.7 Hasil Pengujian Terhadap Kombinasi Variasi *Output*.

Tabel 6.7 Hasil Pengujian Terhadap Kombinasi Variasi *Output*

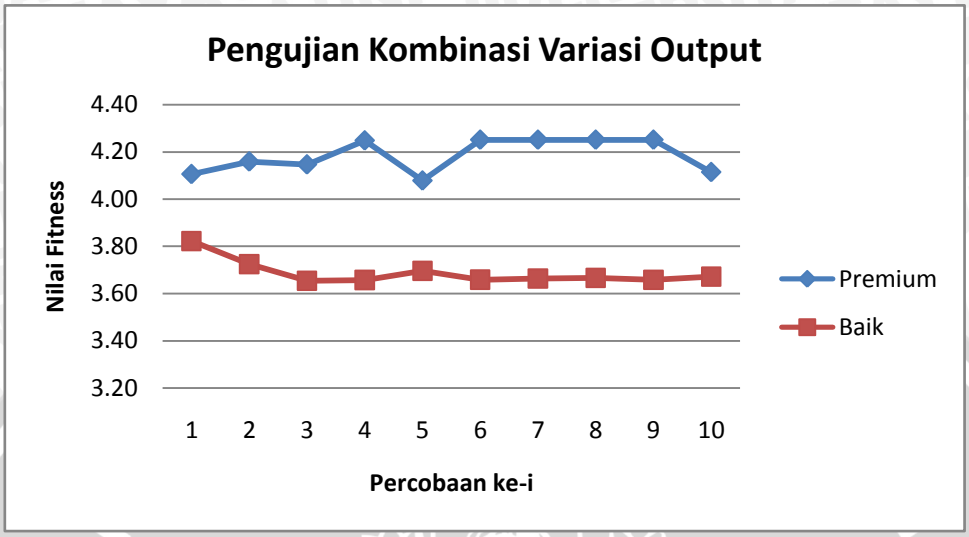
Kandungan Gizi Susu		Nilai *Fitness* Percobaan ke-*i*										Rata-rata *Fitness*
Optimalkan	Minimalkan	1	2	3	4	5	6	7	8	9	10	
Protein	Lemak	4.10605	...	...	...	...	...	...	...	...	4.11450	4.18585

Hasil pengujian terhadap kenaikan kandungan gizi susu protein dan penurunan kandungan gizi lemak untuk kualitas susu Premium diperlihatkan pada Tabel 6.8 Hasil Pengujian Terhadap Kenaikan Protein dan Penurunan Lemak. Sedangkan, hasil pengujian kenaikan kandungan gizi susu protein dan penurunan kandungan gizi lemak untuk kualitas susu Premium dan Baik diperlihatkan secara lengkap pada Lampiran E dan F.

Tabel 6.8 Hasil Pengujian Terhadap Kenaikan Protein dan Penurunan Lemak

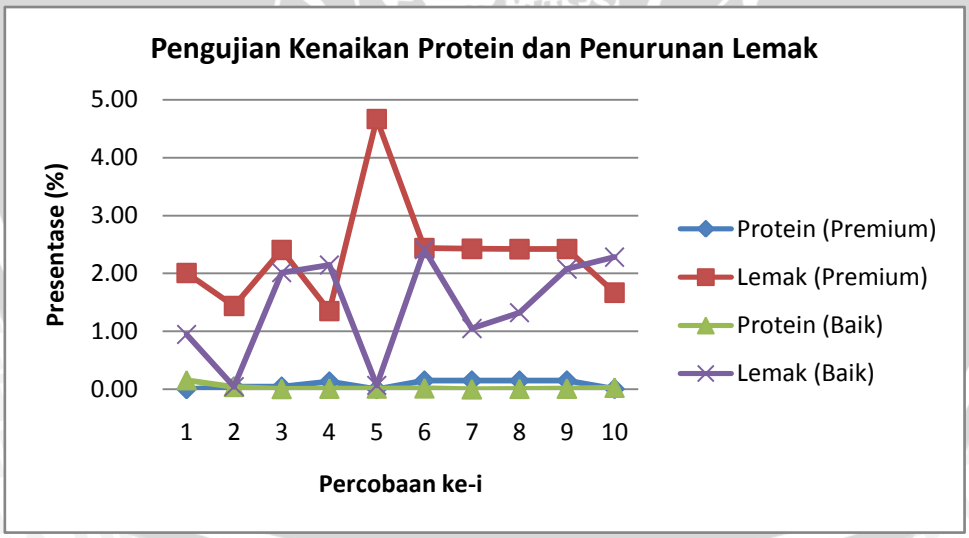
Kasus Optimasi (Protein Optimal dan Lemak Minimal)	Nilai Presentase Percobaan ke-*i*										Rata-rata Presentase
	1	2	3	4	5	6	7	8	9	10	
Protein Dioptimalkan	0.0026	...	...	...	...	...	...	...	...	0.0054	0.0852
Lemak Diminimalkan	2.0105	...	...	...	...	...	...	...	...	1.6685	2.3254

Hasil visualisasi dalam bentuk grafik terhadap Tabel 6.7 Hasil Pengujian Terhadap Kombinasi Variasi *Output* dan Tabel 6.8 Hasil Pengujian Terhadap Kenaikan Protein dan Penurunan Lemak diperlihatkan pada Gambar 6.7 dan 6.8 sebagai berikut.



Gambar 6.7 Grafik Hasil Pengujian Terhadap Kombinasi Variasi *Output*

Berdasarkan Gambar 6.7 Hasil Pengujian Terhadap Kombinasi Variasi *Output* dalam mengoptimalkan protein dan meminimalkan lemak, maka diperoleh rata-rata nilai *fitness* untuk kualitas susu Premium sebesar 4.18585 dan rata-rata nilai *fitness* untuk kualitas susu Baik sebesar 3.68684.



Gambar 6.8 Grafik Hasil Pengujian Terhadap Kenaikan Protein dan Penurunan Lemak

Berdasarkan Gambar 6.8 Grafik Hasil Pengujian Terhadap Kenaikan Protein dan Penurunan Lemak, maka diperoleh rata-rata presentase kenaikan protein dan penurunan lemak untuk kualitas susu Premium sebesar 0.0852% dan 2.3254%. Sedangkan, rata-rata presentase kenaikan protein dan penurunan lemak untuk kualitas Baik sebesar 0.0292% dan 1.4372%.

BAB 7 PENUTUP

Pada bab penutup akan menjelaskan tentang kesimpulan dan saran yang diperoleh dari hasil penelitian yang telah dilakukan pada bab sebelumnya tentang pemodelan komposisi pakan kambing PE untuk optimasi kandungan gizi susu menggunakan ANN dan GA. Kesimpulan yang diambil adalah untuk menjawab rumusan masalah dan menyelesaikan permasalahan yang telah dijelaskan pada Bab 1. Kesimpulan akan dijelaskan dalam sub bab 7.1 dan saran akan diberikan pada sub bab 7.2.

7.1 Kesimpulan

Berdasarkan penelitian dan pengujian yang telah dilakukan terhadap pemodelan komposisi pakan kambing PE untuk optimasi kandungan gizi susu menggunakan *Artificial Neural Network* (ANN) dan *Genetic Algorithm* (GA) dapat diambil beberapa kesimpulan sebagai berikut:

1. Metode *Artificial Neural Network* (ANN) dan *Genetic Algorithm* (GA) dapat diterapkan pada pemodelan komposisi pakan kambing Peranakan Etawa (PE) untuk mengoptimalkan kandungan gizi susu kambing yang diimplementasikan menggunakan bahasa pemrograman Java. Dalam implementasi metode ANN dan GA, langkah pertama yang dilakukan adalah proses pelatihan terhadap data latih dan pengujian terhadap data uji yang tersedia. Objek yang diamati sebagai *input* adalah kandungan nutrisi pakan yang berupa abu, protein kasar (PK), lemak kasar (LK) dan serat kasar (SK). *Output* kandungan gizi susu yang diamati sebagai target adalah protein, lemak, laktosa dan *density.* Parameter yang terdapat pada ANN akan melatih data untuk mencari bobot terbaik dan memprediksi kandungan gizi susu dengan menggunakan MSE (*Mean Square Error*). Semain kecil nilai MSE yang diperoleh, maka sistem yang dihasilkan akan semakin baik. Langkah kedua yang dilakukan adalah memproses satu atau lebih kandungan gizi susu yang akan dioptimalkan menggunakan GA. Parameter yang terdapat pada GA akan melakukan seleksi pada semua individu yang terdapat pada satu generasi untuk dipilih sebagai individu terbaik yang memiliki nilai *fitness* tertinggi. Hasil keluaran yang dihasilkan oleh sistem berupa rekomendasi jumlah pemberian dari masing-masing pakan hijauan yang telah dipilih sebelumnya untuk mendapatkan satu atau lebih kandungan gizi susu yang lebih optimal ataupun minimal.
2. Nilai parameter optimal dari metode ANN dan GA yang diperoleh sistem berdasarkan hasil pengujian yang telah dilakukan terhadap kualitas susu kambing PE Premium dan Baik adalah sama, yaitu dengan menggunakan jumlah iterasi = 1000, jumlah *hidden layer* = 10, nilai *learning rate* = 0.9, ukuran populasi = 100, jumlah generasi = 1000, *cr* = 0.1 dan *mr* = 0.9.
3. Hasil kualitas solusi dari optimasi kandungan gizi susu kambing Peranakan Etawa (PE) dengan menggunakan parameter ANN dan GA yang optimal untuk

kasus pengujian berupa mengoptimalkan protein dan meminimalkan lemak pada bobot badan kambing 36 kg, maka diperoleh rata-rata kandungan protein untuk kualitas Premium naik sebesar 0.0852% dan rata-rata kandungan lemak untuk kualitas Premium turun sebesar 2.3254%. Sedangkan, rata-rata kandungan protein dengan kualitas Baik naik sebesar 0.0292% dan rata-rata kandungan lemak dengan kualitas Baik turun sebesar 1.4372%.

7.2 Saran

1. Pada penelitian ini terdapat masalah dalam pencarian jumlah iterasi ANN untuk mencapai kondisi yang konvergen, sehingga disarankan untuk mencoba menggunakan algoritma *nguyen-widrow* sebagai perbandingan pada sistem dalam hal cepat atau tidaknya proses *running time* sistem dan bobot akhir yang didapatkan.
2. Jumlah pakan hijauan yang dimodelkan dan dioptimalkan dapat dikembangkan menjadi lebih banyak, sehingga pengguna dapat memodelkan dan mengoptimalkan lebih dari dua jenis pakan hijauan.
3. Hasil penelitian dari implementasi algoritma ANN dan GA untuk memodelkan komposisi pakan kambing Peranakan Etawa (PE) untuk optimasi kandungan gizi susu dapat dijadikan sebagai penelitian lebih lanjut dengan tambahan yang berupa mempertimbangkan harga pakan agar diperoleh harga pakan yang minimal.

DAFTAR PUSTAKA

Akuntono, I. (2015, September 29). *Mengkhawatirkan, Angka Kelahiran di RI Tiap Tahun Setara Jumlah Penduduk Singapura*. Retrieved Januari 8, 2016, from Kompas.com [online]: http://nasional.kompas.com/read/2015/09/29/13574351/Mengkhawatirkan.Angka.Kelahiran.di.RI.Tiap.Tahun.Sejumlah.Penduduk.Singapura

Aribowo, A., Lukas, S., & Gunawan, M. (2008). Penerapan Algoritma Genetika pada Penentuan Komposisi Pakan Ayam Petelur. *Seminar Nasional Aplikasi Teknologi Informasi* , B-21 - B-24.

Bramantyo, N. (2006). *Aplikasi Algoritma Genetika untuk Optimasi Desain Resonator Hemlholtz Ganda Menggunakan Matlatb 7.0*. Surakarta: Fakultas Matematika dan Ilmu Pengetahuan Alam : Universitas Sebelas Maret.

Budiarsana, I., & Sutama, I. K. (2006). Karakteristik Produktivitas Kambing Peranakan Etawah. *Balai Penelitian Ternak* , 215-220.

Budiarsana, I.-G., & Sutama, I.-K. (2001). Efisiensi Produksi Susu Kambing Peranakan Etawah. *Seminar Nasional Teknologi Peternakan dan Veteriner* , 427-434.

Cahyono, B. (2006). *Beternak Domba dan Kambing*. Yogyakarta: Kanisius, Cetakan ke Sembilan.

Chamidah, N., Wiharto, & Salamah, U. (2012). Pengaruh Normalisasi Data pada Jaringan Syaraf Tiruan Backpropagasi Gradient Descent Adaptive Gain (BPGDAG) untuk Klasifikasi. *Jurnal ITSMART, Vol.1, No.1* , 28-33.

Edo, R. Z. (2014). Implementasi Backpropagation Neural Network dalam Pembangkitan Otomatis Fungsi Keanggotaan Fuzzy pada Penderita Penyakit Hepatitis. *Skripsi Program Teknologi Informasi dan Ilmu Komputer. Universitas Brawijaya* .

Fakharudin, A. S., Sulaiman, M. N., Salihon, J., & Zainol, N. (2013). Implementing Artificial Neural Network and Genetic Algorithms to Solve Modeling and Optimisation of Biogas Production. *International Conference on Computing and Informatics* , 121-126.

Fernanda, J. W., & Otok, B. W. (2012). Boosting Neural Network dan Boosting Cart pada Klasifikasi Diabetes Militus Tiper II. *Jurnal Matematika, Vol. 2, No. 2* , 33-49.

Fitriyanto, Astuti, T. Y., & Utami, S. (2013). Kajian Viskositas dan Berat Jenis Susu Kambing Peranakan Etawa(PE) pada Awal, Puncak dan Akhir Laktasi. *Jurnal Ilmiah Peternakan* , 299-306.

Gema, R. L. (2014). Jaringan Syaraf Tiruan Propagasi Balik dalam Prediksi Persediaan Ternak Sapi Potong (Studi Kasus di Wilayah Sumatera Barat). *Jurnal KomTekInfo, Fakultas Ilmu Komputer, Vol.1, No.2* , 21-27.

Gunawan, H. (2013). *Prospek Usaha Penggemukan Kambing Potong.* Jakarta: Pustaka Baru Press.

Hamidah, E., Sukada, I. M., & Swacita, I. B. (2012). Kualitas Susu Kambing Peranakan Etawah Post-Thawing pada Penyimpanan Suhu Kamar. *Indonesia Medicus Veterinus Edisi 3* , 361-369.

Hidayati, A. (2012, Mei 30). *Pakan Kambing Perah*. Retrieved Januari 20, 2016, from Universitas Muhammadiyah Malang: http://peternakan.umm.ac.id/id/umm-news-2862-pakan-kambing-perah.html

Istadi, & Rahmayanti, D. (2010). Pemodelan dan Optimasi Hidrolisa Pati Menjadi Glukosa dengan Metode Artificial Neural Network-Genetic Algorithm(ANN-GA). *Fakultas Teknik Universitas Diponegoro, Vol. 31, No.2* , 102-113.

Koten, B., Wea, R., Soetrisno, R., Ngadiyono, N., & Soewignyo, B. (2014). Konsumsi Nutrien Ternak Kambing yang Mendapatkan Hijauan Hasil Tumpangsari Arbila (Phaseolus lunatus) dengan Sorgum sebagai Tanaman Sela pada Jarak Tanam Arbila dan Jumlah Baris Sorgum yang Berbeda. *Ilmu Ternak, Vol.1, No.8* , 38-45.

Mahmudy, W. F. (2013). *Algoritma Evolusi.* Malang: Program Teknologi Informasi dan Ilmu Komputer, Universitas Brawijaya.

Najib, M., Purwananto, Y., & Soelaiman, R. (2012). Hibridisasi Artificial Neural Network dan Genetic Algoritm dalam Penentuan Harga Produk pada Industri Mobil. *Fakultas Teknologi Informasi, ITS* , 1-6.

Nur, D. M., Setyowati, E., & Wahyuningsih, S. (2015). Performans Produksi Susu Kambing Peranakan Etawah(PE) Berdasarkan Paritas, Umur, Bobot Badan, dan Status Kebuntingan di "MADUKARA FARM", Kota Batu. *Universitas Brawijaya* , 1-7.

Nurmila, N., Sugiharto, A., & Sarwoko, E. A. (2012). Algoritma Back Propagation Neural Network untuk Pengenalan Pola Karakter Huruf Jawa. *Jurnal Masyarakat Informatika, Vol. 1, No. 1* , 1-10.

Pramesti, D., Mahmudy, W. F., & Indriati. (2015). Optimasi Komposisi Pakan Kambing Potong Menggunakan Algoritma Genetika. *DORO : Repository Jurnal Mahasiswa PTIIK Universitas Brawijaya, Vol. 5, No. 13* .

Prawira, P., Djaja, Z., & Imelda. (2007). *Susu Kambing Nutrisi Penting Sekaya Air Susu Ibu.* Jakarta: Diamond Interest International.

Prihatminingsih, G. E., Purnomoadi, A., & Harjanti, D. W. (2015). Hubungan antara Konsumsi Protein dengan Produksi, Protein dan Laktosa Susu Kambing Peranakan Ettawa. *Jurnal Ilmu-ilmu Peternakan 25 Vol.2* , 20-27.

Raharjo, J. S. (2013). Model Artificial Neural Network Berbasis Particle Swarm Optimization untuk Prediksi Laju Inflasi. *Jurnal Sistem Komputer, Vol. 3, No. 1* , 10-21.

Sabati, D., Dania, W. A., & Putri, S. A. (2014). Peramalan Permintaan Sari Apel dengan Metode Jaringan Syaraf Tiruan (JST) di KSU Brosesm, Batu. *Fakultas Teknologi Pertanian, Universitas Brawijaya* , 1-10.

Subekti, E. (2009). Ketahanan Pakan Ternak Indonesia. *Mediagro, Vol.5, No.2* , 63-71.

Suhendra, C. D., & Wardoyo, R. (2015). Penentuan Arsitektur Jaringan Syaraf Tiruan Backpropagation (Bobot Awal dan Bias Awal) Menggunakan Algoritma Genetika. *IJCCS, Vol. 9, No. 1* , 77-88.

Susilawati, T., & dkk. (2011). *Agribisnis Kambing.* Malang: UB Press, Cetakan Pertama.

Susilowati, D. R., Utami, S., & Suratim, H. A. (2013). Nilai Berat Jenis dan Total Solid Susu Kambing Sapera di Cilacap dan Bogor. *Jurnal Ilmiah Peternakan, Vol. 1, No. 3* , 1071-1077.

Sutama, I. K. (2011, Oktober 19-25). Kambing Peranakan Etawah Sumberdaya Ternak Penuh Berkah. *SinarTani, Agroinovasi* , pp. 5-13.

Sutama, I. K., & Budiarsana, I. (1997). Kambing Peranakan Etawah Penghasil Susu sebagai Sumber Pertumbuhan Baru Sub-Sektor Peternakan di Indonesia. *Balai Penelitian Ternak* , 156-170.

Umiyasih, U., & Anggraeny, Y. N. (2007). *Petunjuk Teknis Ransum Seimbang, Strategi Pakan pada Sapi Potong.* Pasuruan: Pusat Penelitian dan Pengembangan Peternakan.

Wuryandari, M. D., & Afrianto, I. (2012). Perbandingan Metode Jaringan Syaraf Tiruan Backpropafation dan Learning Vector Quantization pada Pengenalan Wajah. *Jurnal Komputer dan Informatika, Edisi 1, Vol. 1* , 45-51.

Yuliandar, D., Warsito, B., & Yasin, H. (2012). Pelatihan Feed Forward Neural Network Menggunakan Algoritma Genetika dengan Metode Seleksi Turnamen untuk Data Time Series. *Gaussian, Vol. 1, No.1* , 65-72.

Zamani, A. M., Amaliah, B., & Munif, A. (2012). Implementasi Algoritma Genetika pada Struktur Backpropagation Neural Network untuk Klasifikasi Kanker Payudara. *Teknik Pomits, Vol. 1, No. 1* , 1-6.

Zukhri, Z. (2014). *Algoritma Genetika Metode Komputasi Evolusioner untuk Menyelesaikan Masalah Optimasi.* Yogyakarta: C. Vandi Offset.

Zuriati, Y., Maheswari, R., & Susanty, H. (2011). Karakteristik Kualitas Susu Segar dan Yoghurt dari Tiga Bangsa Kambing Perah dalam Mendukung Program Ketahanan dan Diversifikasi Pangan. *Seminar Nasional Teknologi Peternakan dan Veteriner* , 613-619.

LAMPIRAN A SURAT PERMOHONAN DATA SKRIPSI

KEMENTERIAN RISET, TEKNOLOGI, DAN PENDIDIKAN TINGGI
UNIVERSITAS BRAWIJAYA
PROGRAM TEKNOLOGI INFORMASI DAN ILMU KOMPUTER
Gedung A PTIIK Lt. 1, JL. Veteran No.8, Malang, 65145, Indonesia
Telp. : +62-341-577911; Fax : +62-341-577911
http://ptiik.ub.ac.id E-mail : ptiik@ub.ac.id

Nomor : 1492 /UN10.36/AK/2106 24 FEB 2016
Perihal : ***Permohonan data skripsi***

Yth. Kepala Dinas Peternakan Provinsi Jawa Timur
Jl. Jend. A. Ayani 202
Fax (031) 8291853 - 8287165
email : disnak@jatimprov.go.id
Surabaya

Untuk mendukung penyelesaian skripsi mahasiswa berikut :

Nama	:	**Latifah Hanum**
NIM	:	**125150201111022**
Dosen P. A	:	**Indriati, S.T., M.Kom.**
Prodi	:	**Informatika / Ilmu Komputer**

Guna memperoleh data untuk skripsi mahasiswa tersebut di instansi saudara, jenis data yang diperlukan dan rencana waktu pelaksanaan adalah :

Data	:	**Data tertulis mengenai data kandungan nutrisi bahan pakan kambing, data komposisi pakan untuk kambing serta data hasil laboratorium untuk kandungan gizi susu**
Waktu	:	**01 Maret 2016 - 04 Maret 2016**

Atas perhatian dan kerjasamanya kami ucapkan terima kasih.

Mengetahui,
a.n. Ketua Program
Kepala Tata Usaha PTIIK,

Dra. Endang Budha Mahadyawati, M.AP.
NIP. 19581213 19820 2 2001

Dosen Pembimbing Akademik

Indriati, S.T., M.Kom.
NIK/ NIP. 200909 831013 2 001

Tembusan Kepada Yth:
1. Ketua Program Studi Informatika / Ilmu Komputer
2. UPTPT dan HMT Malang
3. Mahasiswa yang bersangkutan

LAMPIRAN B HASIL WAWANCARA

23 Desember 2015

1. Apa saja jenis kambing yang ada di UPT Pembibitan Ternak dan Hijauan Makanan Ternak Singosari-Malang?
   Jawab: Kambing perah peranakan etawa (PE) jenis kaligesing dan senduro.
   Narasumber: Ibu drh. Jaya Wulandari

2. Bagaimana ciri-ciri kambing PE yang dapat diambil susunya?
   Jawab: Kambing yang sedang dalam masa laktasi, yaitu kambing yang sudah pernah melahirkan dan sedang tidak hamil.
   Narasumber: Bapak Rokim, S.Pt dan Bapak Kayat

3. Berapa kali pemerahan susu yang dilakukan setiap satu harinya?
   Jawab: Kambing PE diperah setiap pagi hari jam 08.00 WIB dimana setiap satu hari proses pemerahannya hanya dilakukan satu kali saja.
   Narasumber: Bapak Rokim, S.Pt dan Bapak Kayat

4. Berapa jenis bobot badan kambing PE yang sudah siap perah?
   Jawab: Ada bermacam-macam jenis bobot badan kambing PE yang sudah siap perah, diantaranya 32 kg, 34 kg, 36 kg, 38 kg dan 40kg.
   Narasumber: Bapak Rokim, S.Pt dan Bapak Kayat

5. Apa saja jenis-jenis pakan hijauan dan *leguminose* yang tersedia di UPT Pembibitan Ternak dan Hijauan Makanan Ternak Singosari-Malang?
   Jawab: Ada 10 jenis pakan hijauan dan *leguminose* yang tersedia, yaitu rumput taiwan, rumput odot, rumput setaria, rumput raja, rumput *brachiaria decumbens*, kaliandra, gamal, lamtoro, turi dan tarum.
   Narasumber: Bapak Rokim, S.Pt

6. Diantara 10 jenis pakan hijauan dan *leguminose* yang tersedia, bahan pakan manakah yang sering diberikan pada kambing PE?
   Jawab: Rumput odot dan rumput raja.
   Narasumber: Bapak Rokim, S.Pt

7. Berapa kilogram pemberian pakan untuk kambing PE dalam satu harinya?
   Jawab: Pemberian pakan pada kambing PE diberikan berdasarkan bobot badan kambing tersebut. Pakan yang diberikan pada kambing PE perharinya, yaitu minimal 10% dari bobot badan kambing dan maksimal 12% dari bobot badan kambing.
   Narasumber: Bapak Rokim, S.Pt

8. Selain jenis pakan hijauan dan *leguminose*, adakah jenis pakan lainnya yang diberikan pada kambing PE?
   Jawab: Iya ada, selain pakan hijauan dan *leguminose*, kambing PE juga diberikan pakan berupa konsentrat yang berfungsi untuk tambahan asupan nutrisi pakan yang tidak diperoleh dari pakan hijauan dan *leguminose*.
   Narasumber: Bapak Rokim, S.Pt

9. Apa saja kandungan nutrisi yang terdapat pada pakan hijauan dan *leguminose*?
   Jawab: Kandungan nutrisi yang terdapat pada pakan sebenarnya banyak jenisnya, akan tetapi kandungan nutrisi dari pakan yang berpengaruh terhadap kualitas susu yang dihasilkan adalah abu, protein kasar, lemak kasar dan serat kasar.
   Narasumber: Bapak Rokim, S.Pt

10. Pernahkah pihak UPT Pembibitan Ternak dan Hijauan Makanan Ternak Singosari-Malang memberikan komposisi pakan yang berbeda-beda setiap bulannya?
    Jawab: Pihak UPT Pembibitan Ternak dan Hijauan Makanan Ternak Singosari-Malang tidak pernah memberikan *treatment* berupa pemberian komposisi pakan yang berbeda-beda setiap bulannya karena tidak semua kambing mampu beradaptasi terhadap makanan yang baru dan jika tidak cocok pada pencernaan kambing, maka akan berakibat buruk terhadap kesehatan kambing dan pemulihan kesehatan kambing bisa berlangsung cukup lama.
    Narasumber: Bapak Rokim, S.Pt dan Ibu drh. Jaya Wulandari

11. Adakah uji kandungan gizi susu kambing PE di UPT Pembibitan Ternak dan Hijauan Makanan Ternak Singosari-Malang?
    Jawab: Ada, uji kandungan gizi susu kambing PE dilakukan di laboratorium milik UPT Pembibitan Ternak dan Hijauan Makanan Ternak Singosari-Malang menggunakan alat yang bernama *lactoscan*.
    Narasumber: Ibu drh. Jaya Wulandari dan Bapak Kayat

12. Apa saja kandungan gizi susu yang dihasilkan dari uji laboratorium?
    Jawab: Hasil dari uji laboratorium menggunakan *lactoscan*, yaitu ada lemak, SNF, *density*, *lactose*, *solids*, protein, *added water*, *temperature sample* dan *freeze point*.
    Narasumber: Ibu drh. Jaya Wulandari dan Bapak Kayat

13. Dari macam-macam kandungan gizi susu yang sudah dijelaskan sebelumnya, manakah kandungan gizi susu yang mempunyai pengaruh besar, baik dalam hal kesehatan ataupun kualitas susu yang disukai konsumen dan apa alasannya?
    Jawab: Kandungan gizi susu yang mempunyai pengaruh besar bagi kesehatan ataupun kualitas susu yang disukai konsumen adalah protein, lemak, laktosa

dan *density*. Semakin tinggi kandungan protein, maka susu tersebut akan semakin bagus untuk tubuh. Lemak susu juga bagus untuk tubuh, namun banyak orang yang alergi terhadap lemak susu sapi, sehingga susu kambing yang rendah lemak dapat dijadikan sebagai alternatif untuk orang-orang yang alergi pada lemak susu sapi. Besarnya *density* yang ada pada susu kambing akan membuat susu menjadi semakin kental dan semakin mahal harga jualnya. Selain itu, laktosa yang besar juga akan mempengaruhi rasa kombinasi manis dan asin pada susu.
Narasumber: Ibu drh. Jaya Wulandari

31 Maret 2016

1. Adakah klasifikasi hasil susu berdasarkan kandungan gizi susu yang diperoleh?
   Jawab: Ada. Terdapat tiga klasifikasi penggolongan kualitas susu, yaitu kelas Premium, Baik dan Standar.
   Narasumber: Ibu drh. Jaya Wulandari

2. Bagaimana cara mengklasifikasikan penggolongan kualitas susu berdasarkan kelasnya?
   Jawab: Kandungan gizi susu yang dijadikan patokan adalah protein. Jika protein yang dihasilkan lebih dari 3.7%, maka susu tersebut digolongkan sebagai kualitas Premium. Jika protein yang dihasilkan antara 3.4% - 3.7%, maka susu tersebut digolongkan sebagai kualitas Baik dan jika protein yang dihasilkan kurang dari 3.4%, maka susu tersebut digolongkan sebagai kualitas Standar.
   Narasumber: Ibu drh. Jaya Wulandari

3. Menurut Anda, kandungan gizi susu manakah yang dapat dioptimalkan dan diminimalkan?
   Jawab: Semakin tinggi kandungan protein yang ada di dalam susu, maka semakin bagus pula untuk kesehatan tubuh manusia sehingga protein dapat dioptimalkan kandungannya. Lemak pada susu kambing memiliki keistimewaan dibandingkan lemak susu sapi, yaitu kandungan lemak yang lebih rendah dari susu sapi sehinga lemak dapat diminimalkan kandungannya. Laktosa digunakan untuk menambah rasa pada susu kambing, jadi semakin tinggi kandungan laktosa yang ada pada susu, maka rasanya akan menjadi semakin gurih sehingga laktosa dapat dioptimalkan. Semakin tinggi kandungan *density*, maka susu yang dihasilkan juga akan semakin kental sehingga *density* dapat dioptimalkan. Susu yang memiliki rasa gurih dan kental adalah susu yang sangat disukai oleh konsumen.
   Narasumber: Ibu drh. Jaya Wulandari

4. Apakah ada formulasi baku yang menunjukkan pengaruh pakan yang diberikan terhadap hasil produksi susu kambing?
   Jawab: Untuk formulasi bakunya tidak ada, dari Dinas Provinsi juga tidak memberikan aturan baku mengenai pengaruh pakan kambing terhadap hasil susu, hanya saja sesuai dengan fakta di lapangan, jika pakan yang diberikan memiliki kandungan nutrisi yang baik, juga pasti akan berpengaruh besar ke hasil susunya. Susu yang dihasilkan nanti juga akan memiliki kandungan gizi yang tinggi juga. Dan itupun tidak ada catatan tertulisnya, jadi berdasarkan pengalaman saja.
   Narasumber: Bapak Rokim, S.Pt dan Ibu drh. Jaya Wulandari

LAMPIRAN C DATA UTAMA HASIL UJI LABORATORIUM

No.	Protein (%)	*Fat* (%)	*Lactose* (%)	*Density*	Bobot Badan (kg)	Klasifikasi Berdasarkan THAI
1	4.11	7.18	3.93	28.49	32	Premium
2	4.08	7.07	3.89	28.95	32	Premium
3	3.83	8.19	3.67	25.33	32	Premium
4	3.97	6.52	3.79	27.8	32	Premium
5	8.7	13.59	8.29	61.45	32	Premium
6	3.89	6.25	3.14	22.3	32	Premium
7	3.84	7.5	3.67	25.93	32	Premium
8	3.81	7.45	3.64	25.71	32	Premium
9	3.83	6.72	3.65	26.45	32	Premium
10	4.13	7.48	3.94	28.36	32	Premium
11	4.18	7.15	3.99	29.08	32	Premium
12	4.15	7.36	3.96	28.61	32	Premium
13	4.13	6.67	3.94	29.01	32	Premium
14	3.97	6.91	3.79	27.52	32	Premium
15	4	7.5	3.83	27.3	32	Premium
16	3.98	11.59	7.97	59.49	32	Premium
17	3.97	9.84	7.73	45.98	32	Premium
18	3.83	9.32	5.93	48.91	32	Premium
19	4.15	10.20	5.93	46.71	32	Premium
20	3.79	7.91	6.06	50.76	32	Premium
21	4.09	9.64	3.65	23.46	32	Premium
22	3.88	10.91	5.21	49.22	32	Premium
23	3.98	7.24	6.07	35.32	32	Premium
24	4.02	9.75	6.98	51.93	32	Premium
25	3.8	11.21	5.05	28.95	32	Premium
26	3.83	6.33	5.24	47.57	32	Premium
27	3.97	10.97	4.43	48.78	32	Premium
28	4.04	8.57	7.13	28.59	32	Premium
29	3.84	7.91	5.72	43.93	32	Premium
30	3.87	6.98	4.58	44.01	32	Premium
31	3.63	8	3.48	23.8	32	Baik
32	3.56	6.71	3.41	24.28	32	Baik
33	3.53	8.39	3.38	22.64	32	Baik
34	3.48	7.13	5.21	47.07	32	Baik
35	3.52	12.81	7.76	58.57	32	Baik
36	3.64	10.64	6.33	40.79	32	Baik
37	3.53	6.5	5.07	31.13	32	Baik
38	3.64	13.20	7.57	37.23	32	Baik

39	3.68	12.57	4.93	37.96	32	Baik
40	3.58	11.42	3.98	50.73	32	Baik
41	3.67	12.14	3.64	37.36	32	Baik
42	3.52	9.77	4.2	37.39	32	Baik
43	3.65	7.67	5	30.06	32	Baik
44	3.56	12.74	6.98	45.24	32	Baik
45	3.63	10.81	4.71	35.71	32	Baik
46	3.95	7.76	3.78	26.64	34	Premium
47	3.88	5.71	3.7	27.76	34	Premium
48	3.99	7.36	3.82	27.33	34	Premium
49	3.81	7.38	3.65	25.82	34	Premium
50	4.09	9.91	3.92	26.05	34	Premium
51	3.71	7.69	3.54	24.66	34	Premium
52	4.09	6.89	3.91	28.53	34	Premium
53	3.97	7.18	3.79	27.3	34	Premium
54	3.88	6.29	3.71	27.29	34	Premium
55	4.05	6.43	3.87	28.59	34	Premium
56	4.06	7.3	3.87	27.9	34	Premium
57	3.93	7.31	3.75	26.81	34	Premium
58	4.09	12.82	3.93	23.67	34	Premium
59	3.95	6.95	3.77	27.26	34	Premium
60	3.79	6.38	3.62	26.42	34	Premium
61	3.78	8.65	3.67	28.21	34	Premium
62	3.72	8.66	3.9	24.28	34	Premium
63	3.71	12.07	3.53	27.85	34	Premium
64	3.78	9.78	3.78	28.26	34	Premium
65	3.83	7.06	3.6	28.5	34	Premium
66	3.86	5.9	3.56	24.15	34	Premium
67	3.98	11.01	3.53	23.96	34	Premium
68	4.05	11.33	3.81	25.9	34	Premium
69	3.8	12.35	3.76	23.9	34	Premium
70	3.77	8.67	3.67	26.46	34	Premium
71	3.83	7.16	3.61	24.67	34	Premium
72	3.97	7.99	3.58	26.63	34	Premium
73	3.87	7.28	3.41	27.45	34	Premium
74	4.04	8.54	3.81	26.65	34	Premium
75	3.8	6.97	3.48	27.53	34	Premium
76	3.6	8.11	3.44	23.41	34	Baik
77	3.58	7.94	3.43	23.42	34	Baik
78	3.55	6.71	3.39	24.14	34	Baik
79	3.67	8.45	3.8	28.36	34	Baik
80	3.55	11.61	3.89	27.9	34	Baik
81	3.69	8	3.43	23.46	34	Baik

82	3.64	5.97	3.91	28.4	34	Baik
83	3.45	8.66	3.81	27.16	34	Baik
84	3.53	5.86	3.72	24.32	34	Baik
85	3.63	8.95	3.39	25.12	34	Baik
86	3.69	5.78	3.78	28.27	34	Baik
87	3.57	11.22	3.88	25.13	34	Baik
88	3.54	6.63	3.93	26.61	34	Baik
89	3.63	6.51	3.67	28.24	34	Baik
90	3.69	8.83	3.67	26.52	34	Baik
91	3.92	8.39	3.75	25.89	36	Premium
92	3.81	7.05	3.64	26.04	36	Premium
93	4.54	7.03	4.33	32.16	36	Premium
94	3.87	6.2	3.69	27.25	36	Premium
95	3.71	8.28	3.55	24.19	36	Premium
96	3.84	6.77	3.66	26.5	36	Premium
97	3.85	6.7	3.68	26.69	36	Premium
98	3.92	6.57	3.74	27.32	36	Premium
99	3.87	7.49	3.7	26.2	36	Premium
100	3.79	9.09	3.63	24.2	36	Premium
101	4.24	7.49	4.05	29.25	36	Premium
102	4.2	8.01	4.01	28.5	36	Premium
103	4.29	7.32	4.09	29.81	36	Premium
104	4.21	7.33	4.02	29.18	36	Premium
105	4.1	7.68	3.91	27.92	36	Premium
106	4.05	12	3.89	24.02	36	Premium
107	4.16	7.22	3.97	28.81	36	Premium
108	4.14	7.57	3.95	28.37	36	Premium
109	4.13	10.85	3.75	31.39	36	Premium
110	3.87	6.41	3.55	24.67	36	Premium
111	4.16	8.95	3.88	28.31	36	Premium
112	4.29	9.42	4.18	30.33	36	Premium
113	3.87	10.56	4.19	25.16	36	Premium
114	3.82	8.85	3.92	26.6	36	Premium
115	4.1	7.33	4	26.56	36	Premium
116	3.79	9.23	3.83	26.9	36	Premium
117	3.733	6.59	4.12	30.63	36	Premium
118	3.747	8.31	4.19	28.35	36	Premium
119	3.72	11.36	4.09	30.85	36	Premium
120	3.78	9.75	4.12	27.67	36	Premium
121	3.63	7.16	3.87	24.22	36	Baik
122	3.58	6.95	3.68	28.24	36	Baik
123	3.42	11.21	3.59	27.05	36	Baik
124	3.45	9.83	3.82	27.72	36	Baik

125	3.42	9.61	4.13	28.68	36	Baik
126	3.41	6.98	4.09	24.8	36	Baik
127	3.5	9.59	4.17	29.15	36	Baik
128	3.49	9.87	4.21	30.69	36	Baik
129	3.52	7.21	3.96	32.05	36	Baik
130	3.49	9.1	4.17	28.18	36	Baik
131	3.52	8.02	4.24	27.83	36	Baik
132	3.66	6.51	3.83	31.67	36	Baik
133	3.61	6.64	3.65	26.73	36	Baik
134	3.68	6.32	3.68	26.58	36	Baik
135	3.56	8.49	4.32	28.5	36	Baik
136	4.32	7.48	4.12	29.93	38	Premium
137	3.91	6.87	3.73	27.01	38	Premium
138	3.83	5.95	3.66	27.15	38	Premium
139	3.78	7.93	3.62	25.12	38	Premium
140	3.75	8.33	3.59	24.52	38	Premium
141	3.97	7.26	3.8	27.25	38	Premium
142	4.3	6.19	4.1	30.82	38	Premium
143	3.82	7.27	3.65	25.94	38	Premium
144	3.78	7.35	3.62	25.59	38	Premium
145	3.74	5.72	3.57	26.59	38	Premium
146	4.13	6.13	3.94	29.45	38	Premium
147	4.1	6.03	3.91	29.32	38	Premium
148	4.18	6.47	3.99	29.64	38	Premium
149	4.28	6.87	4.08	30.09	38	Premium
150	4.25	7.9	4.06	29.03	38	Premium
151	4.2	11.83	4.04	25.43	38	Premium
152	3.96	7.96	3.79	26.56	38	Premium
153	4.18	5.99	3.99	30.02	38	Premium
154	3.785	6.17	3.89	28.08	38	Premium
155	3.749	9.04	3.63	27.25	38	Premium
156	3.794	10.58	3.63	24.66	38	Premium
157	3.76	9.42	3.73	26.51	38	Premium
158	3.76	9.66	3.66	25.15	38	Premium
159	3.88	6.1	3.93	25.37	38	Premium
160	4.1	6.62	3.94	28.26	38	Premium
161	3.96	6.76	3.95	29.02	38	Premium
162	3.74	10.12	4.08	29.56	38	Premium
163	3.86	8.55	3.58	26.82	38	Premium
164	4.03	8.24	4	28.86	38	Premium
165	3.73	7.69	4.04	28.79	38	Premium
166	3.41	9.61	3.76	29.46	38	Baik
167	3.45	8.81	3.94	26.56	38	Baik

168	3.56	11.59	3.63	28.56	38	Baik
169	3.65	10.51	4.11	26.45	38	Baik
170	3.68	10.28	3.59	26.15	38	Baik
171	3.41	10.26	3.61	29.85	38	Baik
172	3.42	8.54	3.78	29.87	38	Baik
173	3.54	11.75	3.91	25.75	38	Baik
174	3.56	11.05	4.06	25.77	38	Baik
175	3.44	6.04	3.74	30.19	38	Baik
176	3.42	8.88	3.74	26.36	38	Baik
177	3.65	7.66	3.73	30.6	38	Baik
178	3.69	9.33	3.76	29.98	38	Baik
179	3.57	8.05	3.7	28.28	38	Baik
180	3.55	5.78	4.03	28.88	38	Baik
181	3.95	7.49	3.77	26.86	40	Premium
182	3.95	6.45	8.77	27.69	40	Premium
183	3.94	6.51	3.76	27.55	40	Premium
184	3.91	9.26	3.75	26.09	40	Premium
185	3.93	6.83	3.75	27.24	40	Premium
186	3.9	4.93	3.71	28.51	40	Premium
187	3.77	7.5	3.6	25.35	40	Premium
188	3.79	7.4	3.62	25.58	40	Premium
189	3.86	7.54	3.69	26.11	40	Premium
190	3.76	7.82	3.6	25.04	40	Premium
191	3.84	6.79	3.67	26.51	40	Premium
192	4.03	6.5	3.84	28.31	40	Premium
193	3.97	6.77	3.78	27.57	40	Premium
194	4.2	5.74	4	30.39	40	Premium
195	4.06	11.67	3.9	24.34	40	Premium
196	4.12	7.53	3.94	28.25	40	Premium
197	4.04	5.85	3.85	28.92	40	Premium
198	3.87	5.79	8.18	23.13	40	Premium
199	3.96	9.2	8.28	26.6	40	Premium
200	4.05	6.58	5.31	23.92	40	Premium
201	3.78	11.31	8.55	28.8	40	Premium
202	4.12	11.19	5.83	24.04	40	Premium
203	3.74	9.89	8.36	27.59	40	Premium
204	3.76	9.3	6.45	25.21	40	Premium
205	3.91	9.28	6.40	27.18	40	Premium
206	3.754	4.96	8.47	26.02	40	Premium
207	3.809	8.19	5.17	29.26	40	Premium
208	3.776	10.83	3.39	28.09	40	Premium
209	4.074	8.47	4.34	24.63	40	Premium
210	4.099	6.99	7.31	28.35	40	Premium

211	3.41	6.72	3.26	22.96	40	Baik
212	3.41	5.38	7.76	28.93	40	Baik
213	3.62	10	3.5	23.95	40	Baik
214	3.63	9.13	4.17	25	40	Baik
215	3.64	5.31	7.4	25.79	40	Baik
216	3.45	10.94	5.94	23.61	40	Baik
217	3.65	5.72	7.46	28.73	40	Baik
218	3.48	5.95	3.99	23.73	40	Baik
219	3.64	7.07	5.48	27.61	40	Baik
220	3.69	8.4	7.46	23.9	40	Baik
221	3.492	6.91	4.18	26.02	40	Baik
222	3.486	7.84	6.85	29.38	40	Baik
223	3.6	10.12	7.49	24.23	40	Baik
224	3.625	7.93	6.89	24.67	40	Baik
225	3.462	5.93	5.23	29.76	40	Baik

LAMPIRAN D BOBOT DAN PERUBAHAN BOBOT PADA MANUALISASI

ITERASI PERTAMA

Bobot *Input Layer* ke *Hidden Layer*

V11	V21	V31	V41	V12	V22	V32	V42	V13	V23	V33	V43	V14	V24	V34	V44
0.7105	-0.5918	-0.6764	-0.1384	0.2525	-0.0935	0.1381	-0.7931	-0.0664	-0.3608	0.7026	-0.6888	0.4017	-0.5565	-0.0508	-0.3314
0.7106	-0.5917	-0.6763	-0.1383	0.2547	-0.0922	0.1394	-0.7920	-0.0669	-0.3611	0.7023	-0.6890	0.4020	-0.5563	-0.0506	-0.3313
0.7085	-0.5921	-0.6782	-0.1392	0.2547	-0.0922	0.1393	-0.7920	-0.0678	-0.3613	0.7015	-0.6894	0.4005	-0.5566	-0.0520	-0.3319
0.6981	-0.5966	-0.6911	-0.1434	0.2448	-0.0964	0.1270	-0.7960	-0.0744	-0.3641	0.6933	-0.6921	0.3944	-0.5593	-0.0596	-0.3344
0.6889	-0.6092	-0.6962	-0.1561	0.2390	-0.1043	0.1238	-0.8039	-0.0793	-0.3708	0.6905	-0.6988	0.3883	-0.5675	-0.0629	-0.3426
0.6896	-0.6042	-0.6937	-0.1518	0.2400	-0.0975	0.1274	-0.7980	-0.0789	-0.3679	0.6920	-0.6963	0.3889	-0.5631	-0.0607	-0.3388
0.6584	-0.6291	-0.7146	-0.1582	0.2097	-0.1216	0.1070	-0.8042	-0.1054	-0.3890	0.6742	-0.7017	0.3596	-0.5865	-0.0804	-0.3448
0.6427	-0.6308	-0.7272	-0.1663	0.1965	-0.1230	0.0965	-0.8111	-0.1144	-0.3900	0.6670	-0.7064	0.3502	-0.5875	-0.0880	-0.3498
0.6403	-0.6368	-0.7366	-0.1681	0.1943	-0.1285	0.0879	-0.8127	-0.1158	-0.3935	0.6616	-0.7075	0.3486	-0.5912	-0.0938	-0.3509

Bobot *Hidden Layer* ke *Output Layer*

W11	W21	W31	W41	W12	W22	W32	W42	W13	W23	W33	W43	W14	W24	W34	W44
0.5716	0.4962	0.1278	0.1989	0.9383	0.6603	0.7164	0.7722	0.3699	0.7822	0.1866	0.7479	0.5031	0.3135	0.8484	0.3829
0.5847	0.5092	0.1403	0.2119	0.9228	0.6450	0.7016	0.7569	0.3796	0.7918	0.1959	0.7575	0.5105	0.3209	0.8555	0.3903
0.5916	0.5163	0.1479	0.2192	0.9054	0.6269	0.6823	0.7382	0.3856	0.7981	0.2025	0.7640	0.5196	0.3303	0.8656	0.4000
0.5607	0.4787	0.1063	0.1830	0.9092	0.6314	0.6873	0.7426	0.3798	0.7910	0.1947	0.7572	0.5053	0.3128	0.8463	0.3832
0.5537	0.4722	0.1005	0.1764	0.8840	0.6083	0.6664	0.7190	0.3834	0.7943	0.1977	0.7605	0.5102	0.3173	0.8503	0.3877
0.5680	0.4887	0.1174	0.1919	0.8787	0.6022	0.6601	0.7133	0.3932	0.8056	0.2094	0.7711	0.5169	0.3251	0.8584	0.3951
0.5450	0.4621	0.0922	0.1678	0.8465	0.5648	0.6246	0.6795	0.3582	0.7650	0.1708	0.7343	0.4816	0.2842	0.8194	0.3580

0.5115	0.4297	0.0572	0.1325	0.8375	0.5561	0.6152	0.6699	0.3521	0.7591	0.1644	0.7279	0.4700	0.2729	0.8073	0.3457
0.4827	0.3914	0.0154	0.0979	0.8302	0.5464	0.6046	0.6612	0.3467	0.7519	0.1565	0.7214	0.4604	0.2601	0.7933	0.3342

Perubahan Bobot dari *Hidden Layer* ke *Output Layer*

ΔW11	ΔW21	ΔW31	ΔW41	ΔW12	ΔW22	ΔW32	ΔW42	ΔW13	ΔW23	ΔW33	ΔW43	ΔW14	ΔW24	ΔW34	ΔW44
0.0131	0.0130	0.0125	0.0130	-0.0155	-0.0153	-0.0148	-0.0153	0.0097	0.0096	0.0093	0.0096	0.0074	0.0074	0.0071	0.0074
0.0069	0.0071	0.0076	0.0074	-0.0174	-0.0181	-0.0194	-0.0187	0.0060	0.0062	0.0067	0.0064	0.0091	0.0094	0.0101	0.0097
-0.0309	-0.0376	-0.0416	-0.0363	0.0037	0.0045	0.0050	0.0044	-0.0058	-0.0071	-0.0078	-0.0068	-0.0143	-0.0175	-0.0193	-0.0168
-0.0070	-0.0064	-0.0058	-0.0065	-0.0251	-0.0231	-0.0209	-0.0235	0.0036	0.0033	0.0030	0.0034	0.0049	0.0045	0.0041	0.0046
0.0143	0.0164	0.0170	0.0154	-0.0053	-0.0061	-0.0063	-0.0057	0.0098	0.0113	0.0116	0.0106	0.0068	0.0078	0.0081	0.0073
-0.0229	-0.0266	-0.0253	-0.0241	-0.0322	-0.0374	-0.0355	-0.0338	-0.0350	-0.0406	-0.0386	-0.0368	-0.0353	-0.0410	-0.0389	-0.0371
-0.0335	-0.0324	-0.0349	-0.0353	-0.0090	-0.0087	-0.0094	-0.0095	-0.0061	-0.0059	-0.0064	-0.0064	-0.0116	-0.0112	-0.0121	-0.0122
-0.0288	-0.0383	-0.0419	-0.0345	-0.0073	-0.0097	-0.0106	-0.0087	-0.0054	-0.0072	-0.0079	-0.0065	-0.0097	-0.0128	-0.0140	-0.0116
-0.0364	-0.0352	-0.0335	-0.0354	-0.0271	-0.0262	-0.0250	-0.0263	-0.0076	-0.0074	-0.0071	-0.0074	-0.0076	-0.0073	-0.0070	-0.0074

Perubahan Bobot dari *Hidden Layer* ke *Input Layer*

ΔV11	ΔV21	ΔV31	ΔV41	ΔV12	ΔV22	ΔV32	ΔV42	ΔV13	ΔV23	ΔV33	ΔV43	ΔV14	ΔV24	ΔV34	ΔV44
0.0001	0.0001	0.0001	0.0001	0.0022	0.0013	0.0013	0.0011	-0.0005	-0.0003	-0.0003	-0.0002	0.0003	0.0002	0.0002	0.0001
-0.0021	-0.0004	-0.0019	-0.0009	0.0000	0.0000	0.0000	0.0000	-0.0010	-0.0002	-0.0009	-0.0004	-0.0015	-0.0003	-0.0013	-0.0006
-0.0103	-0.0044	-0.0128	-0.0042	-0.0099	-0.0043	-0.0123	-0.0040	-0.0066	-0.0028	-0.0082	-0.0027	-0.0061	-0.0026	-0.0076	-0.0025
-0.0093	-0.0126	-0.0052	-0.0126	-0.0058	-0.0079	-0.0032	-0.0079	-0.0049	-0.0067	-0.0027	-0.0067	-0.0060	-0.0082	-0.0034	-0.0082
0.0007	0.0050	0.0026	0.0043	0.0010	0.0068	0.0035	0.0059	0.0004	0.0029	0.0015	0.0025	0.0006	0.0044	0.0022	0.0038
-0.0312	-0.0248	-0.0210	-0.0064	-0.0302	-0.0241	-0.0203	-0.0062	-0.0265	-0.0211	-0.0178	-0.0054	-0.0293	-0.0233	-0.0197	-0.0060
-0.0156	-0.0017	-0.0125	-0.0082	-0.0132	-0.0015	-0.0106	-0.0069	-0.0089	-0.0010	-0.0072	-0.0047	-0.0095	-0.0011	-0.0076	-0.0050
-0.0025	-0.0060	-0.0094	-0.0018	-0.0022	-0.0055	-0.0086	-0.0016	-0.0014	-0.0035	-0.0055	-0.0010	-0.0015	-0.0037	-0.0058	-0.0011

-0.0085	-0.0036	-0.0023	-0.0023	-0.0068	-0.0029	-0.0018	-0.0018	-0.0044	-0.0019	-0.0012	-0.0012	-0.0054	-0.0023	-0.0014	-0.0014

ITERASI KEDUA

Bobot *Input Layer* ke *Hidden Layer*

V11	V21	V31	V41	V12	V22	V32	V42	V13	V23	V33	V43	V14	V24	V34	V44
0.6403	-0.6368	-0.7366	-0.1681	0.1943	-0.1285	0.0879	-0.8127	-0.1158	-0.3935	0.6616	-0.7075	0.3486	-0.5912	-0.0938	-0.3509
0.6416	-0.6360	-0.7358	-0.1674	0.1976	-0.1265	0.0898	-0.8110	-0.1157	-0.3935	0.6616	-0.7074	0.3496	-0.5907	-0.0932	-0.3504
0.6411	-0.6361	-0.7362	-0.1676	0.1991	-0.1262	0.0911	-0.8104	-0.1157	-0.3935	0.6616	-0.7074	0.3491	-0.5907	-0.0937	-0.3506
0.6336	-0.6394	-0.7456	-0.1707	0.1918	-0.1293	0.0821	-0.8133	-0.1198	-0.3952	0.6565	-0.7091	0.3456	-0.5923	-0.0980	-0.3520
0.6265	-0.6491	-0.7495	-0.1804	0.1881	-0.1345	0.0801	-0.8184	-0.1230	-0.3996	0.6547	-0.7135	0.3414	-0.5980	-0.1004	-0.3578
0.6273	-0.6434	-0.7466	-0.1755	0.1891	-0.1272	0.0838	-0.8122	-0.1226	-0.3966	0.6563	-0.7109	0.3421	-0.5934	-0.0980	-0.3538
0.5988	-0.6660	-0.7657	-0.1813	0.1619	-0.1489	0.0655	-0.8177	-0.1463	-0.4155	0.6404	-0.7157	0.3154	-0.6146	-0.1159	-0.3592
0.5862	-0.6674	-0.7758	-0.1879	0.1518	-0.1500	0.0574	-0.8230	-0.1523	-0.4161	0.6356	-0.7189	0.3088	-0.6153	-0.1212	-0.3627
0.5842	-0.6723	-0.7834	-0.1894	0.1501	-0.1543	0.0508	-0.8243	-0.1533	-0.4185	0.6319	-0.7196	0.3077	-0.6180	-0.1253	-0.3635

Bobot *Hidden Layer* ke *Output Layer*

W11	W21	W31	W41	W12	W22	W32	W42	W13	W23	W33	W43	W14	W24	W34	W44
0.4827	0.3914	0.0154	0.0979	0.8302	0.5464	0.6046	0.6612	0.3467	0.7519	0.1565	0.7214	0.4604	0.2601	0.7933	0.3342
0.4993	0.4080	0.0315	0.1146	0.8154	0.5316	0.5902	0.6463	0.3574	0.7626	0.1669	0.7322	0.4695	0.2692	0.8021	0.3433
0.5098	0.4191	0.0434	0.1261	0.7981	0.5134	0.5706	0.6275	0.3644	0.7700	0.1749	0.7399	0.4803	0.2806	0.8144	0.3551
0.4822	0.3847	0.0048	0.0928	0.8043	0.5212	0.5793	0.6350	0.3598	0.7642	0.1684	0.7343	0.4674	0.2646	0.7965	0.3397
0.4776	0.3804	0.0009	0.0884	0.7808	0.4993	0.5594	0.6126	0.3641	0.7683	0.1721	0.7384	0.4733	0.2701	0.8015	0.3453
0.4933	0.3987	0.0199	0.1056	0.7769	0.4947	0.5546	0.6083	0.3742	0.7801	0.1843	0.7495	0.4809	0.2789	0.8106	0.3535
0.4733	0.3751	-0.0026	0.0843	0.7442	0.4561	0.5178	0.5733	0.3408	0.7405	0.1466	0.7137	0.4471	0.2390	0.7726	0.3174
0.4431	0.3456	-0.0347	0.0519	0.7369	0.4490	0.5101	0.5655	0.3358	0.7356	0.1413	0.7083	0.4370	0.2292	0.7619	0.3066

0.4164	0.3097	-0.0741	0.0195	0.7309	0.4408	0.5011	0.5582	0.3309	0.7291	0.1342	0.7025	0.4283	0.2174	0.7490	0.2960

Perubahan Bobot dari *Hidden Layer* ke *Output Layer*

ΔW11	ΔW21	ΔW31	ΔW41	ΔW12	ΔW22	ΔW32	ΔW42	ΔW13	ΔW23	ΔW33	ΔW43	ΔW14	ΔW24	ΔW34	ΔW44
0.0166	0.0166	0.0161	0.0167	-0.0148	-0.0148	-0.0144	-0.0149	0.0107	0.0107	0.0104	0.0108	0.0091	0.0091	0.0088	0.0091
0.0105	0.0111	0.0120	0.0115	-0.0172	-0.0182	-0.0196	-0.0188	0.0070	0.0074	0.0080	0.0077	0.0108	0.0114	0.0123	0.0118
-0.0276	-0.0345	-0.0386	-0.0333	0.0062	0.0077	0.0087	0.0075	-0.0047	-0.0058	-0.0065	-0.0056	-0.0128	-0.0160	-0.0179	-0.0154
-0.0046	-0.0043	-0.0039	-0.0044	-0.0235	-0.0219	-0.0199	-0.0224	0.0043	0.0040	0.0037	0.0041	0.0059	0.0055	0.0050	0.0056
0.0157	0.0183	0.0190	0.0172	-0.0039	-0.0046	-0.0047	-0.0043	0.0101	0.0118	0.0122	0.0111	0.0075	0.0088	0.0091	0.0083
-0.0200	-0.0236	-0.0225	-0.0213	-0.0327	-0.0386	-0.0368	-0.0349	-0.0335	-0.0396	-0.0377	-0.0358	-0.0338	-0.0399	-0.0380	-0.0361
-0.0301	-0.0295	-0.0321	-0.0324	-0.0073	-0.0071	-0.0077	-0.0078	-0.0050	-0.0049	-0.0053	-0.0054	-0.0100	-0.0098	-0.0107	-0.0108
-0.0267	-0.0359	-0.0395	-0.0324	-0.0061	-0.0081	-0.0089	-0.0073	-0.0048	-0.0065	-0.0071	-0.0059	-0.0088	-0.0118	-0.0129	-0.0106
-0.0340	-0.0329	-0.0315	-0.0332	-0.0276	-0.0268	-0.0256	-0.0270	-0.0070	-0.0068	-0.0065	-0.0069	-0.0065	-0.0063	-0.0060	-0.0063

Perubahan Bobot dari *Hidden Layer* ke *Input Layer*

ΔV11	ΔV21	ΔV31	ΔV41	ΔV12	ΔV22	ΔV32	ΔV42	ΔV13	ΔV23	ΔV33	ΔV43	ΔV14	ΔV24	ΔV34	ΔV44
0.0014	0.0008	0.0008	0.0007	0.0033	0.0020	0.0019	0.0017	0.0001	0.0000	0.0000	0.0000	0.0010	0.0006	0.0006	0.0005
-0.0005	-0.0001	-0.0005	-0.0002	0.0015	0.0003	0.0013	0.0006	0.0000	0.0000	0.0000	0.0000	-0.0005	-0.0001	-0.0004	-0.0002
-0.0075	-0.0032	-0.0093	-0.0031	-0.0072	-0.0031	-0.0090	-0.0029	-0.0041	-0.0018	-0.0051	-0.0017	-0.0035	-0.0015	-0.0044	-0.0014
-0.0071	-0.0097	-0.0040	-0.0097	-0.0038	-0.0051	-0.0021	-0.0051	-0.0032	-0.0044	-0.0018	-0.0044	-0.0042	-0.0057	-0.0023	-0.0057
0.0008	0.0057	0.0029	0.0049	0.0010	0.0073	0.0038	0.0063	0.0004	0.0030	0.0015	0.0026	0.0007	0.0046	0.0024	0.0040
-0.0285	-0.0227	-0.0191	-0.0058	-0.0272	-0.0217	-0.0183	-0.0056	-0.0237	-0.0189	-0.0159	-0.0048	-0.0267	-0.0212	-0.0179	-0.0055
-0.0126	-0.0014	-0.0101	-0.0066	-0.0101	-0.0011	-0.0081	-0.0053	-0.0060	-0.0007	-0.0048	-0.0031	-0.0066	-0.0007	-0.0053	-0.0035
-0.0020	-0.0049	-0.0076	-0.0014	-0.0017	-0.0043	-0.0067	-0.0013	-0.0010	-0.0023	-0.0036	-0.0007	-0.0011	-0.0026	-0.0041	-0.0008
-0.0074	-0.0031	-0.0020	-0.0020	-0.0055	-0.0023	-0.0014	-0.0014	-0.0032	-0.0013	-0.0008	-0.0008	-0.0043	-0.0018	-0.0011	-0.0011

LAMPIRAN E HASIL PENGUJIAN SKENARIO UNTUK KUALITAS SUSU PREMIUM

1. Hasil Pengujian Terhadap Jumlah Iterasi

Jumlah Iterasi	Nilai MSE Percobaan ke-*i*										Rata-rata MSE
	1	2	3	4	5	6	7	8	9	10	
100	0.019376	0.019813	0.019546	0.019675	0.019244	0.019460	0.019504	0.020279	0.019258	0.019466	0.019562
200	0.018857	0.019163	0.019857	0.018659	0.018832	0.019405	0.018983	0.018987	0.019276	0.019089	0.019111
300	0.018518	0.018789	0.018896	0.018700	0.018743	0.018609	0.018642	0.018977	0.018577	0.018886	0.018734
400	0.018365	0.018106	0.018362	0.018259	0.018845	0.018206	0.018428	0.018747	0.018737	0.019327	0.018538
500	0.017918	0.018573	0.018521	0.018462	0.018086	0.018283	0.018798	0.018721	0.018460	0.018724	0.018455
600	0.017813	0.018172	0.018097	0.018083	0.018235	0.018727	0.018239	0.018784	0.018062	0.018498	0.018271
700	0.017989	0.017755	0.018435	0.018149	0.017936	0.017836	0.018186	0.018553	0.017965	0.018666	0.018147
800	0.018032	0.017926	0.018255	0.018211	0.018197	0.018533	0.018118	0.018269	0.018191	0.017703	0.018144
900	0.018214	0.017796	0.017676	0.018232	0.018168	0.017974	0.018330	0.017584	0.018100	0.018307	0.018038
1000	0.017644	0.017848	0.018018	0.017683	0.018218	0.018255	0.017906	0.018323	0.018103	0.018278	0.018028

2. Hasil Pengujian Terhadap Jumlah *Hidden Layer*

Hidden Layer	Nilai MSE Percobaan ke-*i*										Rata-rata MSE
	1	2	3	4	5	6	7	8	9	10	
1	0.019544	0.019540	0.019562	0.019534	0.019531	0.019575	0.019533	0.019562	0.019505	0.019561	0.019545
2	0.019672	0.018961	0.019633	0.018881	0.017971	0.017960	0.019661	0.017975	0.017988	0.017935	0.018664
3	0.018426	0.018836	0.018391	0.018191	0.017973	0.018631	0.018389	0.018386	0.018448	0.018398	0.018407
4	0.017917	0.017629	0.018088	0.018281	0.017786	0.018113	0.018310	0.017998	0.018248	0.018757	0.018113
5	0.018307	0.017975	0.018476	0.018100	0.018451	0.018323	0.018248	0.017849	0.017908	0.017923	0.018156

6	0.017986	0.017851	0.017510	0.017927	0.017877	0.017607	0.017964	0.017732	0.017983	0.018257	0.017869
7	0.017412	0.017923	0.018194	0.018274	0.017580	0.017459	0.018245	0.018044	0.017858	0.017967	0.017896
8	0.017696	0.017348	0.017688	0.017916	0.017788	0.017482	0.018073	0.017750	0.017953	0.017963	0.017766
9	0.018158	0.017507	0.017932	0.017836	0.017863	0.017537	0.017599	0.017523	0.017888	0.017386	0.017723
10	0.017731	0.017567	0.017458	0.017794	0.017494	0.017723	0.017489	0.017514	0.017308	0.017549	0.017563

3. Hasil Pengujian Terhadap Nilai *Learning Rate*

Learning Rate	**Nilai MSE Percobaan ke-*i***										**Rata-rata MSE**
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	
0.9	0.017329	0.016435	0.017163	0.016042	0.017606	0.017313	0.016601	0.016348	0.015914	0.017189	0.016794
0.8	0.017384	0.016642	0.016849	0.017109	0.016927	0.017625	0.016899	0.017561	0.016968	0.017277	0.017124
0.7	0.017498	0.017452	0.017861	0.016208	0.017538	0.017010	0.017883	0.017332	0.017576	0.016143	0.017250
0.6	0.017582	0.017403	0.017153	0.017800	0.017865	0.017618	0.017627	0.017614	0.017661	0.017277	0.017560
0.5	0.017964	0.017637	0.017820	0.017689	0.017946	0.017730	0.017882	0.017867	0.017976	0.017941	0.017845
0.4	0.018370	0.018120	0.018136	0.018008	0.018073	0.018127	0.018099	0.017688	0.017852	0.017975	0.018045
0.3	0.018042	0.018205	0.018491	0.018272	0.018460	0.018354	0.018277	0.018241	0.017954	0.018222	0.018252
0.2	0.018695	0.018235	0.018384	0.018297	0.018713	0.017925	0.018609	0.018387	0.018715	0.018700	0.018466
0.1	0.018766	0.018675	0.018834	0.018681	0.018640	0.018686	0.018720	0.018656	0.018805	0.018598	0.018706

4. Hasil Pengujian Terhadap Ukuran Populasi

Ukuran Populasi	**Nilai *Fitness* Percobaan ke-*i***										**Rata-rata *Fitness***
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	
10	4.220325	4.173376	4.106050	4.057841	3.801884	4.251835	4.217390	3.790787	4.262006	4.545662	4.142716
20	4.146335	4.033847	4.110849	4.192945	4.039047	4.203107	4.702044	3.977754	4.636667	4.601797	4.264439
30	4.131281	4.144516	4.214139	4.383748	4.624329	4.581078	4.509463	4.265803	4.100955	4.356190	4.331150

40	4.091596	4.326526	4.326526	3.888956	4.686004	4.680421	4.058361	4.367432	4.659302	4.450668	4.353579
50	3.893534	4.461145	4.342869	4.459406	4.774980	4.657652	3.944664	4.162671	4.540691	4.451049	4.368866
60	4.729255	4.168619	4.561719	4.270726	4.306982	4.511433	4.247769	4.366170	4.252379	4.289862	4.370491
70	4.169507	4.453774	4.669192	4.336402	4.569662	4.315753	4.219770	4.281283	4.185565	4.510714	4.371162
80	4.199015	4.501404	4.146713	4.527381	4.333559	4.339147	4.257757	4.569190	4.406204	4.443273	4.372364
90	4.755224	4.359783	4.126360	4.380889	4.113914	4.641645	4.179603	4.605935	4.289470	4.300174	4.375300
100	4.226962	4.231873	4.253044	4.290998	4.455224	4.231696	4.762743	4.175035	4.791820	4.377897	4.379729

5. Hasil Pengujian Terhadap Jumlah Generasi

Jumlah Generasi	Nilai *Fitness* Percobaan ke-*i*										Rata-rata *Fitness*
	1	2	3	4	5	6	7	8	9	10	
100	4.034184	4.290529	4.494060	4.403252	4.238421	4.017008	3.888891	4.129195	4.180135	4.237753	4.191343
200	4.498253	4.240378	4.240290	4.291205	4.304455	4.327772	3.967574	4.222457	3.950889	4.012021	4.205530
300	4.097739	4.459041	3.945637	4.206625	4.002743	4.195528	4.758003	4.334150	4.075920	4.309848	4.238523
400	4.563761	4.149908	4.149908	3.851484	4.691400	3.906456	3.962022	4.724022	4.442599	4.261142	4.270270
500	4.510282	4.443940	4.393057	4.234592	4.047834	4.457358	4.185855	4.170322	4.183147	4.280651	4.290704
600	4.157067	4.191511	4.278015	4.231479	4.058792	4.158793	4.247730	4.619020	4.516309	4.726597	4.318531
700	4.322783	4.271945	4.149149	4.460232	4.232980	4.073048	4.210585	4.324557	4.708484	4.544034	4.329780
800	4.473616	4.293715	4.331274	4.047095	4.300341	4.191559	4.539440	4.244511	4.253266	4.677269	4.335209
900	4.267321	4.541325	4.246665	4.500991	4.342121	4.117206	4.352241	4.328741	4.187744	4.607851	4.349221
1000	4.430069	4.790773	4.654969	4.255362	4.356332	4.321792	4.259523	3.844439	4.176846	4.421393	4.351150

6. Hasil Pengujian Terhadap Kombinasi *Crossover Rate* dan *Mutation Rate*

cr	*mr*	Nilai *Fitness* Percobaan ke-*i*										Rata-rata *Fitness*
		1	2	3	4	5	6	7	8	9	10	
1	0	4.086611	3.984560	3.889733	4.581630	4.048835	3.927298	4.018530	4.434166	4.200000	4.326498	4.149786
0.9	0.1	4.326807	4.186959	4.485570	4.480919	4.236743	4.208082	3.941070	4.459644	3.908071	4.359140	4.259300
0.8	0.2	4.562964	3.974613	4.513923	4.239675	4.024009	4.045732	3.832906	4.229017	4.501858	4.272020	4.219672
0.7	0.3	4.429833	4.430973	3.911252	3.911359	3.897835	4.406692	4.011805	4.200210	4.012723	4.012381	4.122506
0.6	0.4	4.437079	4.038598	3.850479	4.026796	4.167304	4.250644	4.232075	4.370855	4.663282	4.445610	4.248272
0.5	0.5	4.021439	4.455867	4.003609	4.203105	4.238240	4.117349	4.143011	4.011919	3.933959	3.981198	4.110970
0.4	0.6	3.981198	3.981199	3.812514	4.174180	4.174496	4.264175	4.173736	4.303493	4.225014	4.145922	4.123593
0.3	0.7	4.240663	4.182389	4.539156	4.133776	4.207433	4.438531	4.189000	4.140856	3.938205	4.439150	4.244916
0.2	0.8	4.363322	4.468750	3.924078	4.530916	4.450652	4.048972	3.955002	4.304096	4.307600	4.603837	4.295722
0.1	0.9	4.295721	4.032325	4.635006	4.590088	4.359140	4.274646	4.208418	4.258745	4.238776	4.348067	4.324093

7. Hasil Pengujian Terhadap Kombinasi Variasi *Output* yang Dioptimalkan

Kandungan Gizi Susu		Nilai *Fitness* Percobaan ke-*i*										Rata-rata *Fitness*
Optimalkan	Minimalkan	1	2	3	4	5	6	7	8	9	10	
Protein	Lemak	4.10605	4.15900	4.14674	4.24830	4.07855	4.25133	4.25131	4.25137	4.25136	4.11450	4.18585

Kasus Optimasi (Protein Optimal dan Lemak Minimal)	Nilai Presentase Percobaan ke-*i*										Rata-rata Presentase
	1	2	3	4	5	6	7	8	9	10	
Protein Dioptimalkan	0.0026	0.0468	0.0481	0.1341	0.0033	0.1531	0.1529	0.1529	0.1529	0.0054	0.0852
Lemak Diminimalkan	2.0105	1.4391	2.4082	1.3504	4.6682	2.4400	2.4259	2.4212	2.4224	1.6685	2.3254

LAMPIRAN F HASIL PENGUJIAN SKENARIO UNTUK KUALITAS SUSU BAIK

1. Hasil Pengujian Terhadap Jumlah Iterasi

Jumlah Iterasi	Nilai MSE Percobaan ke-*i*										Rata-rata MSE
	1	2	3	4	5	6	7	8	9	10	
100	0.008685	0.009055	0.009164	0.008977	0.008767	0.009312	0.009196	0.009153	0.010377	0.009176	0.009186
200	0.009356	0.009364	0.008862	0.009341	0.009201	0.009670	0.008597	0.009178	0.009308	0.008559	0.009144
300	0.009042	0.008340	0.009174	0.008549	0.008719	0.009002	0.009164	0.009081	0.008973	0.009773	0.008982
400	0.009318	0.008404	0.008815	0.008120	0.008789	0.007749	0.008667	0.008939	0.008110	0.009055	0.008596
500	0.008287	0.007843	0.008065	0.009165	0.009667	0.008494	0.008158	0.008234	0.007898	0.007897	0.008371
600	0.007920	0.008388	0.008694	0.008278	0.007983	0.007668	0.008007	0.007839	0.008093	0.008279	0.008115
700	0.008070	0.007477	0.007844	0.008361	0.007628	0.007919	0.008511	0.008208	0.007672	0.008967	0.008066
800	0.007502	0.008154	0.008258	0.008603	0.007859	0.007923	0.007440	0.008077	0.007514	0.008291	0.007962
900	0.007582	0.008104	0.007407	0.007474	0.007924	0.007189	0.008272	0.007287	0.007587	0.007368	0.007620
1000	0.006709	0.007886	0.007100	0.008129	0.007499	0.007775	0.006856	0.007594	0.007191	0.007161	0.007390

2. Hasil Pengujian Terhadap Jumlah *Hidden Layer*

Hidden Layer	Nilai MSE Percobaan ke-*i*										Rata-rata MSE
	1	2	3	4	5	6	7	8	9	10	
1	0.012119	0.009365	0.009308	0.009333	0.012124	0.009230	0.012098	0.012152	0.009324	0.009225	0.010428
2	0.008936	0.009325	0.008959	0.008829	0.008960	0.008398	0.008954	0.008976	0.008955	0.008943	0.008923
3	0.008097	0.007969	0.008297	0.008356	0.007774	0.009169	0.008061	0.008643	0.008164	0.008932	0.008346
4	0.006633	0.007556	0.007073	0.007510	0.007666	0.007620	0.007312	0.007720	0.007624	0.007310	0.007402
5	0.007140	0.006925	0.007609	0.007179	0.007168	0.007052	0.007357	0.006890	0.007297	0.007338	0.007196

6	0.006329	0.007103	0.006866	0.007217	0.007581	0.007364	0.007388	0.007597	0.007568	0.006931	0.007194
7	0.007010	0.006867	0.006377	0.006467	0.006676	0.006419	0.007057	0.007130	0.007007	0.007285	0.006829
8	0.006664	0.006812	0.007468	0.006752	0.006848	0.007234	0.006909	0.005939	0.006798	0.006326	0.006775
9	0.007253	0.006497	0.006694	0.006764	0.006811	0.006259	0.006392	0.005573	0.007393	0.007367	0.006700
10	0.006579	0.006077	0.005798	0.006747	0.007362	0.007089	0.007090	0.005648	0.006210	0.006757	0.006536

3. Hasil Pengujian Terhadap Nilai *Learning Rate*

Learning Rate	**Nilai MSE Percobaan ke-*i***										**Rata-rata MSE**
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	
0.9	0.006172	0.005050	0.007214	0.005546	0.005573	0.005772	0.004979	0.005344	0.005997	0.005652	0.005730
0.8	0.006528	0.005306	0.004491	0.005695	0.005556	0.005481	0.005776	0.006742	0.005946	0.007006	0.005853
0.7	0.005772	0.006635	0.006729	0.006415	0.006583	0.006842	0.006064	0.006656	0.007213	0.006790	0.006570
0.6	0.006325	0.007014	0.006877	0.007052	0.006757	0.006793	0.006828	0.006953	0.007047	0.007141	0.006879
0.5	0.007121	0.007392	0.006673	0.007386	0.007167	0.007318	0.007357	0.007174	0.007522	0.007188	0.007230
0.4	0.007844	0.007377	0.007084	0.007620	0.007028	0.007486	0.007726	0.007256	0.008020	0.007362	0.007480
0.3	0.007914	0.007741	0.007938	0.008133	0.007452	0.007950	0.007991	0.008134	0.007831	0.007787	0.007887
0.2	0.008544	0.008252	0.008478	0.008226	0.008198	0.008516	0.008292	0.008279	0.008193	0.008594	0.008357
0.1	0.008902	0.008602	0.008222	0.008708	0.008410	0.008569	0.008498	0.008440	0.008790	0.008097	0.008524

4. Hasil Pengujian Terhadap Ukuran Populasi

Ukuran Populasi	**Nilai *Fitness* Percobaan ke-*i***										**Rata-rata *Fitness***
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	
10	3.582212	3.652173	3.611053	3.603994	3.617225	3.566266	3.571551	3.609015	3.558394	3.607493	3.597937
20	3.551962	3.602705	3.617749	3.620470	3.592193	3.596163	3.677296	3.555574	3.675548	3.560069	3.604973
30	3.689914	3.594512	3.638918	3.703484	3.578322	3.731890	3.560018	3.531980	3.531817	3.546191	3.610705

40	3.566010	3.628930	3.543811	3.661443	3.615617	3.593174	3.601929	3.721338	3.548364	3.673091	3.615371
50	3.582368	3.641299	3.628920	3.632861	3.573712	3.595771	3.617082	3.568538	3.592347	3.665082	3.609798
60	3.641697	3.716256	3.696576	3.632798	3.557308	3.670039	3.525328	3.500622	3.510551	3.652658	3.610383
70	3.550566	3.548526	3.614645	3.581097	3.556106	3.710849	3.591109	3.581518	3.708438	3.669216	3.611207
80	3.697429	3.587013	3.637125	3.596450	3.755032	3.694466	3.594855	3.537463	3.595021	3.636683	3.633154
90	3.755986	3.619655	3.537205	3.693339	3.651677	3.580786	3.595079	3.561195	3.737936	3.669732	3.640259
100	3.637439	3.758628	3.633886	3.665112	3.647937	3.665752	3.627634	3.601295	3.845907	3.591328	3.667492

5. Hasil Pengujian Terhadap Jumlah Generasi

Jumlah Generasi	Nilai *Fitness* Percobaan ke-*i*										Rata-rata *Fitness*
	1	2	3	4	5	6	7	8	9	10	
100	3.712688	3.597156	3.628096	3.546029	3.526585	3.682380	3.633466	3.560534	3.518025	3.511946	3.5916905
200	3.611234	3.639519	3.541108	3.656571	3.640718	3.684998	3.562421	3.533572	3.520558	3.571382	3.5962081
300	3.564057	3.583661	3.528036	3.577806	3.568450	3.626415	3.681205	3.522919	3.523620	3.564155	3.5740324
400	3.585188	3.585928	3.562652	3.521677	3.540019	3.537410	3.583727	3.606048	3.601463	3.687088	3.58112
500	3.592920	3.530496	3.572221	3.536557	3.597365	3.573267	3.627490	3.638233	3.732756	3.649981	3.6051285
600	3.554353	3.516712	3.583945	3.517578	3.609703	3.583509	3.481866	3.566623	3.624497	3.586918	3.5625704
700	3.606086	3.597396	3.536828	3.610935	3.512074	3.708952	3.573304	3.575522	3.583752	3.560130	3.5864978
800	3.532956	3.549946	3.691925	3.568445	3.678900	3.578031	3.666175	3.505927	3.665458	3.669490	3.6107254
900	3.692769	3.517970	3.577184	3.573946	3.590929	3.682734	3.679239	3.543017	3.691843	3.565563	3.6115194
1000	3.588251	3.581378	3.647025	3.703833	3.505703	3.599982	3.551989	3.6850727	3.552986	3.7039688	3.6120187

6. Hasil Pengujian Terhadap Kombinasi *Crossover Rate* dan *Mutation Rate*

cr	*mr*	Nilai *Fitness* Percobaan ke-*i*										Rata-rata *Fitness*
		1	2	3	4	5	6	7	8	9	10	
1	0	3.565073	3.550624	3.551335	3.577832	3.519505	3.618957	3.618390	3.577397	3.577425	3.577264	3.573380
0.9	0.1	3.577312	3.616123	3.602222	3.649173	3.531920	3.631854	3.704873	3.664950	3.541776	3.676418	3.619662
0.8	0.2	3.580071	3.579974	3.580117	3.580117	3.765953	3.770762	3.759752	3.722042	3.522042	3.855159	3.671599
0.7	0.3	3.566242	3.626024	3.726024	3.597903	3.580248	3.586354	3.568886	3.659871	3.676330	3.578326	3.616621
0.6	0.4	3.628708	3.554321	3.548274	3.793802	3.873680	3.766689	3.864419	3.489683	3.499789	3.699279	3.671864
0.5	0.5	3.641922	3.512237	3.496411	3.476132	3.651162	3.567402	3.490550	3.807566	3.804211	3.500466	3.594806
0.4	0.6	3.699930	3.852418	3.465502	3.673209	3.673209	3.650927	3.556693	3.673209	3.673482	3.673566	3.659214
0.3	0.7	3.823947	3.852884	3.881444	3.875994	3.521780	3.524992	3.567703	3.511871	3.614722	3.507928	3.668326
0.2	0.8	3.644758	3.781792	3.599425	3.890404	3.695215	3.591678	3.724161	3.632729	3.514140	3.653454	3.672775
0.1	0.9	3.792592	3.572843	3.690802	3.526916	3.866055	3.559240	3.613985	3.734948	3.552317	3.884938	3.679464

7. Hasil Pengujian Terhadap Kombinasi Variasi *Output* yang Dioptimalkan

Kandungan Gizi Susu		Nilai *Fitness* Percobaan ke-*i*										Rata-rata *Fitness*
Optimalkan	Minimalkan	1	2	3	4	5	6	7	8	9	10	
Protein	Lemak	3.82155	3.72407	3.653797	3.657012	3.69573	3.65836	3.663482	3.665487	3.657538	3.671417	3.68684

Kasus Optimasi (Protein Optimal dan Lemak Minimal)	Nilai Presentase Percobaan ke-*i*										Rata-rata Presentase
	1	2	3	4	5	6	7	8	9	10	
Protein Dioptimalkan	0.1572	0.0414	0.0062	0.0112	0.0136	0.0160	0.0009	0.0075	0.0108	0.0274	0.0292
Lemak Diminimalkan	0.9495	0.0468	2.0150	2.1463	0.0704	2.4118	1.0496	1.3221	2.0777	2.2824	1.4372