ESTIMASI HASIL PRODUKSI BENIH TANAMAN KENAF (*HIBISCUS CANNABINUS L.)* MENGGUNAKAN METODE *EXTREME LEARNING MACHINE* (ELM) PADA BALAI PENELITIAN TANAMAN PEMANIS DAN SERAT (BALITTAS)

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:
Audia Refanda Permatasari
NIM: 145150201111041

PROGRAM STUDI TEKNIK INFORMATIKA
JURUSAN TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2018

PENGESAHAN

ESTIMASI HASIL PRODUKSI BENIH TANAMAN KENAF *(HIBISCUS CANNABINUS L.)* MENGGUNAKAN METODE *EXTREME LEARNING MACHINE* (ELM) PADA BALAI PENELITIAN TANAMAN PEMANIS DAN SERAT (BALITTAS)

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh :
Audia Refanda Permatasari
NIM: 145150201111041

Skripsi ini telah diuji dan dinyatakan lulus pada
10 Juli 2018
Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I	Dosen Pembimbing II
Dian Eka Ratnawati,S.Si, M.Kom	Bayu Rahayudi,S.T, M.T
NIP:19730619 200212 2 001	NIP: 19740712 200604 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, 16 Juli 2018

Audia Refanda Permatasari

NIM: 145150201111041

KATA PENGANTAR

Puji syukur kehadirat Tuhan YME yang telah memberikan rahmat dan karunia-Nya sehingga laporan skripsi yang berjudul "Estimasi Hasil Produksi Benih Tanaman Kenaf (*Hibiscus Cannabinus L.*) Pada Balai Tanaman Pemanis Dan Serat (Balittas) Menggunakan Metode *Extreme Learning Machine* (ELM)" dapat terselesaikan.

Penulis menyadari bahwa skripsi ini tidakakan berjalan lancar tanpa bantuan dari banyak pihak. Oleh karena itu, penulis ingin menyampaikan rasa hormat dan terimakasih kepada:

1. Ibu Dian Eka Ratnawati, S.Si,M.Kom dan Bapak Bayu Rahayudi, S.T, M.T selaku dosen pembimbing skripsi yang telah memberikan arahan terhadap penulis dengan penuh kesabaran sehingga skripsi ini dapat terselesaikan.
2. Bapak Agus Wahyu Widodo, S.T, M.Sc selaku Ketua Program Studi Teknik Informatika.
3. Bapak Tri Astoto Kurniawan, S.T, M.T, Ph.D selaku Ketua Jurusan Teknik Informatika.
4. Kedua orang tua tercinta Bapak Agung Iswanto dan Ibu Reffi Liandari, adik Shakila Ananta Zahra serta keluarga besar yang telah mendoakan, memberikan kasih sayang serta memberikan dukungan berupa moril maupun materil, sampai terselesaikannya skripsi ini.
5. Muhammad Luthfi Aries Sandy yang selalu memberikan dukungan dan semangat, doa kepada penulis.
6. Heny Dwi Jayanti, Dhea Azahria, Anim Rofi'Ah, Alysha Ghea, Novi Nur, Miracle F, dan GLDev selaku sahabat penulis selama berada di Malang yang telah memberikan semangat serta doa hingga terselesaikannya skripsi ini.
7. Octaviani Wulandari, Siti Toyyibah, Yessica Geovani, Ayunda Novitasari, Tiyara Cantika, Novia Andriani, Monica Nurcahya, Annisa Wulandari, Sari Endah Pratiwi, Calvin Afiano dan Putri Melati selaku sahabat penulis yang berada di Bandung. Yang selalu memberikan semangat dan dukungan kepada penulis, sehingga penulis semangat untuk menyelesaikan skripsi ini.
8. Irma Rahmadanti, Robih Dini, Bahruddin El Hayat yang telah menjadi teman berdiskusi selama pengerjaan skripsi, dan memberikan semangat agar terselesaikannya skripsi ini.
9. Teman-teman BIOS EXALT yang telah memberikan semangat dan dukungan, memberikan kesan dan banyak pelajaran kepada penulis.
10. Teman-teman Informatika angkatan 2014, seluruh dosen, dan civitas akademik Fakultas Ilmu Komputer yang telah banyak memberi bantuan

dan dukungan selama penulis menempuh studi dan selama penyelesaian skripsi ini.

Penulis menyadari bahwa dalam penyusunan skripsi ini masih sangat banyak kekurangan, sehingga memerlukan saran dan kritik yang membangun bagi penulis. Akhir kata penulis berharap skripsi ini dapat memberikan manfaat bagi semua pihak yang menggunakannya.

Malang, 16 Juli 2018

Penulis

audiarefandaa@gmail.com

ABSTRAK

Balai Penelitian Tanaman Pemanis dan Serat (Balittas) mengembangkan berbagai jenis tanaman serat, salah satunya adalah tanaman kenaf. Pihak Balai Balittas lebih mengedepankan produksi benih tanaman kenaf. Dalam memproduksi benih tanaman kenaf, pihak Balittas mengalami kendala yang dapat menghambat proses produksi benih tanaman kenaf. Kendala tersebut terjadi pada saat melakukan estimasi terhadap hasil produksi benih yang akan dihasilkan. Untuk itu, pada penelitian ini penulis membuat suatu sistem estimasi produksi benih tanaman kenaf menggunakan metode *Extreme Learning Machine.* Metode ini adalah salah satu metode jaringan syaraf tiruan yang memiliki keunggulan dalam segi *learning speed.* Terdapat tahapan-tahapan pada metode ELM antara lain adalah normalisasi, *training, testing,* dan denormalisasi. Hasil evaluasi sistem pada penelitian ini menggunakan *Mean Absolute Percentage Error* (MAPE). Berdasarkan pengujian yang dilakukan didapatkan nilai rata-rata MAPE terbaik sebesar 0,160% berdasarkan hasil pengujian parameter terbaik yaitu jumlah neuron sebesar 8, fungsi aktivasi *sigmoid biner* dan perbandingan data 90%:10%. Pada pengujian *k-fold cross validation* nilai MAPE terbaik berada pada nilai K sebesar 10 yaitu 0,431%.

Kata Kunci : Tanaman Kenaf, Benih Tanaman Kenaf, *Extreme Learning Machine,* MAPE

ABSTRACT

Balai PenelitianTanaman Pemanis dan Serat (Balittas) develops various types of fiber plants, one of them is kenaf. Balittas main focus is putting forward kenaf seed production. In producing kenaf seeds, the Balittas has constraints that can slow down the production process of kenaf seeds. The constraint occurs when estimating the seed production. In this research the author make an estimation system of kenaf seed production using Extreme Learning Machine method. This method is one of the artificial neural network method that has an advantage in learning speed. The steps in ELM method are normalization, training, testing and denormalization. This research uses Mean Absolute Percentage Error (MAPE) for evaluation. *From the evaluation, the best MAPE is 0,160% based on best parameter. That is 8 of number neurons on hidden layer, binary activation function, and using 90%:10% ratio for training data and testing data. On k-fold cross validation the best MAPE is 0,4314% on K=10.*

Keywords **:** *Kenaf, Kenaf Seed, Extreme Learning Machine, MAPE*

DAFTAR ISI

PENGESAHAN ii
PERNYATAAN ORISINALITAS iii
KATA PENGANTAR iv
DAFTAR ISI viii
DAFTAR TABEL xi
DAFTAR GAMBAR xiii
DAFTAR KODE PROGRAM xv
DAFTAR LAMPIRAN xvi
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 3
1.6 Sistematika pembahasan 4
BAB 2 LANDASAN KEPUSTAKAAN 6
2.1 Kajian Pustaka 6
2.2 Tanaman Kenaf 9
2.2.1 Morfologi Tanaman Kenaf 9
2.3 Benih Tanaman Kenaf 10
2.4 Jaringan Syaraf Tiruan (JST) 11
2.1.1 Arsitektur Jaringan Syaraf Tiruan 12
2.6 Fungsi Aktivasi 13
2.7 Normalisasi Data 13
2.8 *Extreme Learning Machine* (ELM) 14
2.8.1 Proses Training 14
2.8.2 Proses *Testing* 15
2.9 Proses Denormalisasi Data 16

2.10 *Mean Absolute Percentage Error (MAPE)* 16
2.11 *K-Fold Cross Validation* 17
BAB 3 METODOLOGI 18
3.1 Studi Literatur 18
3.2 Pengumpulan Data 19
3.3 Perangkat yang Digunakan Pada Penelitian 19
3.4 Perancangan 20
3.5 Implementasi 20
3.6 Pengujian dan Analisis 20
3.7 Kesimpulan dan Saran 20
BAB 4 Perancangan 21
4.1 Formulasi Permasalahan 21
4.2 Alur Proses Algoritma *Extreme Learning Machine* (ELM) 21
4.2.1 Normalisasi Data 23
4.2.2 Proses Perhitungan *Training* 24
4.2.3 Proses Perhitungan *Testing* 36
4.2.4 Denormalisasi Data 39
4.2.5 Evaluasi Sistem Menggunakan MAPE 39
4.3 Perhitungan Manual ELM 40
4.4 Perancangan Antarmuka 50
4.4.1 Perancangan Antarmuka Halaman *Input File* 50
4.4.2 Perancangan Halaman Inisialisasi *Input Weight* Dan Bias 51
4.4.3 Perancangan Antarmuka Halaman Normalisasi 51
4.4.4 Perancangan Antarmuka Halaman *Training* 52
4.4.5 Perancangan Antarmuka Halaman *Testing* 53
4.4.6 Perancangan Antarmuka Halaman Hasil 53
4.4.7 Perancangan Antarmuka Halaman *K-Fold Cross Validation* 54
4.5 Perancangan Uji Coba Dan Evaluasi Sistem 55
4.5.1 Pengujiian Jumlah *Neuron* Pada *Hidden Layer* 55
4.5.2 Pengujian Fungsi Aktivasi 55

4.5.1 Pengujian Persentase Perbandingan Data *Training* Terhadap Data *Testing* 56
4.5.2 Pengujian *K-Fold Cross Validation* 56
BAB 5 IMPLEMENTASI 58
5.1 Implementasi Sistem 58
5.1.1 Implementasi Proses Normalisasi Data 58
5.1.2 Implementasi Inisialisasi *Input Weight* serta Bias 58
5.1.3 Implementasi *Output Hidden Layer* 59
5.1.4 Implementasi *Output Hidden Layer* Menggunakan Fungsi Aktivasi61
5.1.5 Implementasi Matriks *Moore-Penrose Pseudo Inverse* 63
5.1.6 Implementasi *Inverse* Matriks OBE 64
5.1.7 Implementasi *Output Weight* 65
5.1.8 Implementasi Proses Perhitungan Nilai Estimasi 66
5.1.9 Implementasi Denormalisasi Data 67
5.1.10 Implementasi Evaluasi Sistem Menggunakan MAPE 67
5.2 Implementasi Antarmuka 68
5.2.1 Implementasi Antarmuka Halaman *Input File* 68
5.2.2 Implementasi Antarmuka Halaman *Weight&*Bias 68
5.2.3 Implementasi Antarmuka Halaman Normalisasi 69
5.2.4 Implementasi Antarmuka Halaman *Training* 69
5.2.5 Implementasi Antarmuka Halaman *Testing* 70
5.2.6 Implementasi Antarmuka Halaman Hasil 70
5.2.7 Implementasi Antarmuka Halaman *K-Fold Cross Validation* 71
BAB 6 PENGUJIAN DAN ANALISIS 72
6.1 Pengujian dan Analisis Jumlah *Neuron* Pada *Hidden Layer* 72
6.2 Pengujian dan Analisis Fungsi Aktivasi 73
6.3 Pengujian dan Analisis Persentase Perbandingan Jumlah Data *Training* Terhadap Data *Testing* 74
6.4 Pengujian *K-Fold Cross Validation* 76
BAB 7 PENUTUP 79

7.1 Kesimpulan 79
7.2 Saran 80
DAFTAR PUSTAKA 81
LAMPIRAN A Data karakterisasi tanaman kenaf 83
LAMPIRAN B Surat keterangan pengambilan data skripsi 87

DAFTAR TABEL

Tabel 2.1 Kajian Pustaka 7
Tabel 4.1 Data Karakterisasi Tanaman Kenaf 41
Tabel 4.2 Nilai *Max-Min Normalization* 42
Tabel 4.3 Hasil Normalisasi Masing-Masing Fitur 42
Tabel 4.4 Matriks *Input Weight* 43
Tabel 4.5 Hasil Transpose Matriks *Input Weight* 43
Tabel 4.6 Matriks Nilai Bias 43
Tabel 4.7 Matriks *Output Hidden Layer* 44
Tabel 4.8 Matriks *Output Hidden Layer* Menggunakan Fungsi Aktivasi 45
Tabel 4.9 Transpose Matriks *Output Hidden Layer* Dengan Fungsi Aktivasi 45
Tabel 4.10 Perkalian Matriks *Transpose* Dengan Matriks *Output Hidden Layer* Menggunakan Fungsi Aktivasi *Sigmoid Biner* 46
Tabel 4.11 Matriks I*nverse* 46
Tabel 4.12 Matriks Nilai *Moore-Penrose Pseudo Invers* (H^+) 47
Tabel 4.13 Nilai *Output Weight* 47
Tabel 4.14 Normalisasi Pada Data *Testing* 47
Tabel 4.15 Matriks *Output Hidden Layer* 48
Tabel 4.16 Perkalian Matriks *Output Hidden Layer* dengan Menggunakan Fungsi Aktivasi *Sigmoid* 48
Tabel 4.17 Hasil Perhitungan Nilai *Output Layer* 49
Tabel 4.18 Denormalisasi Data 49
Tabel 4.19 Pengujian *Input Neuron* Pada *Hidden Layer* 55
Tabel 4.20 Pengujian Nilai Fungsi Aktivasi 56
Tabel 4.21 Pengujian Peresentase Perbandingan Jumlah Data *Training* dan *Testing* 56
Tabel 4.22 Pengujian *K-Fold Cross Validation* 57
Tabel 6.1 Pengujian Jumlah *Neuron* Pada *Hidden Layer* 72
Tabel 6.2 Pengujian Fungsi Aktivasi 74

Tabel 6.3 Pengujian Persentase Perbandingan Jumlah Data *Training* Terhadap Data *Testing* 75

Tabel 6.4 Pengujian *K-Fold Cross Validation* dengan K=5 76

Tabel 6.5 Pengujian *K-Fold Cross Validation* dengan K=10 77

DAFTAR GAMBAR

Gambar 2.1 Struktur *Neuron* Jaringan Syaraf Tiruan 11

Gambar 2.2 Arsitektur *Multilayer Network* 12

Gambar 2.3 Ilustrasi *k-fold cross validation fold*=10 17

Gambar 3.1 Diagram Alir Metodologi Penelitian 18

Gambar 4.1 Diagram Alir Sistem Estimasi Produksi Benih Tanaman Kenaf Menggunakan ELM 22

Gambar 4.2 Diagram Alir Normalisasi Data 24

Gambar 4.3 Diagram Alir Training 25

Gambar 4.4 Inisialisasi *Input Weight* dan Bias 26

Gambar 4.5 Diagram Alir *Output Hidden Layer* dengan Fungsi Aktivasi Sigmoid Biner 27

Gambar 4.6 Diagram Alir *Transpose* Matriks *Input Weight* 28

Gambar 4.7 Diagram Alir Perkalian Matriks Data *Training* dengan *Transpose Weight* 30

Gambar 4.8 Diagram Alir Fungsi Aktivasi *Sigmoid Biner* 31

Gambar 4.9 Diagram Alir Fungsi Aktivasi *Sigmoid Biner* 31

Gambar 4.10 Diagram Alir Matriks *Moore-Penrose Pseudo Invers* 32

Gambar 4.11 Diagram Alir *Output Weight* 36

Gambar 4.12 Diagram Alir *Testing* 37

Gambar 4.13 Diagram Alir *Testing* 37

Gambar 4.14 Diagram Alir Denormalisasi Data 39

Gambar 5.1 Implementasi Antarmuka Halaman *Input File* 68

Gambar 5.2 Implementasi Antarmuka Halaman *Weight&*Bias 69

Gambar 5.3 Implementasi Antarmuka Halaman Normalisasi 69

Gambar 5.4 Implementasi Antarmuka Halaman *Training* 70

Gambar 5.5 Implementasi Antarmuka Halaman *Testing* 70

Gambar 5.6 Implementasi Antarmuka Halaman Hasil 71

Gambar 5.7 Implementasi Antarmuka Halaman *K-Fold Cross Validation* 71

Gambar 6.1 Grafik Pengujian Jumlah *Neuron* Pada *Hidden Layer* 73

Gambar 6.2 Grafik Pengujian Fungsi Aktivasi 74
Gambar 6.3 Grafik Pengujian Persentase Perbandingan Data *Training* Terhadap Data *Testing* 75
Gambar 6.4 Grafik Pengujian *K-Fold Cross Validation K=5* 76
Gambar 6.5 Grafik Pengujian K-Fold Cross Validation K=10 77

DAFTAR KODE PROGRAM

Kode Program 5.1 Proses Normalisasi Data 58
Kode Program 5.2 Inisialisasi *Input Weight* serta Bias 59
Kode Program 5.3 *Output Hidden Layer* 60
Kode Program 5.4 *Output Hidden Layer* Dengan Fungsi Aktivasi 62
Kode Program 5.5 Matriks *Moore Penrose Pseudo Inverse* 63
Kode Program 5.6 *Inverse* Matriks OBE 65
Kode Program 5.7 *Output Weight* 66
Kode Program 5.8 Perhitungan Nilai Estimasi 66
Kode Program 5.9 Denormalisasi Data 67
Kode Program 5.10 Evaluasi Sistem Menggunakan MAPE 68

DAFTAR LAMPIRAN

LAMPIRAN A Data karakterisasi tanaman kenaf 83
LAMPIRAN B Surat keterangan pengambilan data skripsi 87

BAB 1 PENDAHULUAN

1.1 Latar belakang

Salah satu instansi pemerintahan yang bergerak di bidang pertanian adalah Balai Penelitian Tanaman Pemanis dan Serat (Balittas). Balai Penelitian Tanaman Pemanis dan Serat (Balittas) adalah Balai Penelitian Nasional yang merupakan UPT Badan Litbang Pertanian, Kementrian Pertanian berdasarkan SK Menteri Pertanian No 613 Kpts/OT.210/1984 (Balittas, 2015). Balittas mengembangkan berbagai jenis tanaman serat, salah satunya adalah tanaman kenaf. Tanaman kenaf (*Hibicus cannabinus L.*) adalah salah satu tanaman serat yang memiliki manfaat serta dapat dikembangkan secara optimal (Hossain, et al., 2011). Serat tanaman kenaf memiliki banyak manfaat. Dari beberapa manfaat tersebut, dapat dijadikan sebagai bahan baku pembuatan kertas, bahan dasar otomotif, bahan baku pembuatan tambang, bahan baku pembuatan tas, dan sebagainya yang berbahan baku serat. (Sudjindro, 2007).

Pada proses pengembangan, komoditas tanaman kenaf diperbanyak melalui penyebaran benih, maka sangat diharuskan menggunakan benih kenaf yang bermutu tinggi. Benih yang bermutu tinggi merupakan syarat utama bagi produktivitas pertanian yang baik pula. Akan tetapi, hal tersebut tidak terlepas dari beberapa faktor yang menunjang kualitas baik suatu benih. Diantaranya adalah penanganan pada saat pemanenan harus baik, benih yang disimpan harus sudah kering atau sudah memiliki sedikit kadar air (Hasanah, 2002). Permintaan produsen terhadap produksi benih tanaman kenaf sangat tinggi. Oleh karena itu pihak Balittas lebih mengedepankan produksi benih tanaman kenaf.

Dalam memproduksi benih tanaman kenaf, pihak Balittas mengalami kendala. Kendala tersebut terjadi pada saat melakukan estimasi terhadap hasil produksi benih yang akan dihasilkan. Cara untuk mengestimasi hasil produksi benih masih terbilang sangat manual, yakni dengan menggunakan metode *sampling*. (Marjani, 2017). Akan tetapi cara tersebut tidak efektif, karena pada proses penanganan benih seringkali tidak tepat. Penanganan benih dimulai dari proses panen hingga penjemuran. Apabila estimasi yang dilakukan tidak sesuai, maka seringkali terjadi produksi benih yang berlebih ataupun kekurangan. Produksi benih yang berlebih menyebabkan adanya pemborosan terhadap pengeluaran. Kemudian apabila produksi benih terjadi kekurangan, maka akan terjadi penurunan terhadap kualitas benih tanaman kenaf. Hal tersebut dapat menghasilkan benih dengan kualitas yang buruk.

Dengan adanya permasalahan tersebut, maka penulis berkeinginan untuk merancang suatu sistem yang dapat mengestimasi hasil produksi benih tanaman kenaf. Hal tersebut dilakukan supaya membantu pihak Balittas dalam melakukan estimasi terhadap hasil produksi benih tanaman kenaf, sehingga dapat melakukan penanganan benih dengan tepat dan menghasilkan produksi benih tanaman kenaf yang berkualitas tinggi dengan memiliki produksi biji berjumlah banyak. Dalam

melakukan penelitian estimasi hasil produksi benih tanaman kenaf, terdapat empat buah peubah prediktor. Untuk melakukan pemilihan empat prediktor, didasarkan pada teori mengenai bagian pada tanaman kenaf yang berpengaruh terhadap produksi total tanaman kenaf. Diantara prediktor tersebut antara lain adalah umur tanaman pada saat berbunga, diameter batang bagian bawah, jumlah kapsul masak, dan berat benih 10 tanaman (Cahyanti, 2017).

Ada berbagai macam metode yang dapat diimplementasikan untuk melakukan suatu estimasi dengan hasil yang akurat. Dari berbagai macam metode tersebut salah satunya adalah metode *Extreme Learning Machine* (ELM). Metode ELM merupakan suatu metode *feedforward* dan dapat dikatakan sebagai metode baru dalam Jaringan Syaraf Tiruan (JST), dengan memiliki satu *hidden layer* atau sering kali disebut dengan *single hidden layer feed forward neural network* (SLFNs). Metode ELM memiliki kelebihan dalam segi *learning speed* dibandingan dengan metode JST lainnya. Dalam segi performa serta konsumsi waktu, metode ELM lebih unggul dibandingakan metode *Support Vector Machine* (SVM) dan *Backpropagation* (BP) (Huang, et al., 2006).

Penelitian yang dilakukan oleh (Pati, et al., 2013) melakukan sebuah perbandingan metode *Backpropagation* (BPNN) dengan metode *Extreme Learning Machine* (ELM). Studi kasus yang diterapkan adalah melakukan prediksi terhadap permintaan barang di perusahaan Orissa Power Transmission di California. Hasil yang didapatkan pada penelitian tersebut menunjukan bahwa kinerja ELM dalam segi waktu pelatihan lebih unggul dibandingkan dengan metode BPNN. Akurasi yang didapatkan pada saat menggunakan metode BPNN berkisar antara 92,743% sampai dengan 97,371% sedangkan pada saat menggunakan metode ELM akurasi yang didapatkan berkisar antara 94,098% sampai dengan 98,44%. Hal tersebut menunjukkan bahwa akurasi yang didapatkan dengan menggunakan metode ELM jauh lebih baik.

Penelitian yang dilakukan oleh (Naji, et al., 2015) yaitu melakukan estimasi terkait memperkirakan konsumsi bangunan berdasarkan ketebalan bahan bangunan. Dari hasil penelitian, didapatkan bahwa variasi dalam konsumsi energi bangunan sebagian besar dipengaruhi oleh bahan insulasi. Nilai error yang didapatkan dengan menggunakan RMSE adalah sebesar 0,03.

Penelitian yang dilakukan oleh (Shamshirband, et al., 2015) adalah melakukan estimasi terhadap parameter *k* dan *c* yang merupakan faktor dari distribusi angin probabilistik. Nilai evaluasi pada penelitian ini menggunakan *Mean Absolute Percentage Error* (MAPE). MAPE yang didapatkan adalah sebesar 8,46%. Hal tersebut menunjukkan hasil yang baik dalam melakukan estimasi, karena nilai MAPE yang dihasilkan kurang dari 10%.

Penelitian yang dilakukan oleh (Hidayatullah & Umar, 2014) adalah melakukan estimasi terhadap radiasi matahari yang dilakukan setiap jamnya. Data yang

digunakan pada penelitian merupakan data radiasi matahari yang dihitung setiap jamnya pada permukaan horizontal (W/m^2), temperature udara (^oC), kelembapan relatif (%) serta kecepatan angin (m/s). Dari hasil penelitian ini didapatkan nilai error terkecil menggunakan MSE sebesar 5,88E-14.

Dari beberapa penelitian yang dilakukan sebelumnya membuktikan bahwa dengan menggunakan metode *Extreme Learning Machine* akan mampu menghasilkan sistem untuk mengestimasi suatu permasalahan. Dengan performa *learning speed* yang baik serta nilai akurasi yang dihasilkan pun terbilang baik, maka dari itu akan diusulkan penelitian yang dapat melakukan estimasi terhadap hasil produksi benih tanaman kenaf menggunakan metode *Extreme Learning Machine* (ELM). Diharapkan sistem ini mampu menangani permasalahan yang dihadapi oleh pihak Balittas, sehingga dapat melakukan estimasi terhadap hasil produksi benih tanaman kenaf dan melakukan penanganan benih dengan baik sehingga menghasilkan kualitas unggul terhadap hasil produksi benih.

1.2 Rumusan masalah

Berdasarkan penjelasan dari latar belakang, maka akan didapatkan rumusan masalah sebagai berikut:

1. Bagaimana implementasi metode *Extreme Learning Machine* (ELM) dalam melakukan estimasi hasil produksi benih tanaman kenaf?
2. Berapa nilai MAPE yang dihasilkan pada hasil pengujian sistem estimasi hasil produksi benih tanaman kenaf menggunakan metode *Extreme Learning Machine* (ELM)?

1.3 Tujuan

Adapun tujuan dari penelitian ini adalah sebagai berikut:

1. Mengimplementasikan metode *Extreme Learning Machine* (ELM) untuk melakukan estimasi produksi benih tanaman kenaf.
2. Mengetahui nilai MAPE yang dihasilkan berdasarkan pengujian sistem estimasi hasil produksi benih tanaman kenaf menggunakan metode *Extreme Learning Machine* (ELM).

1.4 Manfaat

Adapun manfaat yang didapat dari penelitian ini antara lain adalah:

1. Membantu Balai Penelitian Tanaman Pemanis dan Serat (Balittas) dalam mengestimasi produksi benih tanaman kenaf.
2. Membantu Balai Penelitian Tanaman Pemanis dan Serat (Balittas) dalam melakukan penanganan produksi benih tanaman kenaf sehingga menghasilkan mutu benih dengan kualitas yang baik.

1.5 Batasan masalah

Dalam penelitian ini dibatasi oleh beberapa hal antara lain adalah :

1. Data yang di gunakan adalah hasil penelitian yang dilakukan oleh pihak Balittas yaitu penelitian mengenai karakterisasi tanaman kenaf pada tahun 2013 berjumlah 100 data.
2. Dalam melakukan implementasi sistem, menggunakan metode *Extreme Learning Machine* (ELM).
3. Evaluasi pada sistem menggunakan *Mean Absolute Percentage Error* (MAPE).
4. Parameter yang digunakan adalah umur tanaman saat berbunga, diameter batang bagian bawah, jumlah kapsul masak, dan berat benih 10 tanaman.

1.6 Sistematika pembahasan

Berikut merupakan sistematika pembahasan pada penelitian yang mencakup setiap babnya:

BAB I PENDAHULUAN

Pada Bab I membahas beberapa sub bab diantaranya adalah membahas mengenai latar belakang, rumusan masalah, tujuan, manfaat, batasan masalah, serta sistematika pembahasan

BAB II TINJAUAN PUSTAKA

Pada Bab II membahas mengenai dasar teori yang dapat mendukung dari objek penelitian yang dilakukan saat ini berdasarkan dengan teori dan sumber pustaka serta metode yang digunakan dalam penelitian. Antara lain adalah penjelasan mengenai tanaman kenaf, benih tanaman kenaf, jaringan syaraf tiruan, serta metode yang di gunakan yaitu metode *Extreme Learning Machine* (ELM).

BAB III METODOLOGI PENELITIAN

Pada Bab III membahas terkait dengan langkah-langkah yang dilakukan pada penulisan skripsi yakni terdiri dari studi literatur, pengumpulan data, analisis kebutuhan, perancangan, implementasi, pengujian dan analisis, dan tahap terakhir yaitu kesimpulan.

BAB IV PERANCANGAN

Pada Bab IV membahas tentang proses perancangan sistem yang terdiri dari perancangan manualisasi, perancangan antarmuka, perancangan uji coba serta perancangan evaluasi sistem.

BAB V IMPLEMENTASI

Pada Bab V membahas mengenai implementasi sistem yang dibuat berdasarkan perancangan yang dilakukan sebelumnya.

BAB VI PENGUJIAN DAN ANALISIS

Pada Bab VI membahas tentang pengujian yang dilakukan serta memberikan analisis terhadap hasil yang telah didapat.

BAB VII PENUTUP

Pada Bab VII membahas mengenai kesimpulan yang didapatkan berdasarkan hasil serta pengujian yang dilakukan pada penelitian ini. Kemudian, saran yang bertujuan untuk membangun penelitian selanjutnya, agar dapat menghasilkan hasil yang lebih baik dari penelitian sebelumnya.

BAB 2 LANDASAN KEPUSTAKAAN

2.1 Kajian Pustaka

Salah satu metode yang dapat digunakan untuk melakukan estimasi adalah metode *Extreme Learning Machine* (ELM). Dalam model matematis yang dihasilkan oleh ELM dapat dikatakan lebih sederhana diantara metode Jaringan Syaraf Tiruan lainnya (Singh & Balasundaram, 2007). Oleh karena itu dalam penelitian ini akan menggunakan metode *Extreme Learning Machine* (ELM) untuk melakukan estimasi hasil produksi benih tanaman kenaf.

Penelitian mengenai estimasi atau perkiraan telah banyak dilakukan sebelumya diantaranya adalah yang dilakukan oleh (Hidayatullah & Umar, 2014). Pada penelitian ini, dilakukan estimasi terhadap panel surya agar didapatkan keluaran listrik dengan hasil yang optimal. Dalam melakukan proses perhitungan data, dibagi menjadi data *training* dan data *testing.* Dalam pembagian data tersebut terdapat 84 data untuk setiap parameternya. Data yang digunakan untuk proses *training* adalah sebesar 80% yaitu sebanyak 292 data, serta 20% digunakan untuk data *testing* yakni sebanyak 72 data. Masukkan terhadap sistem yang digunakan adalah radiasi matahari dengan rata-rata per jam pada permukaan yang horizontal (W/m^2), kecepatan angin (m/s), kelembapan relatif (%), serta temperatur udara yang dihasilkan (^oC). Dari hasil penelitian tersebut didapatkan nilai *error* MSE 5,88E-14 serta didapatkan pula *learning speed* sebesar 0,0156 per detik.

Penelitian yang dilakukan oleh (Pati, et al., 2013) adalah melakukan sebuah prediksi terhadap permintaan barang pada perusahaan Orissa Power Transmission di California. Pada penelitian tersebut membandingkan metode *Extreme Learning Machine* (ELM) dengan metode *Backpropagation* (BPNN) dalam melakukan prediksi. Data yang digunakan pada penelitian tersebut adalah data permintaan barang pada bulan April 2011 hingga Maret 2012. Hasil yang didapatkan pada penelitian tersebut menunjukkan bahwa metode ELM mendapatkan hasil yang lebih unggul dalam performa kecepatan waktu pelatihan. Selain itu akurasi yang didapatkan pada penelitian ini menunjukkan bahwa metode ELM lebih baik dibandingkan metode BPNN yaitu mencapai 98,44%.

Penelitian yang dilakukan oleh (Shamshirband, et al., 2015) adalah estimasi terhadap nilai parameter bentuk (*k)* dan skala *(c)* yang merupakan faktor dari distribusi angin probabilistik atau disebut dengan *weibull*. Energi angin adalah salah satu sumberdaya yang paling tepat dan dapat digunakan sebagai pemasok kebutuhan energi. Data yang digunakan merupakan data kecepatan angin selama 5 tahun terakhir di kota Aligoodarz. Kecepatan angin tersebut diukur menggunakan *anemometer*. *Anemometer* diukur pada ketinggian 10 m diatas permukaan tanah dan diukur secara tiga periode per jam. Langkah pertama yang dilakukan adalah menganalisis tiga kecepatan angin per jam kemudian dirata-ratakan agar didapatkan

data harian. Tujuan utama dari penelitian ini adalah untuk menilai kemahiran algoritma ELM dalam melakukan estimasi parameter *k* dan *c* yang merupakan fungsi dari distribusi *weibull* dalam skala bulanan. Akurasi yang didapatkan dengan menggunakan MAPE adalah sebesar 8,60%. Hal tersebut menunjukkan akurasi yang baik dikarenakan nilai MAPE kurang dari 10%.

Penelitian selanjutnya dilakukan oleh (Naji, et al., 2015) yang melakukan estimasi terhadap kebutuhan energi bahan bangunan. Permintaan energi serta bahan bangunan yang semakin meningkat dapat menimbulkan masalah bagi masa depan yang akan berkelanjutan. Efisiensi energi bangunan dapat ditingkatkan dengan cara melakukan estimasi agar dapat membantu arsitek dan insinyur dalam menciptakan struktur yang berkelanjutan. Data yang digunakan sebagai *input* adalah berupa material dinding, jenis material, ketebalan material, serta sifat dari material tersebut. Dari hasil penelitian tersebut maka di dapatkan nilai error RMSE sebesar 0,03.

Penelitian yang akan dilakukan kali ini adalah melakukan estimasi terhadap hasil produksi benih tanaman kenaf. Data yang digunakan merupakan hasil dari penelitian karakterisasi tanaman kenaf yang dilakukan oleh pihak Balittas. Dari data tersebut terdapat beberapa prediktor yang mampu mempengaruhi hasil dari estimasi. Diantara prediktor tersebut adalah umur tanaman pada saat berbunga, diameter batang bagian bawah, jumlah kapsul masak, serta berat benih 10 tanaman. Sistem akan mengolah data menggunakan metode *Extreme Learning Machine* (ELM) untuk menghasilkan *output* berupa estimasi hasil produksi benih tanaman kenaf serta akan didapatkan nilai evaluasi sistem menggunakan perhitungan *Mean Absolute Percentage Error* (MAPE).

Untuk perbandingan pada objek serta metode pada penelitian yang telah dilakukan maka akan ditunjukkan oleh Tabel 2.1.

Tabel 0.1 Kajian Pustaka

No	Judul	Objek	Metode	Hasil
1	(Hidayatullah & Umar, 2014)	Panel Surya	*Extreme Learning Machine*	*Output* yang dihasilkan adalah estimasi intensitas radiasi matahari. Nilai *error* yang dihasilkan dengan menggunakan MSE adalah 5,88E-14.

Tabel 0.1 Kajian Pustaka (Lanjutan)

No	Judul	Objek	Metode	Hasil
2	(Pati, et al., 2013)	Permintaan barang pada perusahaan Orissa Power di California	*Extreme Learning Machine* (ELM) dan *Backpropagation* (BPNN)	Perbandingan metode ELM dengan metode BPNN dalam melakukan prediksi terhadap permintaan barang menunjukkan bahwa metode ELM memiliki kelebihan waktu pelatihan dibandingkan metode BPNN. Akurasi yang didapatkan dengan menggunakan metode ELM lebih unggul yaitu mencapai 98,44%
3	(Shamshirband, et al., 2015)	Kecepatan angin di kota Aligodarz	*Extreme Learning Machine*	Nilai parameter bentuk (*k*) dan skala (*c*) yang merupakan faktor kecepatan angin. Nilai MAPE yang dihasilkan adalah sebesar 8,60%
4.	(Naji, et al., 2015)	Energi bangunan	*Extreme Learning Machine*	Estimasi mengenai kebutuhan bangunan. Nilai

				error dengan menggunakan RMSE adalah 0,03

Tabel 0.1 Kajian Pustaka (Lanjutan)

No.	Judul	Objek	Metode	Hasil
5	Estimasi Hasil Produksi Benih Tanaman Kenaf (*Hibiscus Cannabinus L.*) Menggunakan Metode *Extreme Learning Machine* (ELM) Pada Balai Penelitian Tanaman Pemanis Dan Serat (Balittas)	Benih tanaman kenaf	*Extreme Learning Machine*	Estimasi hasil produksi benih tanaman kenaf. Evaluasi sistem menggunakan MAPE

2.2 Tanaman Kenaf

Tanaman kenaf (*Hibiscus Cannabinus L.)* merupakan tanaman yang masuk ke dalam keluarga *malvaceae* dan merupakan *genus Hibicus* (Djajadi, et al., 2009). Tanaman kenaf merupakan tanaman yang memiliki umur yang relatif pendek. Tanaman tersebut berkisar 140 harian. Kenaf merupakan tanaman yang di kembang biakkan melalui benih. Tanaman kenaf merupakan tanaman serat yang hampir seluruh bagian pada tanaman tersebut dapat dimanfaatkan. Baik itu batang, daun, dan biji. Batang kenaf dapat dimanfaatkan sebagai bahan baku untuk industri, seperti industri *fibre board, geo textile, soil remediation, pulp* dan kertas, karpet, dan lain sebagainya yang berbahan dasar serat. Karena serat dari tanaman kenaf ini bersifat ramah lingkungan, serta memiliki nilai ekonomis yang cukup tinggi. Kemudian bagian daun tanaman kenaf memiliki protein yang cukup baik yang dapat dimanfaatkan sebagai pakan ternak *ruminansia* dan unggas. Selain itu biji tanaman kenaf dapat dimanfaatkan sebagai bahan dasar pembuatan minyak goreng karena mengandung 20% lemak (asam lemak) tidak jenuh (*oleat* dan *linoleat*) (Balittas, 2015).

2.2.1 Morfologi Tanaman Kenaf

Tanaman kenaf memiliki morfologi yang dapat mempelajari struktur tubuh tanaman kenaf. Diantara morfologi tersebut adalah (Nurdjajati, et al., 2009):

- **Batang**

Tanaman kenaf memiliki tinggi mencapai 4 meter. Tinggi tanaman kenaf tersebut bergantung pada varietas, kesuburan tanah serta waktu tanam yang dibutuhkan. Batang tanaman kenaf memiliki diameter mencapai 25 mm bergantung pada varietas tanaman kenaf serta lingkungan tempat tanaman kenaf tersebut bertumbuh. Pada permukaan batang, kenaf memiliki tekstur yang licin, berbulu halus, berbulu kasar, maupun berduri. Untuk memberikan ciri terhadap batang tersebut, dapat dibedakan berdasarkan warna serta tingkat halus atau tidaknya batang tanaman kenaf tersebut.

- **Daun**

Daun yang terdapat pada tanaman kenaf terletak dengan jarak berselang-seling, serta letaknya terdapat dalam cabang serta batang utama. Pada bagian daun tersebut akan terlihat perbedaan warna yang dihasilkan terutama pada bagian urat daun serta tepi daun. Tekstur daun terebut pun ada yang berduri, berbulu, tidak berduri, maupun tidak berbulu.

- **Bunga**

Tanaman kenaf adalah tanaman yang memiliki serbuk sari. Akan tetapi sebesar 4% dapat terjadi penyerbukan secara menyilang. Tanaman kenaf dapat menghasilkan bunga pada minggu ke-12 setelah proses tanam. Bunga pada tanaman kenaf biasanya dapat berdiri sendiri. Pada waktu proses pemekaran, akan membutuhkan waktu yang cukup singkat. Waktu tersebut dapat terjadi sebelum matahari terbit, kemudian akan menutup pada siang hari atupun sore hari.

- **Buah**

Buah yang terdapat pada tanaman kenaf berbentuk bulat meruncing menyerupai kapsul. Ukuran buah pada tanaman kenaf adalah mencapai 2-2,5 cm serta memiliki diameter sebesar 1-1,5 cm. Pada struktur buah tanaman kenaf ini memiliki variasi yang berbeda-beda. Diantaranya terdapat buah yang pada permukaannya memiliki bulu-bulu yang berukuran pendek, bertekstur halus, maupun bulu yang berjumlah cukup banyak. Tanaman kenaf memiliki waktu yang bervariasi dalam pemasakan buah. Buah-buah yang berada di posisi bawah akan terlebih dahulu masak dibandingkan dengan buah yang ada pada bagian pucuk atau atas. Hal tersebut akan mengakibatkan tingkat kemasakan buah menjadi *heterogen*.

- **Biji**

Biji pada tanaman kenaf cenderung memiliki bentuk menyerupai ginjal. Biji tersebut memiliki diameter sekitar 0,3-0,5 cm. Warna yang dihasilkan oleh biji tanaman kenaf adalah kelabu agak kecokelatan. Bentuk dari biji tanaman kenaf ini bervariasi, ada yang berbentuk *reniform, subreniform,* serta *angular.* Berat yang dihasilkan pada suatu biji tanaman kenaf adalah berisi 30.000 hingga 40.000 butir. Dalam biji tersebut 20% kandungannya merupakan minyak.

2.3 Benih Tanaman Kenaf

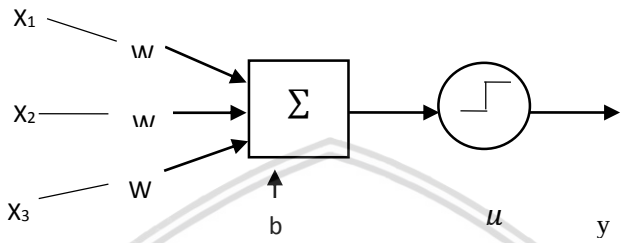
Tanaman kenaf merupakan salah satu tanaman yang diperbanyak melalui benih, hal tersebut menyebabkan benih yang digunakan haruslah memiliki mutu yang tinggi agar dapat menunjang kualitas tanaman kenaf. Benih yang memiliki mutu yang tinggi adalah suatu dasar bagi pertanian dalam peningkatan kualitas yang lebih baik. Baik itu kondisi pada sebelum, selama, ataupun sesudah panen harus tetap diperhatikan. Dalam penanganan benih tanaman kenaf, haruslah diperhatikan, karena mutu benih yang baik belum tentu baik juga apabila penanganannya kurang baik (Hasanah, 2002). Benih tersebut memiliki mutu yakni mutu genetik, fisiologik, dan fisik. Yang dimaksud dengan mutu genetik adalah benih yang memiliki genetik murni dan sangat baik apabila benih tersebut ditanam. Maka akan sesuai dengan yang diharapkan serta dideskripsikan oleh pemulianya. Selanjutnya adalah mutu fisiologik. Mutu fisiologik adalah mutu benih yang dapat menghasilkan tanaman yang normal karena ditentukan oleh daya hidup (*viabilitas*). Sedangkan mutu fisik merupakan mutu yang didasari oleh penampilan fisik layaknya kebersihan, butiran, kesegaran, ataupun utuh atau tidaknya kulit benih, serta adanya luka dan retak-retak (Suryadi, 2000). Dalam memproduksi benih dengan kualitas yang baik memang dilakukan dengan waktu yang cukup panjang. Proses tersebut dimulai dari pemilihan bahan tanaman, pemeliharaan tanaman, panen, dan melakukan penanganan setelah roses panen dilakukan (Hasanah, 2002). Dalam memproduksi benih kenaf, harus memperhatikan faktor-faktor lainnya yang dapat mempengaruhi kualitas benih diantaranya adalah faktor lokasi, cuaca, isolasi, tenaga yang terampil dan murah, dan transportasi yang cukup memadai.

2.4 Jaringan Syaraf Tiruan (JST)

Jaringan syaraf tiruan (JST) adalah sebuah sistem yang dapat memproses suatu informasi yang didalamnya memiliki suatu karakteristik khusus sehingga mirip dengan Jaringan Syaraf Biologi (JSB). Jaringan syaraf tiruan memiliki konsep berupa model matematis sehingga didapatkan pernyataan seperti berikut (Wuryandari & Afrianto, 2012) :

1. *Neuron* merupakan proses untuk mengolah informasi yang dilakukan pada suatu elemen yang sederhana.
2. Dengan menggunakan penghubung, maka akan dialirkan sinyal yang terjadi pada saraf/*neuron.*
3. Pada masing-masing penghubung, memiliki suatu bobot yang dapat bersesuaian. Bobot tersebut dapat berfungsi untuk mengalikan suatu sinyal yang dikirimkan sebelumnya melalui *neuron.*
4. Masing-masing syaraf akan memberikan jalan keluar terhadap fungsi aktivasi yang telah dilakukan penjumlahan sebelumnya.

Terdapat banyak jenis jaringan syaraf, akan tetapi padan masing-masing jenis tersebut memiliki sebuah komponen yang serupa. Layaknya struktur otak pada manusia, yang memiliki beberapa *neuron* dan saling berhubungan memiliki fungsi untuk menyalurkan informasi kepada *neuron-neuron* yang lainnya (Sinuhaji, 2009). Hubungan yang terjadi antar satu *neuron* dengan *neuron* yang lainnya disebut dengan bobot. Pada Gambar 2.1 akan menjelaskan mengenai struktur *neuron* pada jaringan syaraf tiruan.

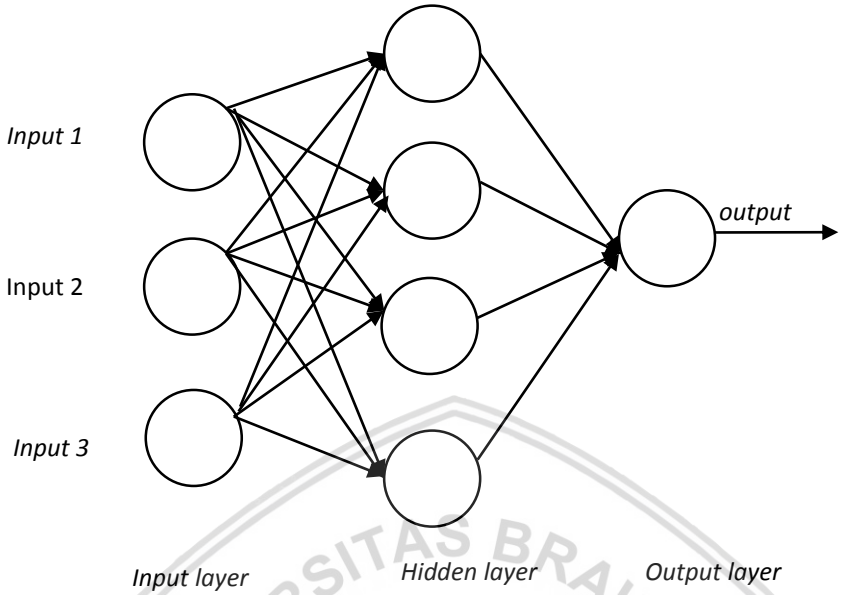


Gambar 0.1 Struktur *Neuron* Jaringan Syaraf Tiruan

Pada Gambar 2.1 menunjukkan bahwa model pada *neuron* memiliki nilai sebanyak n sebagai masukkan. Nilai n yang menjadi masukkan adalah x_1, x_2, …… x_n. Berdasakan masukkan tersebut kemudian akan diproses oleh bobot w_1, w_2, …, w_n. Hasil dari pengolahan tersebut akan diolah oleh *neuron*, sehingga menghasilkan sinyal seperti berikut $x_i^1 = x_i w_i$, $i = 1, 2, \ldots\ldots, n$. Berikutnya *neuron* bertugas untuk menjumlahkan sinyal hasil masukkan dan dibandingkan apakah nilai tersebut mengalami aktivasi atau tidak. Perbandingan yang menunjukkan tingkat aktivasi tersebut ditunjukkan melalui nilai ambang (*threshold)*. Agar dapat melanjutkan informasi yang telah didapatkan, maka nilai *input* yang dihasilkan harus melalui nilai ambang tertentu sehingga nilai *input* tersebut dapat diaktifkan dan kemudian dilanjutkan agar menghasilkan keluaran yang akan dikirimkan untuk *neuron-neuron* lainnya (Sinuhaji, 2009).

1.1.1 Arsitektur Jaringan Syaraf Tiruan

Pada jaringan syaraf tiruan terdapat terdapat arsitektur jaringan yang memiliki bayak lapisan atau biasa disebut dengan (*Multilayer Network)* (Sinuhaji, 2009). (*Multilayer Network)* atau disebut dengan jaringan dengan memiliki banyak lapisan. Pada jaringan *multilayer network* terdapat lebih dari 1 lapisan yang letaknya diantara lapisan *input* dengan lapisan *output.* Jaringan *multilayer network* memiliki lebih dari satu jaringan tersembunyi (*hidden node).* Jumlah dari *hidden node* tersebut bergantung pada masalah yang jenisnya dapat berbeda-beda. Pada jaringan ini dapat menyelesaikan masalah yang lebih kompleks dibandingkan dengan jaringan *single layer network.* Akan tetapi menggunakan pembelajaran yang lebih rumit dibandingkan dengan jaringan *single layer.* Gambar 2.2 akan menunjukkan mengenai arsitektur *multilayer network*



Gambar 0.2 Arsitektur *Multilayer Network*

1.6 Fungsi Aktivasi

Fungsi aktivasi adalah suatu fungsi yang akan melakukan transformasi terhadap suatu *input* menjadi *output* tertentu. Adapun menurut (Srimuang, et al., 2015) fungsi aktivasi yang dapat diterapkan adalah sebagai berikut:

1. Fungsi Aktivasi *Sigmoid Biner.*

$$h(x) = \frac{1}{1+exp^{-H_{init}}} \quad (2.1)$$

Keterangan:

$h(x)$ = Fungsi aktivasi

$exp^{-H_{init}}$ = Exponensial dengan pangkat minus nilai H_{init}

2. Fungsi Aktivasi *Sin*

$$h(x) = \sin(H_{init}) \quad (2.2)$$

Keterangan:

$h(x)$ = Fungsi aktivasi

$\sin(H_{init})$ = Nilai sin pada H_{init}

3. Fungsi aktivasi *Sigmoid Bipolar.*

$$h(x) = \frac{1-exp^{-Hinit}}{1+exp^{-Hinit}} \quad (2.3)$$

Keterangan:

$h(x)$ = Fungsi aktivasi

$exp^{-H_{init}}$ = Exponensial dengan pangkat minus nilai H_{init}

1.7 Normalisasi Data

Proses normalisasi data dilakukan karena range antara nilai *input* memiliki variasi yang berbeda-beda yakni dapat bernilai puluhan bahkan dapat mencapai ribuan. Nilai *input* yang akan di proses oleh nilai *output* yang bernilai kecil, harus disesuaikan agar nilai yang dihasilkan dapat menghasilkan nilai normalisasi yang stabil. Dalam penelitian kali ini, akan dilakukan proses normalisasi yang bertujuan agar antara satu data dengan data lainnya dapat saling menyesuaikan. Normalisasi yang digunakan adalah *Min-Max Normalization* (Jain & Bhandare, 2011) yang akan dijabarkan melalui Persamaan 2.4 berikut:

$$\acute{d} = \frac{(d - \min(x))}{(\max(x) - \min(x))} \quad (2.4)$$

Keterangan:

$\acute{d}$ = nilai yang didapatkan dari hasil normalisasi

d = nilai asli pada data

min = nilai minimum yang dihasilkan melalui *dataset* pada fitur x

max = nilai maksimum yang dihasilkan melalui *dataset* pada fitur x

1.8 *Extreme Learning Machine* (ELM)

Metode *Extreme Learning Machine*(ELM) merupakan metode *feedforward* dan dapat dikatakan sebagai metode baru pada jaringan syaraf tiruan dengan memiliki satu *hidden layer* atau seringkali disebut dengan *single hidden layer feed forward neural network (SLFNs)* (Huang, et al., 2006). Pada jaringan *feedforward,* menerapkan parameter-parameter yang nilainya dapat ditentukan secara manual layaknya *input weight* dan bias. Keduanya dibangkitkan secara acak pada kurun waktu yang bersamaan.

Metode *Extreme Learning Machine* dibentuk agar dapat memperkecil *learning speed* yang dihasilkan. Adapun alasan dibalik *learning speed* yang rendah, antara lain adalah (Huang, et al., 2006):

1. Dalam melakukan proses *training,* metode pembelajaran menerapkan *slow gradient based learning algorithm.*
2. Dengan menggunakan metode pembelajaran secara *iterative,* maka akan menghasilkan parameter pada masing-masing jaringan.

1.8.1 Proses Training

Sebelum dilakukan proses estimasi, maka dilakukan proses *training.* Hal tersebut berfungsi agar mendapatkan nilai *output weight.* Adapun tahapan-tahapan dalam melakukan proses *training* diantaranya adalah sebagai berikut (Huang, et al., 2006):

1. Tahap pertama yaitu melakukan inisialisasi terhadap *input weight* dan bias. Nilai tersebut kemudian diinisialisasi secara acak menggunakan nilai dengan *range* antara -1 sampai dengan 1 (Liang, et al., 2006).
2. Perhitungan *output hidden layer* (H_{init}) akan dijabarkan pada Persamaan 2.5. Setelah didapatkan hasil dari perhitungan *output hidden layer,* kemudian dilakukan perhitungan *output hidden layer* dengan fungsi aktivasi.

$$H_{init\ ij} = (\sum_{k=1}^{n} w_{jk}.x_{jk}) + b_j \quad (2.5)$$

Keterangan:

$H_{init\ ij}$ = Matriks *output hidden layer.*

i = [1,2,….,N) nilai N adalah keseluruhan dari jumlah data.

j = [1,2,….$\widetilde{N}$] nilai $\widetilde{N}$ adalah keseluruhan dari jumlah *hidden neuron.*

n = Jumlah nilai *input* pada *neuron.*

w = Nilai *input weight.*

x = Nilai *input* data yang akan digunakan.

b = Nilai bias.

3. Melakukan perhitungan nilai H^+ atau matriks *moore-penroses pseudo inverse* dengan menggunakan Persamaan 2.6.

$$H^+ = (H(x)^T * H(x))^{-1} * H^T \quad (2.6)$$

Keterangan:

H^+ = Matriks *Moore Penrose Pseudo Inverse*.

$H(x)$ = Matriks *output hidden layer* dengan fungsi aktivasi.

$(H(x)^T * H(x))^{-1}$ = Matriks *inverse* perkalian H *transpose* dengan *output layer* menggunakan fungsi aktivasi.

H^T = Matriks transpose *output hidden layer.*

4. Melakukan perhitungan *output weight* yang berasal dari *hidden layer* menuju *output layer.* Perhitungan ini dilakukan dengan cara mengalikan matriks *moore*

penrose pseudo inverse dengan nilai matriks target yang telah dinormalisasi. Pada Persamaan 2.7 akan dijabarkan cara untuk menghitung nilai *output weight.*

$$\beta = H^{+}T \quad (2.7)$$

Keterangan:

β = Nilai matriks *output weight.*

H^{+} = Nilai matriks *Moore-Penrose Pseudo Inverse.*

T = Nilai matriks target.

1.8.2 Proses *Testing*

Dalam proses *testing* memiliki tujuan agar didapatkan hasil estimasi melalui hasil *training* yang dilakukan sebelumnya. Pada proses *testing* terdapat *input weight,* bias serta *output weight* yang dihasilkan melalui proses *training.* Adapun langkah-langkah yang dilakukan dalam melakukan proses *testing* adalah sebagai berikut (Huang, et al., 2006):

1. Menghitung inisialisasi nilai *bobot masukan* serta bias secara *random* yang dihasilkan melalui proses *training.*
2. Nilai *output hidden layer* dihitung menggunakan fungsi aktivasi.
3. Untuk mendapatkan hasil berupa *output layer* adalah melalui nilai *bobot keluaran* yang didapatkan melalui proses *training.* Hal tersebut dilakukan dengan cara mengalikan nilai *output weight* dengan *output hidden layer* menggunakan fungsi aktivasi. Adapun Persamaan 2.8 akan menjabarkan mengenai perhitungan *output layer* yang berguna sebagai hasil akhir.

   $$y = H(x).\beta \quad (2.8)$$

   Keterangan:

   y = *Output layer.*

   $H(x)$ = *Output hidden layer* yang dihitung menggunakan fungsi aktivasi.

   β = *Output weight* yang didapatkan melalui proses *training.*

1.9 Proses Denormalisasi Data

Setelah dilakukan normalisasi pada data, hal yang dilakukan selanjutnya adalah proses denormalisasi data. Proses denormalisasi berfungsi untuk membangkitkan nilai yang sebelumnya telah dilakukan normalisasi, menjadi suatu nilai yang bernilai *real.* Pada Persamaan 2.9 akan dijabarkan mengenai proses denormalisasi data.

$$d = d' \, (\max - \min) + min \quad (2.9)$$

Keterangan:

d = nilai hasil perhitungan denormalisasi

d' = nilai hasil dari prediksi sebelum dilakukan denormalisasi

min = nilai minimum yang tersedia pada dataset dalam fitur x

max = nilai maksimum yang tersedia pada dataset dalam fitur x

1.10 *Mean Absolute Percentage Error (MAPE)*

Dalam melakukan perhitungan evaluasi, dilakukan menggunakan nilai *Mean Absolute Percentage Error* (MAPE). Dengan menggunakan nilai MAPE akan membandingkan selisih diantara nilai ramalan dengan nilai yang aktual. Dalam Persamaan 2.10 akan dijabarkan mengenai rumus perhitungan nilai MAPE (Kim & Kim, 2016).

$$MAPE = \frac{1}{n} \Sigma_{i=1}^{n} \left| \frac{\hat{y}_{i-y_i}}{\hat{y}_i} \times 100 \right| \quad (2.10)$$

Keterangan :

$\hat{y}_i$ = hasil dari prediksi.

y_i = nilai aktual

n = jumlah banyaknya data yang digunakan

Dengan menggunakan evaluasi MAPE memiliki fungsi apabila ukuran atau besar variabel yang digunakan pada prediksi dianggap penting dalam melakukan suatu evaluasi penetapan prediksi. *Mean Absolute Percentage Error* merupakan perhitungan nilai evaluasi yang dapat dengan mudah dipahami oleh berbagai pengguna karena dengan menggunakan MAPE perhitungan tersebut menggunakan nilai persentasi sebagai pernyataan tingkat *error* yang telah dihasilkan dalam melakukan prediksi. Hal tersebut memberikan hasil yang cukup aktual dalam melekaukan prediksi yang sesungguhnya (Kim & Kim, 2016).

Berdasarkan nilai mape yang dihasilkan terdapat beberapa kriteria nilai MAPE (Chang, et al., 2007). Kriteria tersebut akan ditunjukkan pada Tabel 2.3.

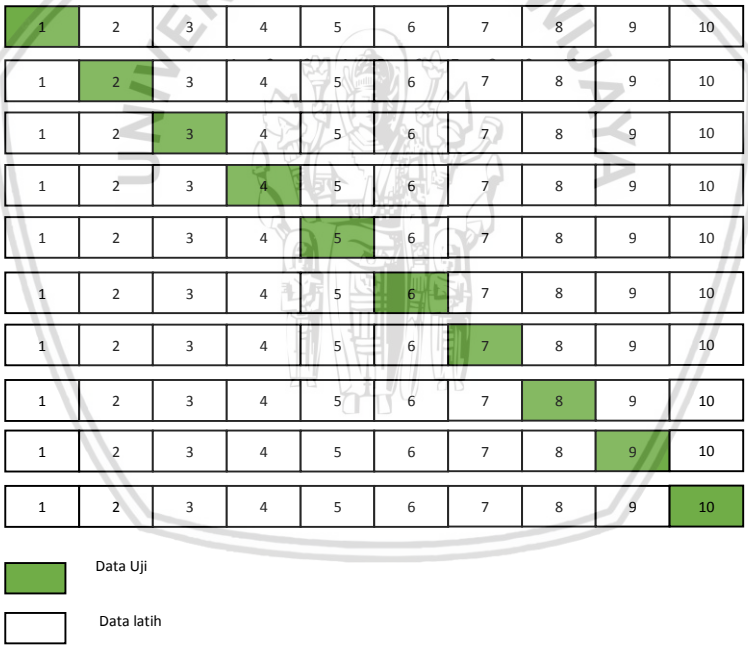
Tabel 0.2 Tabel Kriteria MAPE

Nilai MAPE	Kriteria
<10%	Sangat Baik
10%-20%	Baik
20%-50%	Cukup
>50%	Buruk

1.11 *K-Fold Cross Validation*

Cross Validation adalah suatu teknik untuk melakukan validasi terhadap sistem agar mengetahui keakuratan dari suatu model berdasarkan dataset tertentu. Salah satu teknik yang ada pada *cross validation* adalah *k-fold cross validation. K-fold cross validation* dilakukan dengan cara membagi suatu data ke dalam nilai k *fold*/segmen yang sama dan kemudian dilakukan iterasi sebanyak n kali. Masing-masing iterasi yang dijalankan menggunakan suatu data partisi ke-K yang digunakan untuk data uji dan partisi sisa lainnya digunakan untuk data latih. Data yang digunakan untuk pengujian haruslah dilakukan secara bergiliran dimulai dari nilai k pertama hingga k terakhir. Hal tersebut bertujuan agar masing-masing data pada *fold* mempunyai kesempatan untuk dilakukan uji coba (Kohavi, 2015).

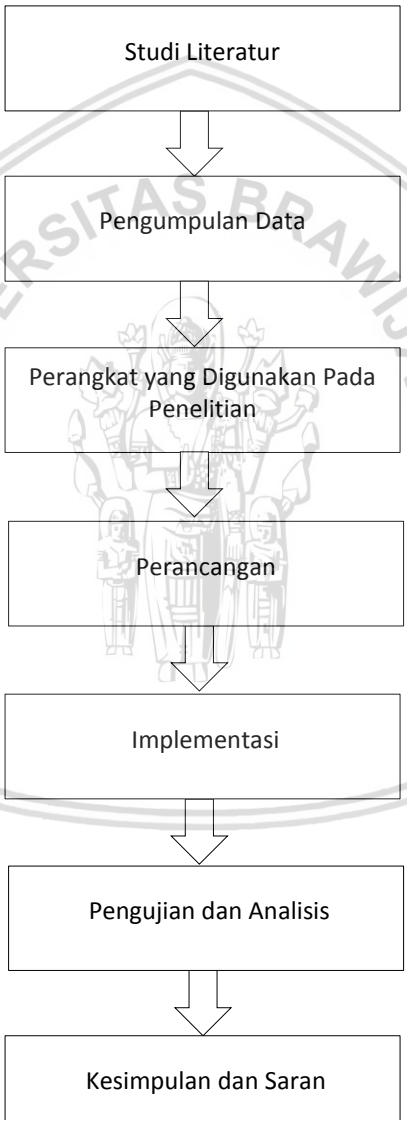
Pada penggunaan *k-fold cross validation* langkah pertama yang dilakukan adalah menentukan banyaknya segmen/*fold.* Nilai *fold* yang biasa digunakan dan direkomendasikan dalam pemilihan suatu model terbaik adalah berjumlah 10 *fold.* Pada Gambar 2.3 akan ditunjukkan mengenai ilustrasi *fold* bernilai 10.



Gambar 0.3 Ilustrasi *k-fold cross validation fold*=10

BAB 3 METODOLOGI

Pada bab metodologi dipaparkan mengenai langkah-langkah pada penelitian estimasi hasil produksi benih tanaman kenaf pada Balai Penelitian Tanaman Pemanis Dan Serat (Balittas) menggunakan metode *Extreme Learning Machine (ELM).* Dalam metodolgi penelitian kali ini dilakukan beberapa tahap diantaranya adalah studi literatur, pengumpulan data, perangkat yang digunakan pada penelitian, perancangan, implementasi, pengujian dan analisis, serta kesimpulan. Tahapan tersebut akan di Gambarkan dengan diagram alir pada Gambar 3.1.



Gambar 3.1 Diagram Alir Metodologi Penelitian

3.1 Studi Literatur

Dalam tahap ini akan di jelaskan mengenai dasar teori yang dilakukan dengan cara mengumpulkan, mempelajari, dan menganalisis berbagai macam literatur yang mendukung penelitian ini yaitu estimasi produksi benih tanaman kenaf

menggunakan metode *Extreme Learning Machine (ELM).* Pada tahap studi literatur memiliki fungsi dan tujuan untuk menunjang penulis dalam proses penyusunan skripsi yang diperoleh dari sumber yang terpercaya seperti buku, jurnal, karya tulis ilmiah, penelitian sebelumnya maupun berasal dari penjelasan terhadap objek yang diteliti. Adapun beberapa literatur yang akan di pelajari antara lain adalah:

1. Jaringan Syaraf Tiruan.
2. Metode *Extreme Learning Machine* (ELM).
3. Tanaman Kenaf.
4. Benih tanaman kenaf.

3.2 Pengumpulan Data

Pada tahap pengumpulan data adalah tahapan yang dilakukan untuk mengumpulkan data-data yang akan digunakan dalam melakukan penelitian serta untuk dilakukan pengujian sistem. Adapun data-data yang digunakan pada penelitian ini antara lain adalah:

1. Data yang digunakan pada penelitian ini adalah data karakterisasi tanaman kenaf pada tahun 2013 yang dilakukan oleh peneliti dari pihak Balittas.
2. Data karakterisasi tanaman kenaf yang didapatkan adalah berupa parameter-parameter yang berfungsi untuk menentukan hasil produksi benih tanaman kenaf. Parameter tersebut antara lain adalah umur tanaman saat berbunga, diameter batang bagian bawah, jumlah kapsul masak, dan berat benih untuk 10 tanaman. Untuk berat benih 10 tanaman dari satu petak telah mewakili keseluruhan tanaman.
3. Dalam penelitian ini data yang didapatkan akan dibagi menjadi dua bagian, yaitu data untuk proses *training* serta data untuk proses *testing.*
4. Data karakterisasi tanaman kenaf secara keseluruhan akan ditunjukkan pada Lampiran A.

3.3 Perangkat yang Digunakan Pada Penelitian

Pada bab ini, memiliki tujuan untuk mengidentifikasi perangkat apa saja yang akan digunakan oleh peneliti agar tidak menyimpang dari permasalahan dan tujuan penelitian. Adapun perangkat yang digunakan pada penelitian ini adalah:

1. Spesifikasi Perangkat Keras (*Hardware)*.
    a. Intel®Core™ i3-2350M CPU @ 2.30 Hz 2.30 GHz.
    b. Memory 4.00 GB RAM 500 GB.
2. Spesifikasi Perangkat Lunak (*Software).*
    a. Sistem Operasi Windows 8 64-bit.
    b. IDE *Netbeans* 8.0.1
    c. *Microsoft Excel* 2013.
    d. *Microsoft Word* 2013.

3.4 Perancangan

Pada tahap perancangan akan dibuat dan dirancang sedemikian rupa agar terciptanya suatu sistem yang lebih baik dari segala aspek. Baik dalam segi model maupun dari aristekturnya agar mempermudah implementasi maupun pada saat pengujian. Sistem memiliki beberapa langkah yang telah disesuaikan oleh arsitektur yang ada. Langkah-langkah tersebut antara lain adalah:

1. Perancangan *manualisasi* bertujuan untuk menyelesaikan masalah dengan cara membuat studi kasus perhitungan metode *Extreme Learning Machine* (ELM) secara sederhana dengan menggunakan dataset yang telah tersedia.
2. Perancangan pengujian sistem yang dilakukan berupa pengujian jumlah data *training* terhadap data *testing,* jumlah *neuron* pada *hidden layer,* fungsi aktivasi dan pengujian validasi sistem menggunakan *k-fold cross validation.*

3.5 Implementasi

Pada tahap implementasi akan dilakukan pembuatan program menggunakan bahasa pemrograman Java. Dalam mengimplementasikan program, penulis mengacu pada perancangan sistem yaitu menggunakan metode *Extreme Learning Machine* (ELM). Pada tahap implementasi akan ditunjukkan mengenai pembuatan *user interface* pada sistem dan penerapan metode *Extreme Learning Machine* menggunakan bahasa pemrograman java.

3.6 Pengujian dan Analisis

Pada pengujian sistem dilakukan guna menunjukkan bahwa sistem akan bekerja dengan baik sesuai dengan yang diinginkan. Adapun pengujian yang dilakukan antara lain adalah:

1. Uji coba persentase perbandingan jumlah data *training* terhadap data *testing.*
2. Uji coba jumlah *neuron* pada *hidden layer.*
3. Uji coba fungsi aktivasi.
4. Uji coba *k-fold cross validation.*

Dalam mengevaluasi sistem estimasi hasil produksi benih tanaman kenaf, maka digunakan perhitungan *Mean Average Percentage Error* (MAPE).

3.7 Kesimpulan dan Saran

Kesimpulan dilakukan apabila telah menyelesaikan beberapa tahap sebelumnya yaitu perancangan, implementasi, serta pengujian dan analis. Dalam tahap akhir ini tentunya telah dipertimbangkan berdasarkan rumusan masalah yang telah dibuat sebelumnya. Setelah semua tahap telah di lakukan maka penulis akan memberikan saran agar penelitian selanjutnya dapat lebih baik dari sebelumnya.

BAB 4 PERANCANGAN

4.1 Formulasi Permasalahan

Penelitian kali ini akan menyelesaikan suatu permasalahan mengenai estimasi hasil produksi benih tanaman kenaf pada Balittas. Hal tersebut dilakukan agar terjadi ketepatan dalam penanganan produksi benih yang akan dihasilkan. Estimasi yang akan dilakukan adalah dengan menggunakan metode *Extreme Learning Machine.* Dengan digunakannya metode ELM, akan menghasilkan estimasi dengan tingkat akurasi yang baik. Selain itu ELM mampu menghasilkan *learning speed* yang cukup baik.

Sistem yang akan dibangun terdiri dari masukan berupa data karakterisasi tanaman kenaf. Kemudian data tersebut tersedia dalam format xls. Terdapat parameter perhitungan yang dimiliki diantaranya adalah banyaknya data uji serta data latih, banyaknya *neuron* pada *hidden layer,* serta fungsi aktivasi yang akan digunakan*. Input weight* ditentukan dengan jumlah *input neuron* pada *hidden layer.* Dalam penelitian yang dilakukan, terdapat 4 *input* sistem diantaranya adalah: umur tanaman pada saat berbunga, diameter batang bagian bawah, jumlah kapsul masak, berat benih 10 tanaman. Pada proses perhitungannya, ELM membagi data menjadi proses *training* dan *testing.* Sebelum masuk ke dalam proses *training,* hal yang dilakukan adalah dengan melakukan normalisasi data, yakni mengubah data supaya menghasilkan persebaran data secara seimbang. Setelah dilakukan normalisasi pada data maka tahap selanjutnya adalah melakukan *training.* Pada proses *training* menggunakan perhitungan *output hidden layer* dengan fungsi aktivasi. Setelah itu, hasil perhitungan akan dilanjutkan agar memperoleh keluaran berupa *output weight* yang akan dilanjutkan ke tahap *testing* untuk mengkalkulasi keluaran yang dihasilkan oleh sistem dan menghasilkan nilai estimasi atau *output layer*. Dengan menggunakan MAPE akan dilakukan evaluasi untuk dihitung tingkat kesalahan dari sistem yang telah dihasilkan.

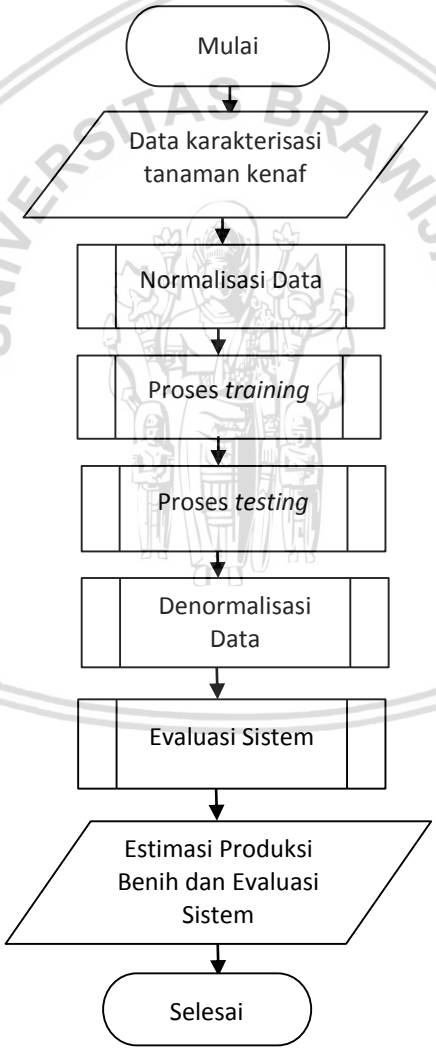
4.2 Alur Proses Algoritma *Extreme Learning Machine* (ELM)

Pada bagian ini akan dijelaskan mengenai Gambaran alur sistem estimasi hasil produksi benih tanaman kenaf menggunakan metode *Extreme Learning Machine.* Alur yang akan dijelaskan dimulai dari proses normalisasi data, proses *training,* proses *testing,* denormalisasi data, serta melakukan evaluasi terhadap sistem menggunakan MAPE. Pada Gambar 4.1 akan ditunjukkan mengenai alur dari sistem estimasi hasil produksi benih menggunakan metode *Extreme Learning Machine.*

Berdasarkan alur proses pada Gambar 4.1 akan dijelaskan tahapan-tahapan dari diagram tersebut. Berikut adalah penjelasannya.

1. Sistem mendapatkan *input*an berupa data karakterisasi tanaman kenaf.

2. Proses normalisasi data *training* dan *testing* menggunakan *min-max normalization* berdasarkan Persamaan 2.4 dan akan ditunjukkan pada Gambar 4.2
3. Proses perhitungan *training* dilakukan setelah tahap normalisasi data. Pada proses ini akan ditunjukkan pada Gambar 4.3. Pada tahapan ini akan menghasilkan nilai *output weight*.
4. Proses *testing* dilakukan setelah proses *training* yang akan ditunjukkan pada Gambar 4.12. Pada tahapan ini akan menghasilkan hasil estimasi sebelum masuk ke tahap denormalisasi data.
5. Proses denormalisasi data menghasilkan nilai estimasi dengan menggunakan metode ELM yang ditunjukkan pada Gambar 4.14.
6. Dilakukan evaluasi terhadap sistem dengan menggunakan nilai MAPE yang akan ditunjukkan pada Gambar 4.15.



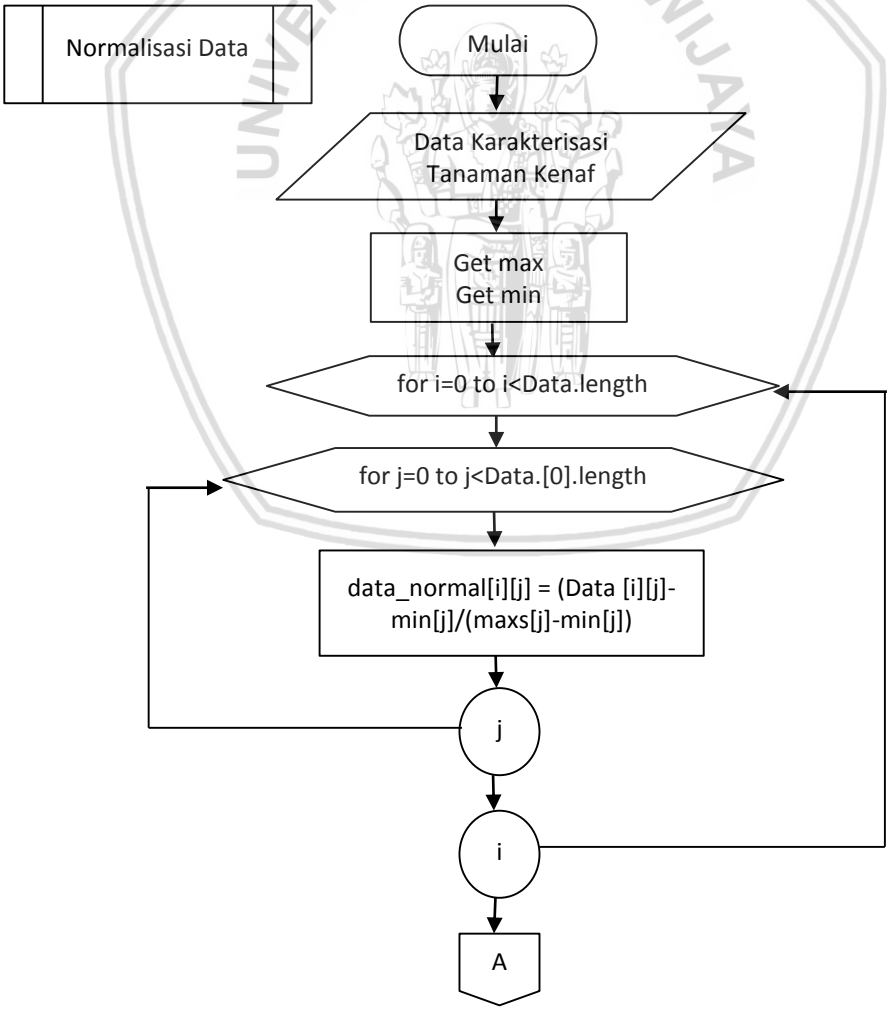
Gambar 4.1 Diagram Alir Sistem Estimasi Produksi Benih Tanaman Kenaf Menggunakan ELM

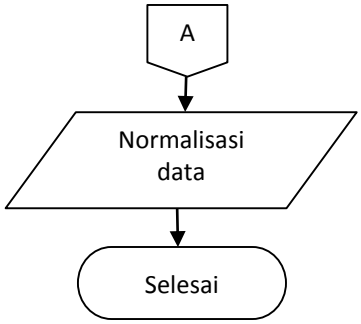
4.2.1 Normalisasi Data

Pada proses normalisasi dilakukan standarisasi terhadap keseluruhan data yang akan digunakan sehingga data tersebut berada pada *range* tertentu. Normalisasi data berfungsi agar struktur pada data bernilai konsisten sehingga menghasilkan data yang cukup baik. Hal tersebut dikarenakan kualitas pada data dapat berpengaruh terhadap nilai *output* pada sistem yang akan dihasilkan. Normalisasi yang digunakan pada penelitian ini adalah *min-max normalization* berdasarkan Persamaan 2.4. Pada Gambar 4.2 akan ditunjukkan mengenai diagram alir proses normalisasi data

Berdasarkan diagram alir normalisasi data pada Gambar 4.2 terdapat tahapan-tahapan normalisasi data diantaranya adalah sebagai berikut.

1. Sistem mendapatkan *input* berupa data karakterisasi tanaman kenaf.
2. Mencari nilai maksimal dan minimal dari masing-masing fitur pada data.
3. Melakukan perhitungan normalisasi data menggunakan *min-max normalization* berdasarkan Persamaan 2.4.
4. Didapatkan nilai hasil normalisasi.

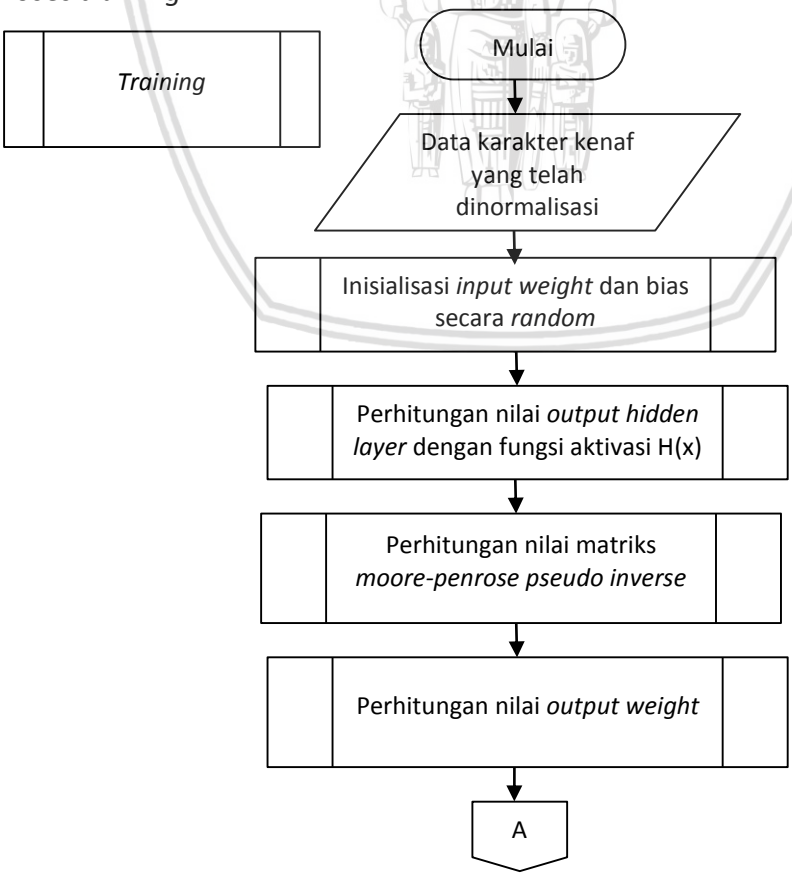


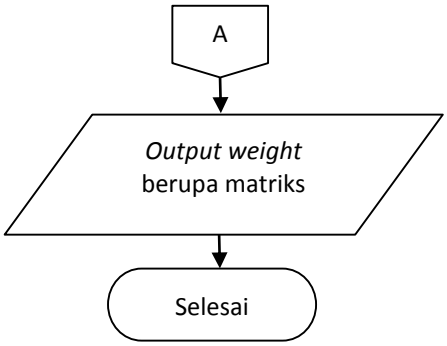


Gambar 4.2 Diagram Alir Normalisasi Data

4.2.2 Proses Perhitungan *Training*

Sebelum dilakukan proses estimasi terhadap produksi benih tanaman kenaf, langkah yang harus dilakukan untuk mendapatkan nilai *output weight* adalah melakukan proses *training.* Tahapan-tahapan yang dilakukan oleh proses *training* antara lain adalah inisialisasi *input weight* dan bias secara *random,* menghitung nilai *output hidden layer* dengan fungsi aktivasi. Dari hasil perhitungan *output hidden layer,* kemudian dilakukan perhitungan matriks *moore-penrose pseudo inverse.* Dari hasil matriks *moore-penrose pseudo inverse* didapatkan nilai *output weight.* Setelah mendapatkan nilai *output weight* maka dilanjutkan kedalam perhitungan proses *testing.* Pada Gambar 4.3 ditunjukkan mengenai diagram alir proses *training.*





Gambar 4.3 Diagram Alir Training

Berdasarkan diagram alir proses *training* pada Gambar 4.3 terdapat tahapan-tahapan *training* adalah sebagai berikut.

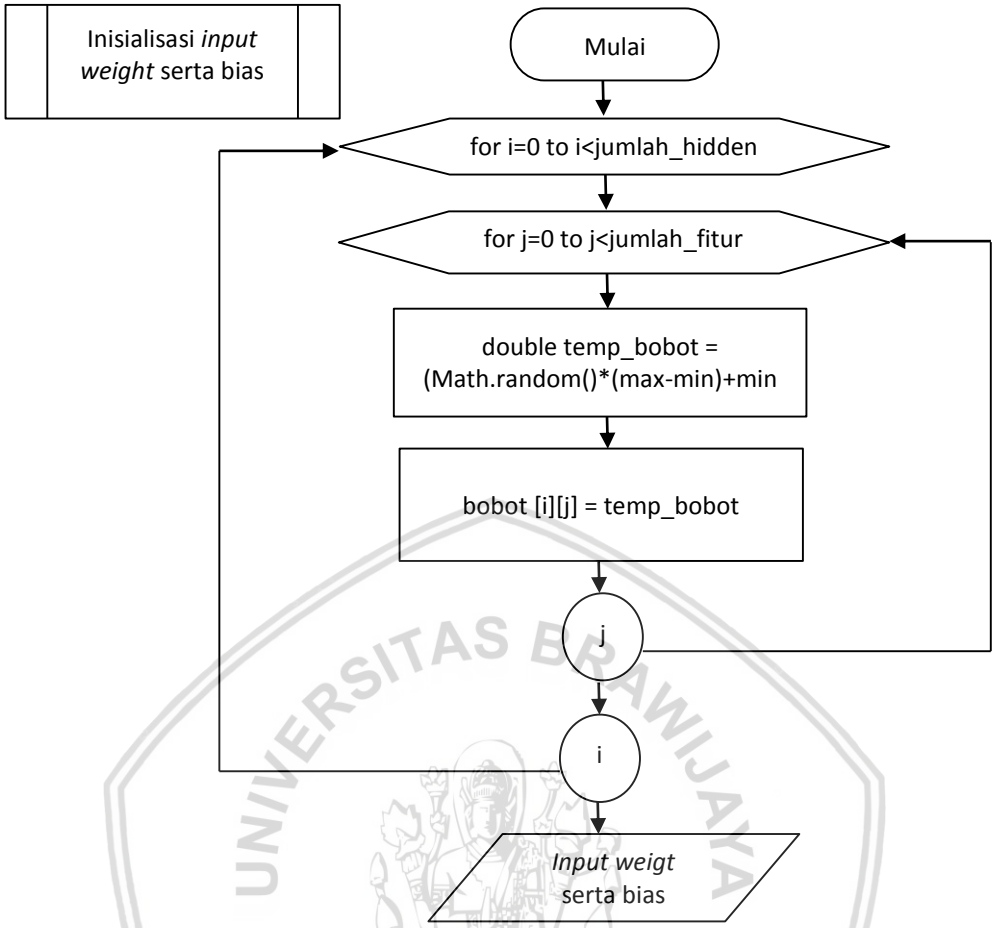
1. Melakukan inisialisasi terhadap *input weight* dan bias secara *random.*
2. Melakukan perhitungan *output hidden layer* berdasarkan Persamaan 2.5. Kemudian dilanjutkan dengan menghitung fungsi aktivasi menggunakan Persamaan 2.1 hingga Persamaan 2.3. Diagram alir perhitungan *output weight* akan ditunjukkan pada Gambar 4.11
3. Melakukan perhitungan matriks *moore-penrose pseudo inverse* berdasarkan Persamaan 2.6. Pada Gambar 4.9 akan ditunjukkan mengenai diagram alir matriks *moore-penrose pseudo inverse.*
4. Melakukan perhitungan *output weight* berdasarkan Persamaan 2.7. Hasil dari *output weight* selanjutnya akan diproses pada proses *testing.* Pada Gambar 4.11 akan ditunjukkan mengenai diagram alir *output weight.*
5. Didapatkan nilai *output weight* berupa matriks.

4.2.2.1 Inisialisasi *Input Weight* dan Bias

Proses inisialisasi pada *input weight* serta bias dilakukan dengan cara me*random* data dengan *range* sebesar -1 sampai dengan 1. Nilai *input weight* serta bias disesuaikan dengan jumlah *neuron* pada *hidden layer.* Pada Gambar 4.4 akan ditunjukkan mengenai diagram alir inisialisasi *input weight* serta bias.

Berdasarkan diagram alir inisialisasi *input weight* dan bias pada Gambar 4.4, terdapat tahapan-tahapan inisialisasi nilai *input weight* serta bias adalah sebagai berikut.

1. Proses perulangan dilakukan untuk mendapatkan nilai *weight* dan bias secara random.
2. Inisialiasi *input weight* serta bias dilakukan dengan cara merandom nilai dengan *range* -1 sampai dengan 1.
3. Didapatkan nilai inisialisasi *input weight* serta bias.



Gambar 4.4 Inisialisasi *Input Weight* dan Bias

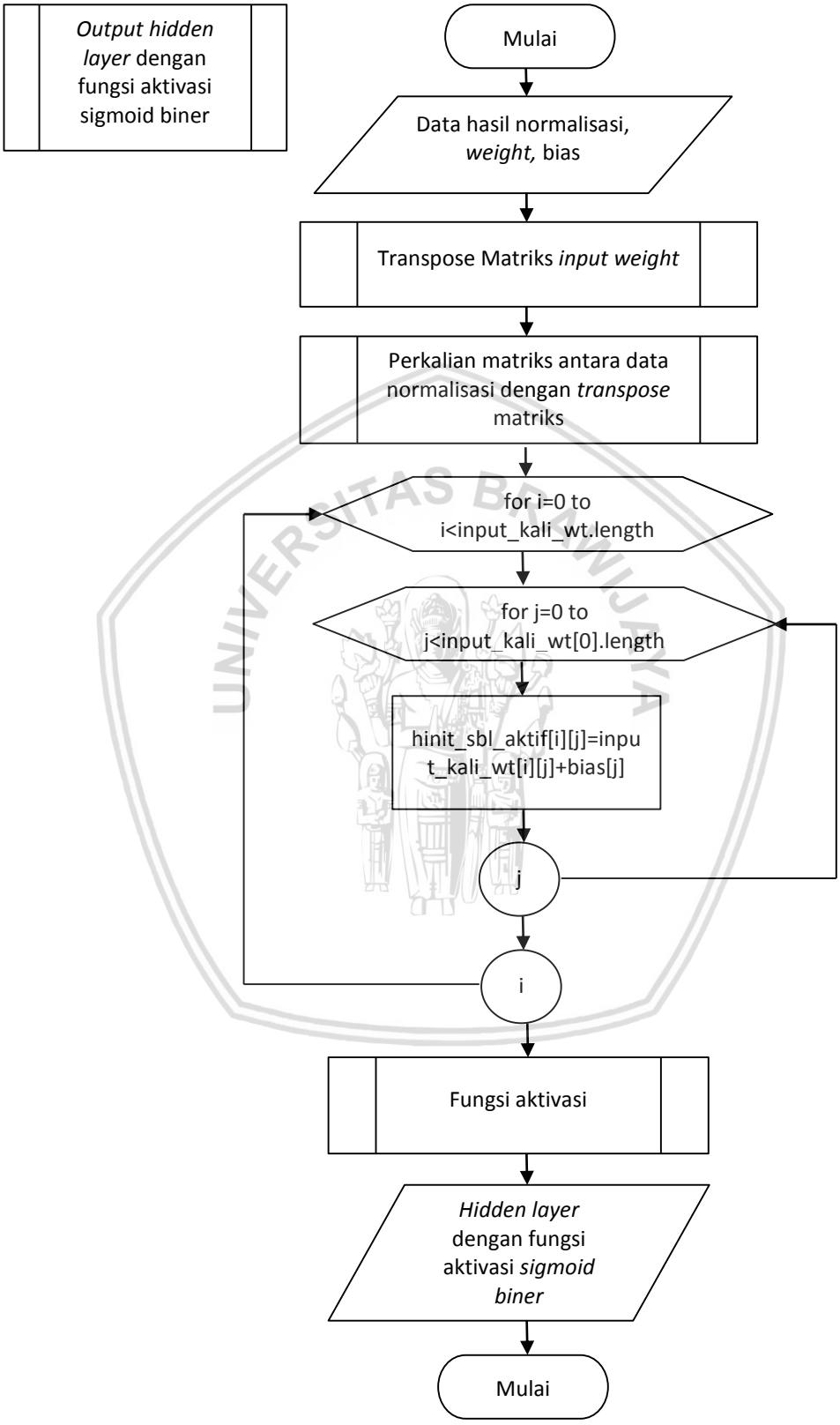
4.2.2.2 Proses Perhitungan *Output Hidden Layer* Dengan *Fungsi Aktivasi Sigmoid Biner*

Pada proses ini berfungsi untuk menghitung nilai *output hidden layer* berdasarkan Persamaan 2.5. Kemudian setelah mendapatkan nilai *output hidden layer* maka dihitung dengan menggunakan fungsi aktivasi berdasarkan Persamaan 2.1 hingga Persamaan 2.3. Pada Gambar 4.5 akan ditunjukkan mengenai diagram alir *output hidden layer* dengan fungsi aktivasi.

Berdasarkan diagram alir proses perhitungan *output hidden layer* dengan fungsi aktivasi pada Gambar 4.5 terdapat tahapan-tahapan menghitung nilai *output hidden layer* dengan fungsi aktivasi *sigmoid biner* diantaranya adalah:

1. Nilai masukkan berupa nilai yang telah dinormalisasi, *input weight,* bias.
2. Dilakukan transpose matriks nilai *input weight.*
3. Melakukan perhitungan perkalian matriks nilai yang telah dinormalisasi dengan matriks *input weight* yang telah ditranspose.
4. Dilakukan perhitungan *output hidden layer* berdasarkan Persamaan 2.5 kemudian dihitung menggunakan fungsi aktivasi *sigmoid biner* berdasarkan Persamaan 2.1.

5. Mendapatkan nilai keluaran berupa *output hidden layer* dengan fungsi aktivasi *sigmoid biner.*



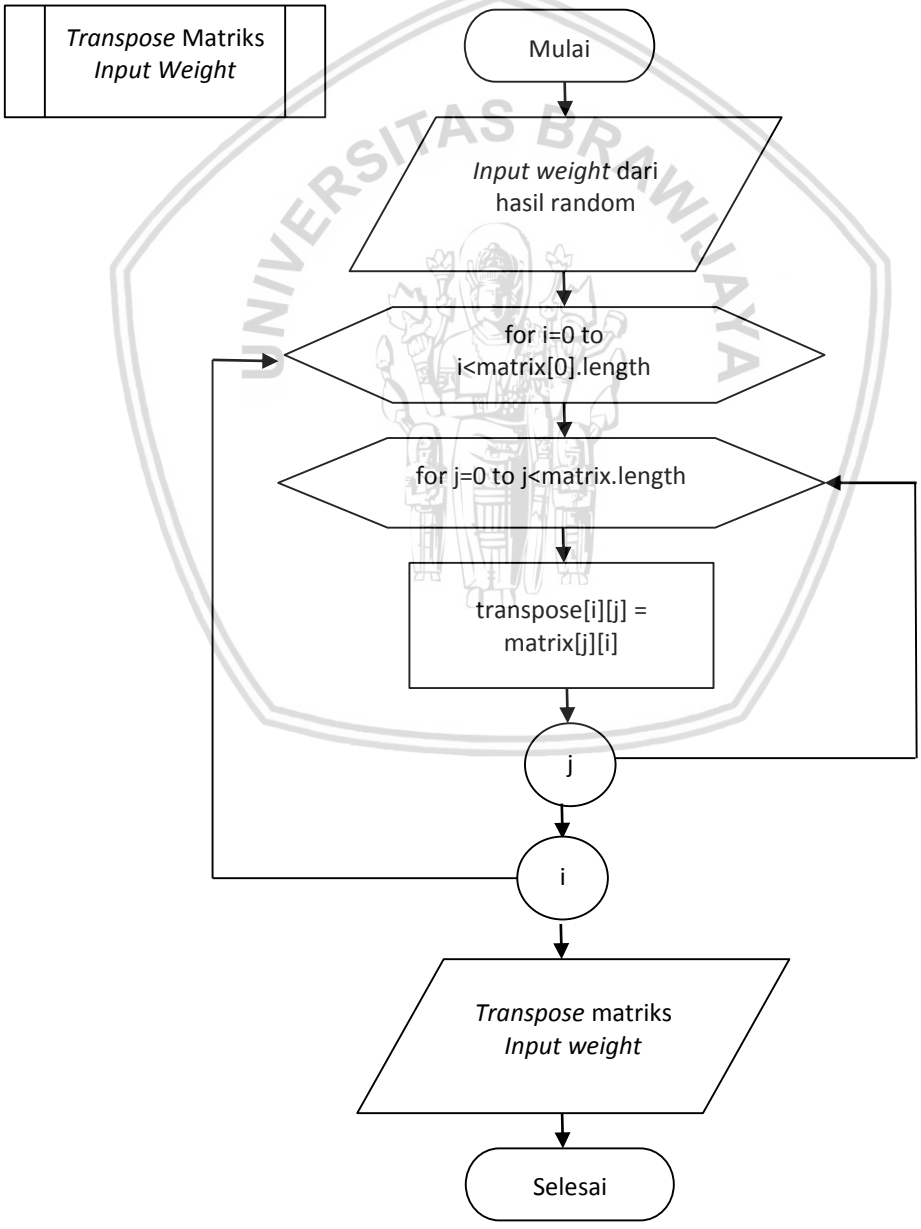
Gambar 4.5 Diagram Alir *Output Hidden Layer* dengan Fungsi Aktivasi Sigmoid Biner

4.2.2.3 *Transpose Matriks Input Weight*

Setelah mendapatkan nilai *input weight* secara random maka dilakukan proses *transpose* matriks *input weight* untuk melakukan proses perhitungan *output hidden layer.* Pada Gambar 4.6 akan ditunjukkan mengenai diagram alir *transpose* matriks *input weight.*

Berdasarkan diagram alir *transpose* matriks pada Gambar 4.6 terdapat tahapan-tahapan sebagai berikut.

1. Nilai masukan berupa *input weight* yang didapatkan dari hasil *random* data dengan *range* -1 sampai dengan 1.
2. Proses perulangan dilakukan terhadap data dengan cara merubah baris menjadi kolom pada data.
3. Didapatkan hasil berupa nilai matriks *transpose input weight*.



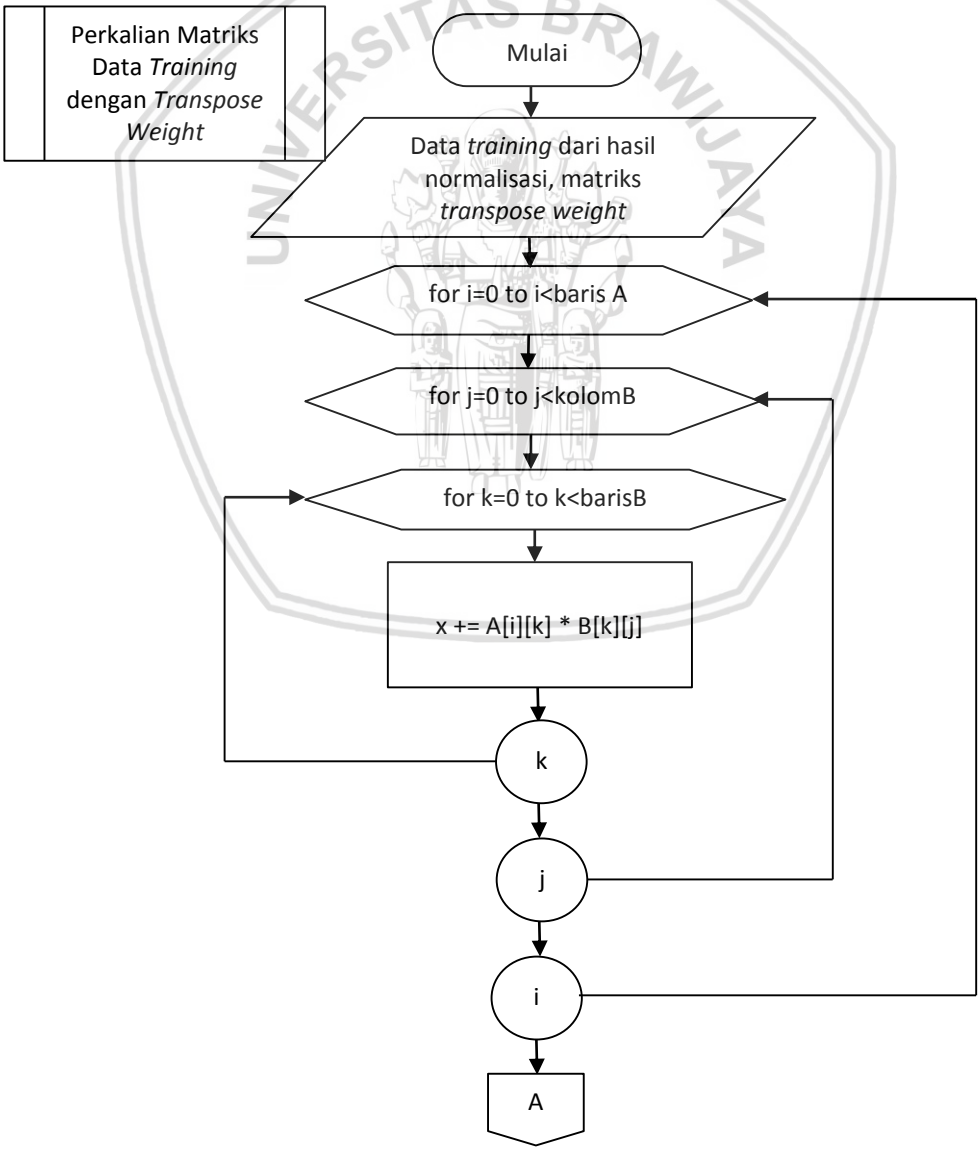
Gambar 4.6 Diagram Alir *Transpose* Matriks *Input Weight*

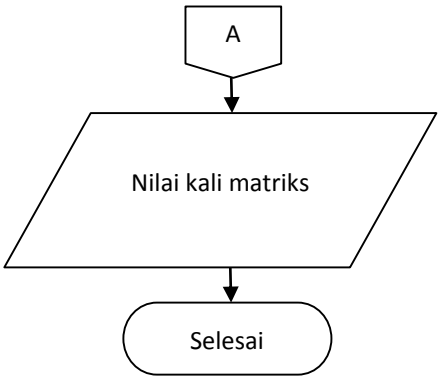
4.2.2.4 Perkalian Matriks Data *Training* dengan *Transpose Input Weight*

Setelah dilakukan normalisasi pada data, kemudian melakukan inisialisasi terhadap *input weight* secara *random* dengan *range -1* sampai dengan 1. Dari hasil inisilisasi *weight* tersebut selanjutnya dilakukan *transpose* matriks. Kemudian dilakukan perkalian antara data yang sudah dinormalisasi dengan *transpose* matriks. Dari Gambar 4.7 akan ditunjukkan mengenai diagram alir perkalian matriks antara data *training* dengan nilai *transpose weight.*

Berdasarkan diagram perkalian matriks data *training* dengan nilai *transpose weight* pada Gambar 4.7 terdapat tahapan-tahapan sebagai berikut.

1. Nilai masukkan berupa data *training* yang telah dinormalisasi, nilai *weight* yang telah dilakukan *transpose.*
2. Dilakukan perulangan terhadap baris dan kolom serta dilakukan perkalian matriks antara data *training* dengan *transpose weight.*
3. Didapatkan nilai berupa hasil kali matriks data *training* dengan *transpose*

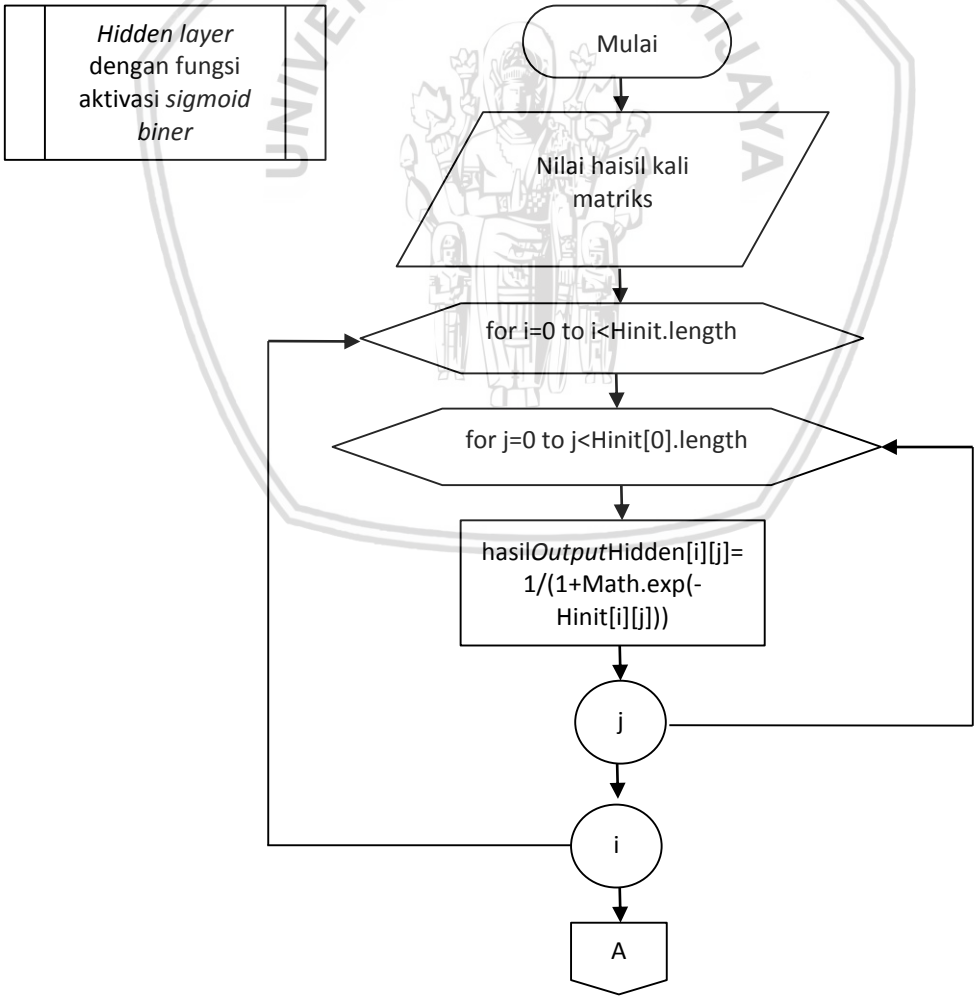


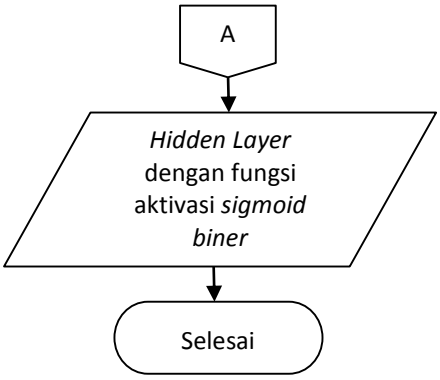


Gambar 4.7 Diagram Alir Perkalian Matriks Data *Training* dengan *Transpose Weight*

4.2.2.5 Perhitungan Fungsi Aktivasi *Sigmoid Biner*

Perhitungan fungsi aktivasi dihitung berdasarkan nilai masukkan yang diambil dari hasil perkalian matriks data *training* dengan *transpose weight.* Perhitungan fungsi aktivasi berada pada Persamaan 2.1 hingga Persamaan 2.3. Pada Gambar 4.8 akan ditunjukkan Gambar mengenai alur dari fungsi aktivasi sigmoid biner.





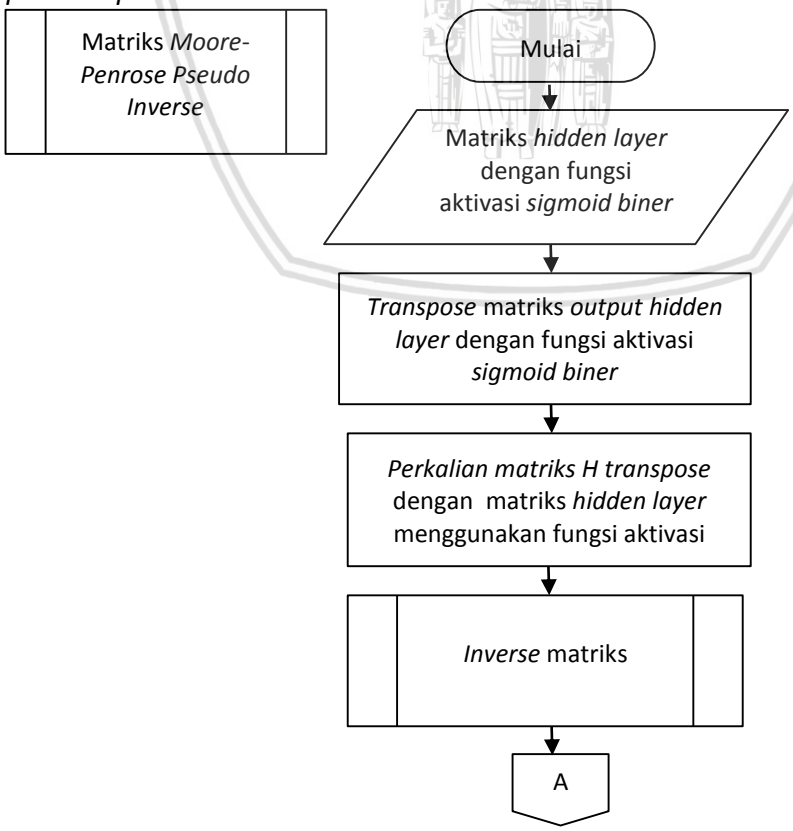
Gambar 4.8 Diagram Alir Fungsi Aktivasi *Sigmoid Biner*

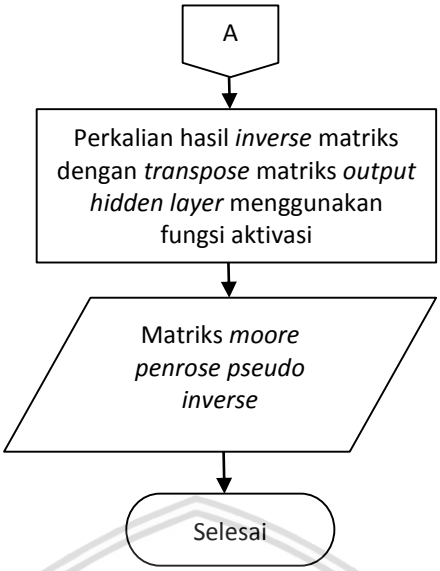
Berdasarkan diagram alir fungsi aktivasi pada Gambar 4.8 terdapat tahapan-tahapan sebagai berikut.

1. Nilai masukkan didapatkan dari hasil perkalian matriks data *training* dengan matriks *transpose weight.*
2. Hasil perkalian matriks tersebut kemudian dihitung menggunakan fungsi aktivasi menggunakan *sigmoid biner* berdasarkan Persamaan 2.1.
3. Nilai keluaran berupa *hidden layer* dengan fungsi aktivasi.

4.2.2.6 Proses Perhitungan Matriks *Moore-Penrose Pseudo Inverse*

Untuk mendapatkan nilai *output weight* dilakukan perkalian matriks yang disebut dengan matriks *moore-penrose pseudo inverse* berdasarkan Persamaan 2.6. Pada Gambar 4.9 akan ditunjukkan mengenai diagram alir matriks *moore-penrose pseudo inverse.*



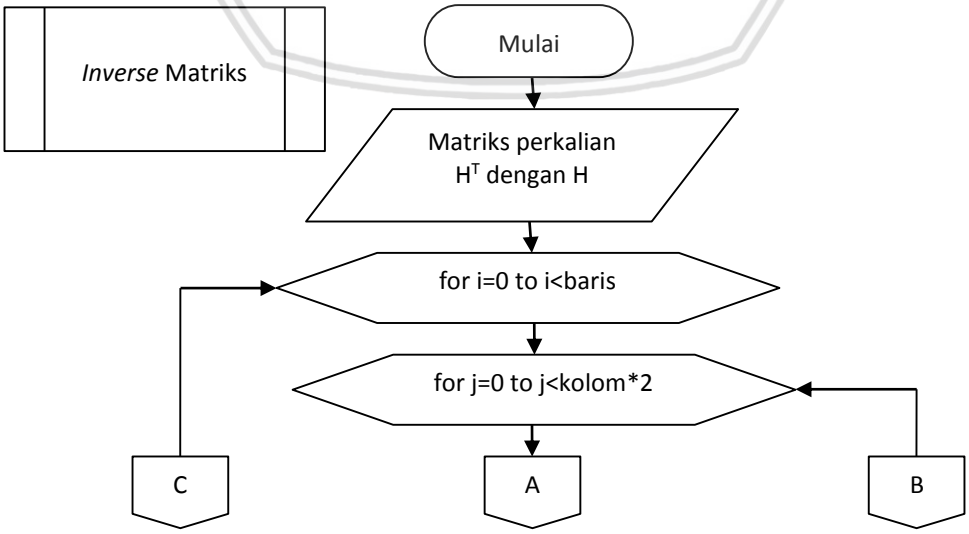


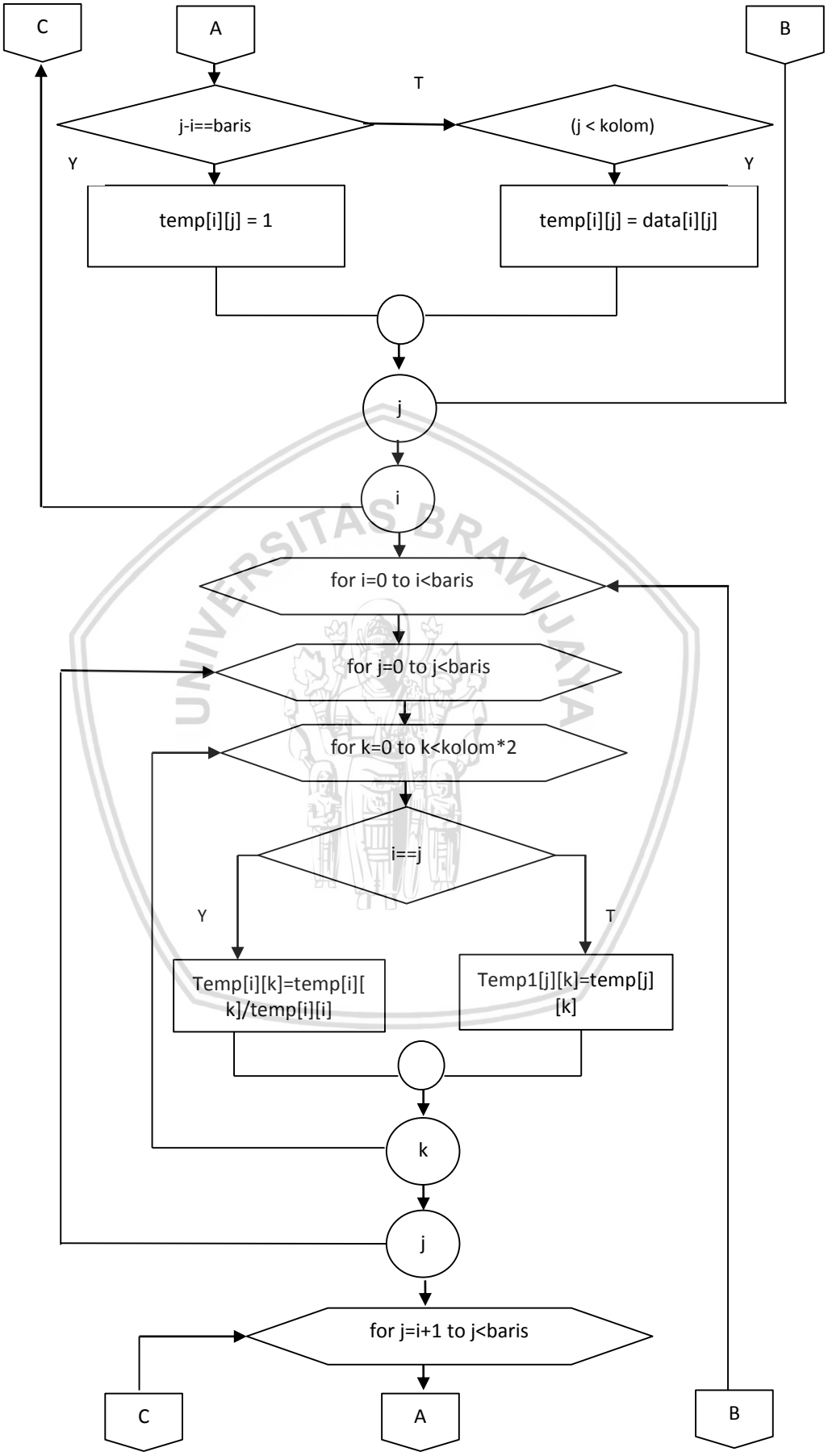
Gambar 4.9 Diagram Alir Matriks *Moore Penrose Pseudo Inverse*

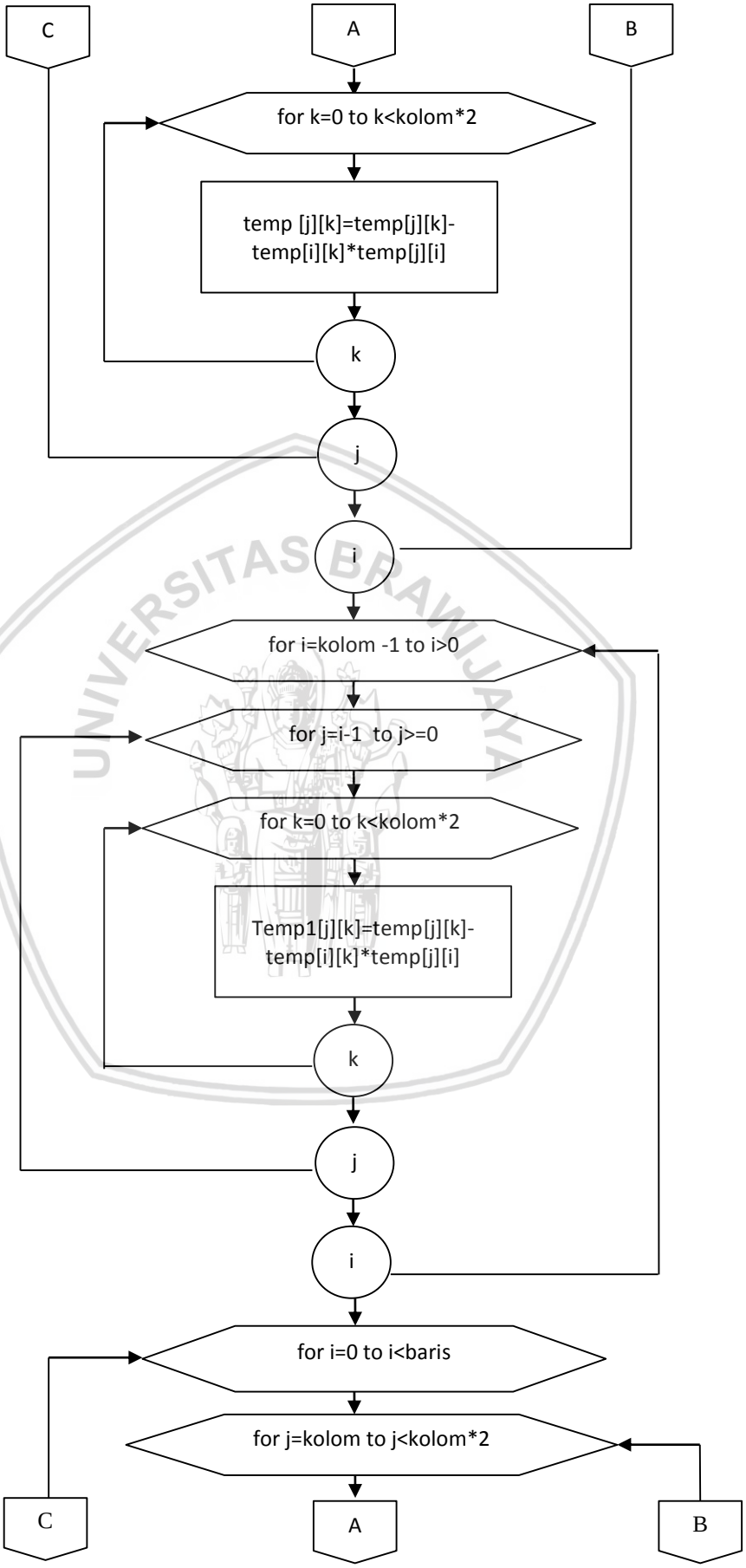
Berdasarkan diagram alir matriks *moore-penrose pseudo inverse* pada Gambar 4.9 terdapat tahapan-tahapan sebagai berikut.

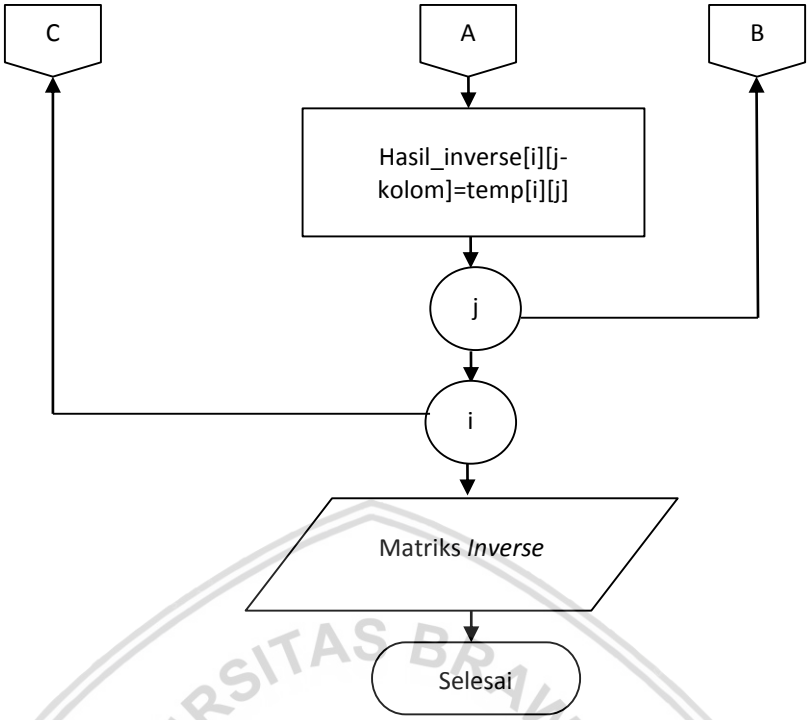
1. Nilai masukkan berupa hasil dari matriks *output hidden layer* dengan fungsi aktivasi *sigmoid biner.*
2. *Transpose* matriks *output hidden layer* dikalikan dengan matriks *output hidden layer* dengan fungsi aktivasi *sigmoid biner.*
3. Setelah mendapatkan keluaran berdasarkan perkalian matriks sebelumnya kemudian dilakukan *inverse* matriks.
4. Setelah mendapatkan hasil *inverse* matriks kemudian dikalikan dengan *matriks transpose output hidden layer.*

4.2.2.7 *Inverse* Matriks




C
A
B
T
j-i==baris
(j < kolom)
Y
Y
temp[i][j] = 1
temp[i][j] = data[i][j]
j
i
for i=0 to i<baris
for j=0 to j<baris
for k=0 to k<kolom*2
i==j
Y
T
Temp[i][k]=temp[i][k]/temp[i][i]
Temp1[j][k]=temp[j][k]
k
j
for j=i+1 to j<baris
C
A
B


C
A
B
for k=0 to k<kolom*2
temp [j][k]=temp[j][k]-
temp[i][k]*temp[j][i]
k
j
i
for i=kolom -1 to i>0
for j=i-1 to j>=0
for k=0 to k<kolom*2
Temp1[j][k]=temp[j][k]-
temp[i][k]*temp[j][i]
k
j
i
for i=0 to i<baris
for j=kolom to j<kolom*2
C
A
B



Gambar 4.10 Diagram Alir *Inverse* Matriks

Berdasarkan diagram alir *inverse* matriks pada Gambar 4.10 terdapat tahapan-tahapan sebagai berikut.

1. Proses perulangan dilakukan untuk mendapatkan matriks identitas.
2. Melakukan cek kondisi, apakah matriks bernilai kosong atau tidak.
3. Perhitungan Matriks OBE dengan cara merubah matiks H dengan matriks identitas begitupula sebaliknya, mengubah matiks identitas menjadi matriks H.
4. Kemudian matriks H yang berubah menjadi matriks identitas menjadi matriks baru yang berasal dari perubahan matriks identitas berdasarkan hasil *inverse* matriks H.

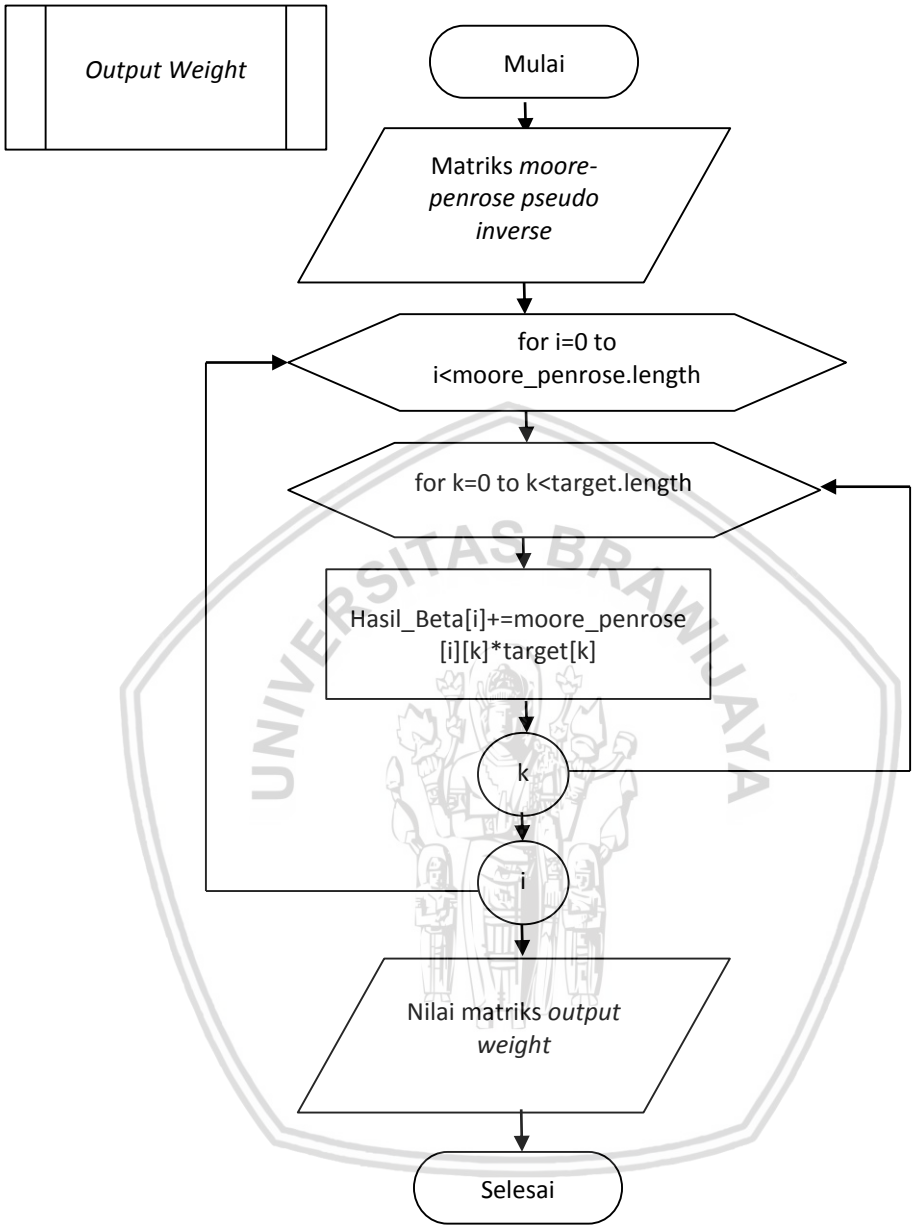
4.2.2.8 Proses *Perhitungan Output Weight*

Tahap akhir dari proses *training* adalah menghitung nilai *output weight* berdasarkan Persamaan 2.8. Hasil yang didapatkan dari perhitungan nilai *output weight* akan diproses pada proses perhitungan *testing.* Pada Gambar 4.11 akan ditunjukkan Gambar mengenai diagram alir *output weight.*

Berdasarkan diagram alir *output weight* pada Gambar 4.11 terdapat tahapan-tahapan sebagai berikut.

1. Nilai masukkan berupa matriks *moore-penrose pseudo inverse* serta data *training* yang telah dilakukan normalisasi sebelumnya.
2. Pada proses perhitungan nilai *output weight* didasarkan pada Persamaan 2.7.
3. Nilai *output weight* didapatkan dari perkalian *matriks moore-penrose pseudo inverse* dengan matriks target yang berada pada proses normalisasi untuk data *training.*

4. Didapatkan nilai *output weight* yang kemudian akan diproses pada proses *testing* untuk mendapatkan nilai *output layer.*



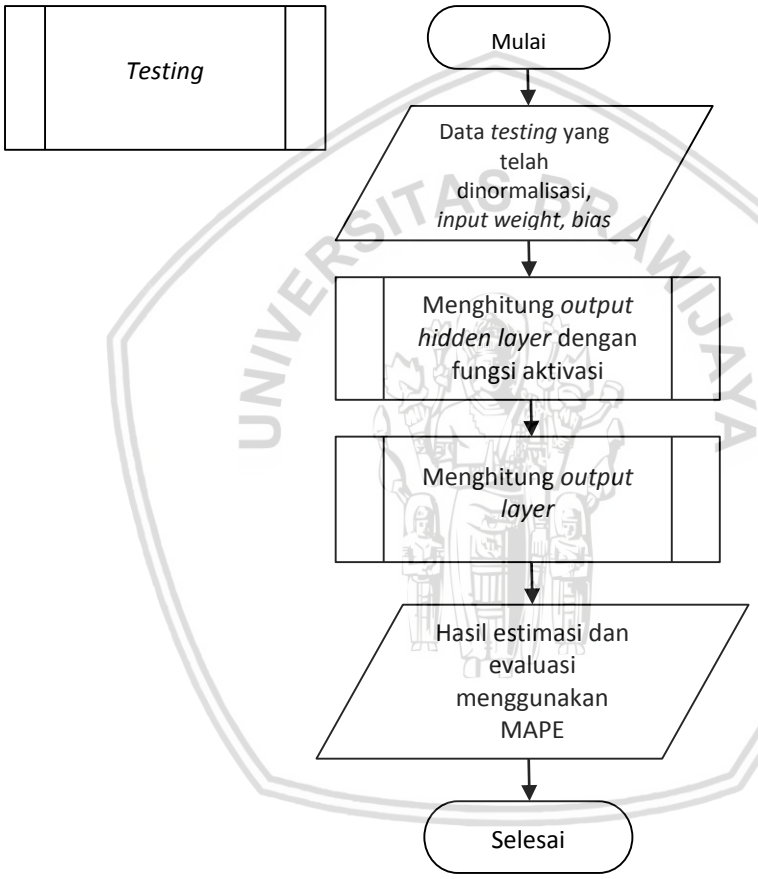
Gambar 4.11 Diagram Alir *Output Weight*

4.2.3 Proses Perhitungan *Testing*

Proses *testing* dilakukan untuk mendapatkan keluaran berupa *output layer.* Perhitungan yang dilakukan tidak jauh berbeda dengan proses *training.* Pada Gambar 4.12 akan ditunjukkan mengenai diagram alir proses *testing.*

Berdasarkan diagram alir proses *testing* pada Gambar 4.12 terdapat tahapan-tahapan sebagai berikut.

1. Nilai masukan pada proses *testing* berupa nilai *testing* yang telah dinormalisasi. *Input weight* dan bias yang digunakan sama dengan pada proses *training.*
2. Melakukkan perhitungan *hidden layer* dengan fungsi aktivasi. Proses ini serupa dengan proses yang berada pada proses *training.*
3. Melakukan perhitungan nilai *output layer* dengan *output weight* yang didapatkan dari proses *training* sebelumnya. Proses perhitungan *testing* akan ditunjukkan pada Gambar 4.12.



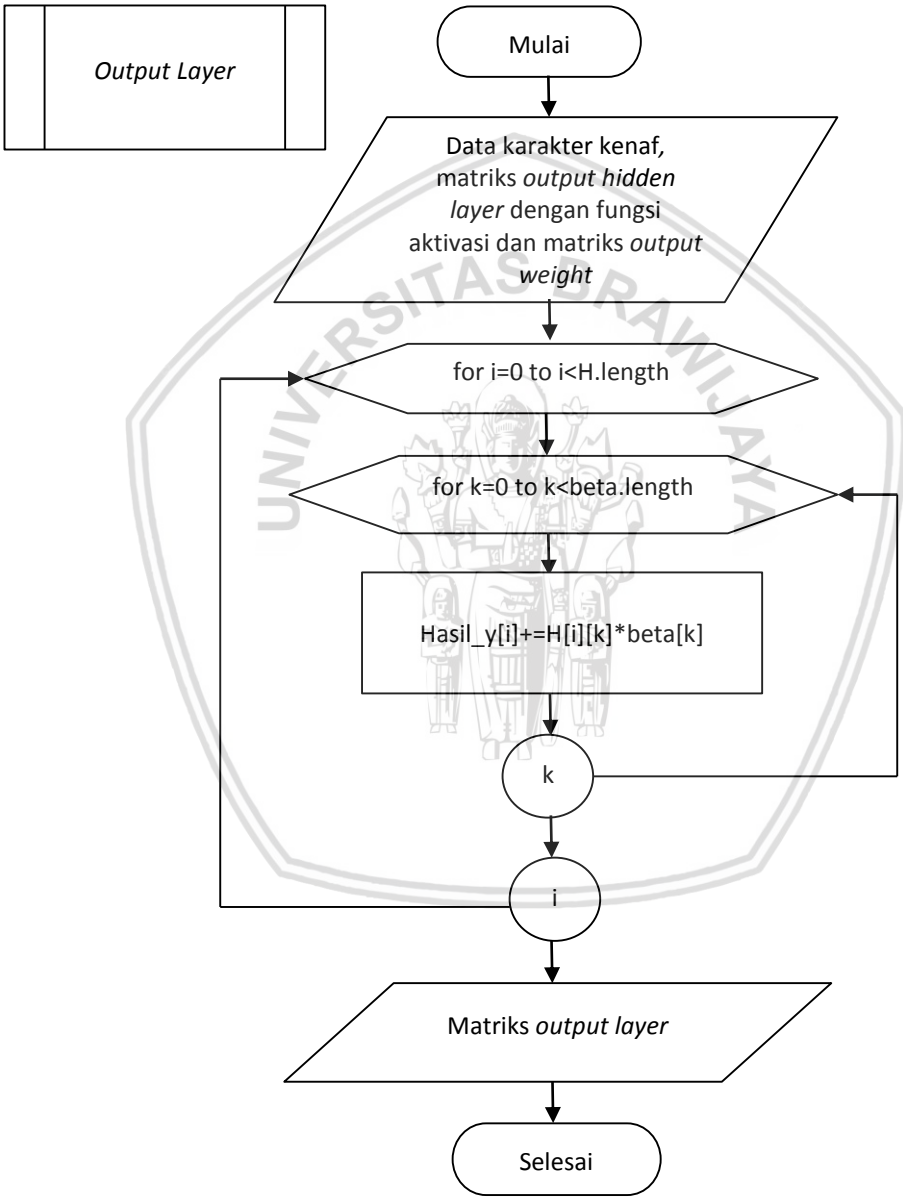
Gambar 4.12 Diagram Alir *Testing*

4.2.3.2 Proses Perhitungan *Output Layer*

Proses perhitungan *output layer* akan menghasilkan nilai estimasi produksi benih tanman kenaf sebelum dilakukan denormalisasi. Pada Gambar 4.13 akan ditunjukkan mengenai diagram alir proses perhitungan *output layer.*

Berdasarkan diagram alir proses perhitungan *output layer* pada Gambar 4.13 terdapat tahapan-tahapan sebagai berikut.

1. Nilai masukkan berupa data *testing,* matriks *hidden layer* dengan fungsi aktivasi, matriks *output weight.*
2. Agar menghasilkan matriks *output layer* maka dilakukan perkalian antara matriks *output hidden layer* menggunakan fungsi aktivasi dengan matriks *output weight.*
3. Didapatkan nilai estimasi atau *output layer* sebelum dilakukan proses denormalisasi data.



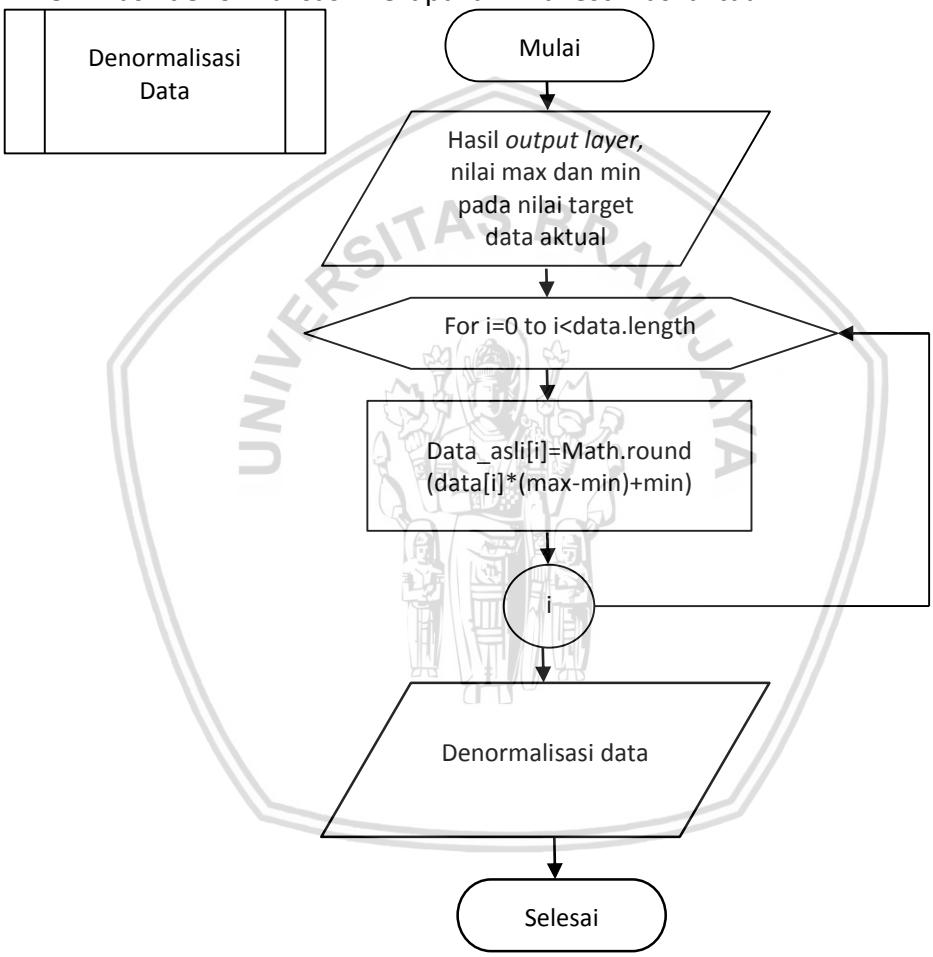
Gambar 4.13 Diagram Alir *Output Layer*

4.2.4 Denormalisasi Data

Proses perhitungan denormalisasi data dilakukan agar mengubah data hasil normalisasi menjadi data aktual. Pada Gambar 4.14 akan ditunjukkan mengenai diagram alir denormalisasi data.

Berdasarkan diagram alir proses denormalisasi data pada Gambar 4.14 terdapat tahapan-tahapan sebagai berikut.

1. Nilai masukkan berupa hasil *otput layer,* nilai minimal serta maksimal dari data.
2. Proses perhitungan denormalisasi berada pada Persamaan 2.9.
3. Hasil denormalisasi merupakan nilai estimasi aktual.



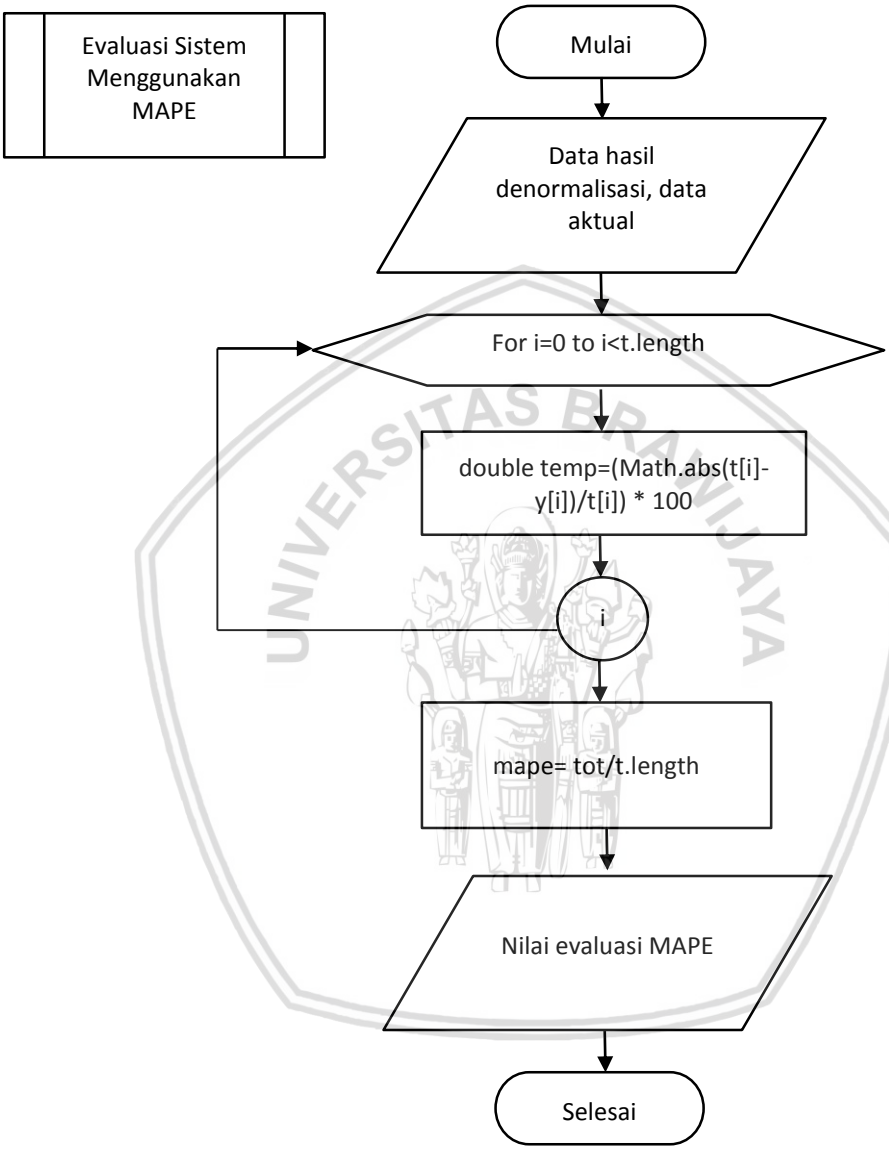
Gambar 4.14 Diagram Alir Denormalisasi Data

4.2.5 Evaluasi Sistem Menggunakan MAPE

Tahap akhir dari estimasi produksi benih tanaman kenaf adalah melakukan evaluasi terhadap sistem yang dibangun. Evaluasi sistem menggunakan MAPE berdasarkan Persamaan 2.10. Pada Gambar 4.15 akan ditunjukkan mengenai diagram alir evaluasi sistem menggunakan MAPE.

Berdasarkan diagram alir proses evaluasi sistem menggunakan MAPE pada Gambar 4.15 terdapat tahapan-tahapan sebagai berikut.

1. Nilai masukkan berupa data yang telah dilakukan denormalisasi.
2. Proses perhitungan MAPE berdasarkan Persamaan 2.11
3. Hasil evaluasi sistem berupa nilai *error* dengan satuan persen.



Gambar 4.15 Diagram Alir Evaluasi MAPE

4.3 Perhitungan Manual ELM

Dalam perhitungan manual diberikan contoh perhitungan dari gambaran sistem yang akan dibuat. Hal tersebut dilakukan supaya mengetahui keakuratan perhitungan sistem. Dalam proses perhitungan manual, terdapat langkah-langkah yang dilakukan antara lain adalah proses normalisasi data, proses *training* dan yang terakhir proses *testing.*

Adapun tahap tahap perhitungan manual algoritma ELM adalah sebagai berikut.

1. Melakukan Inisialisasi Terhadap Data yang Digunakan.
   Dalam Tabel 4.1 akan ditunjukkan mengenai inisialisasi data yang digunakan serta pola dari pembelajaran data tersebut. *Input* yang digunakan dalam proses perhitungan ini adalah umur tanaman pada saat berbunga, diameter batang bagian bawah, jumlah kapsul masak, berat benih 10 tanaman serta target estimasi.

Tabel 4.1 Data Karakterisasi Tanaman Kenaf

No	X1	X2	X3	X4	Y
1	70	13,82	9,1	30	2017
2	81	14,51	5,4	75	2326
3	46	8,53	10,17	85	2193
4	88	25,74	21,6	110	2279
5	86	13,18	134,2	95	1319
6	35	8,26	44,1	325	3006
7	49	6,68	1,4	75	2258
8	45	10,15	16,2	170	2534
9	43	6,47	0,6	65	2184
10	46	7,29	1,2	60	2163
11	67	19,01	28,6	75	1994
12	58	14,03	1,6	60	2160
13	59	10,32	0,3	65	2239
14	43	6,69	4,4	70	2173
15	82	15,46	9,6	30	2065
16	43	5,56	3,12	35	2024
17	81	21,39	5,4	175	2746
18	46	6,25	0,9	75	2250
19	89	15,41	4,8	25	2123
20	58	8,5	3,6	80	2296

Keterangan:
X1 : Umur Tanaman Pada Saat Berbunga
X2 : Diameter Batang Bagian Bawah
X3 : Jumlah Kapsul Masak
X4 : Berat Benih 10 Tanaman
T : Data Produksi Benih Tanaman Kenaf

Data sampling yang digunakan dalam proses perhitungan manual adalah sejumlah 20 data yang dibagi menjadi 15 data digunakan untuk proses *training* dan 5 data digunakan untuk proses *testing.* Data yang digunakan untuk *training* merupakan data ke-1 hingga ke-15 sedangkan data yang digunakan untuk proses *testing* dimulai dari data ke-16 hingga 20.

2. Normalisasi Data.

Pada proses normalisasi data dilakukan standarisasi data sebelum memasuki ke dalam tahap *training* dan *testing* dengan menggunakan *range* tertentu. Pada normalisasi kali ini *range* yang digunakan adalah [0, 1]. Nilai maksimal dan minimal yang didapatkan berasal dari masing-masing fitur. Adapun tahapan-tahapan proses normalisasi yang akan dijabarkan berdasarkan rumus *Min- Max Normalization* pada Persamaan 2.4 adalah sebagai berikut.

Tahap 1: Untuk setiap fitur dicari nilai maksimal dan minimalnya. Hal tersebut akan ditunjukkan pada Tabel 4.2 berikut

Tabel 4.2 Nilai *Max-Min Normalization*

	X1	X2	X3	X4	T
Max	89	25,74	134,2	325	3006
Min	35	5,56	0,3	25	1319

Tahap 2 : Dari masing-masing fitur dihitung normalisasi data mengunakan *min-max normalization.* Berikut akan dijabarkan mengenai perhitungan normalisasi berdasarkan Persamaan 2.4. Fitur yang digunakan untuk perhitungan manual adalah data x pada fitur 1 $(x_{1,1}) = 70$

$$\acute{d} = \frac{(d - \min(x))}{(\max(x) - \min(x))}$$

$$= \frac{(70-35)}{(89-35)} = 0{,}648$$

Hasil normalisasi menggunakan *min-max normalization* yang telah dilakukan pada tahap manualisasi akan ditunjukkan pada Tabel 4.3

Tabel 4.3 Hasil Normalisasi Masing-Masing Fitur

No	X1	X2	X3	X4	T
1	0,648	0,409	0,065	0,016	0,413
2	0,851	0,443	0,038	0,166	0,596
3	0,203	0,147	0,073	0,2	0,518
4	0,981	1	0,159	0,283	0,569
5	0,944	0,377	1	0,233	0
6	0	0,133	0,327	1	1
7	0,259	0,055	0,008	0,166	0,556
8	0,185	0,227	0,118	0,483	0,720
9	0,148	0,045	0,002	0,133	0,512
10	0,203	0,085	0,006	0,116	0,500
11	0,592	0,666	0,211	0,166	0,400
12	0,425	0,419	0,009	0,116	0,498
13	0,444	0,235	0	0,133	0,545
14	0,148	0,055	0,030	0,15	0,506
15	0,870	0,490	0,069	0,016	0,442

Tabel 4.3 Hasil Normalisasi Masing-Masing Fitur (Lanjutan)

No	X1	X2	X3	X4	T
16	0,148	0	0,021	0,033	0,417
17	0,851	0,784	0,030	0,5	0,845
18	0,203	0,034	0,004	0,166	0,551
19	1	0,488	0,033	0	0,476
20	0,425	0,145	0,024	0,183	0,579

3. Melakukan Inisialisasi *Input Weight* serta Bias.
   Dalam melakukan proses manualisasi menggunakan *neuron* sebanyak 2. Untuk melakukan inisialisasi terhadap *input weight* dan bias maka dilakukan *random* nilai dengan *range* antara -1 sampai dengan 1. Pada Tabel 4.4 akan ditunjukkan mengenai hasil *random* matriks *input weight.*

Tabel 4.4 Matriks *Input Weight*

w	1	2	3	4
1	0,813	-0,603	-0,919	0,378
2	0,381	0,545	0,593	0,690

Setelah dilakukan inisialisasi terhadap nilai *input weight* kemudian nilai tersebut di*transpose* sehingga menghasilkan matriks keluaran baru. Pada Tabel 4.5 akan ditunjukkan matriks *transpose input weight.*

Tabel 4.5 Hasil Transpose Matriks *Input Weight*

w transpose		
1	0,813	0,381
2	-0,603	0,545
3	-0,919	0,593
4	0,378	0,690

Dikarenakan jumlah *neuron* yang digunakan adalah sebanyak 2, maka nilai bias yang akan diinisialisasi pun sebanyak 2. Jumlah *neuron* memiliki nilai yang sama dengan bias. Pada Tabel 4.6 akan ditunjukkan matriks dari nilai bias dari hasil *random*.

Tabel 4.6 Matriks Nilai Bias

1	2
0,424	0,631

4. Proses Training
   Dalam melakukan proses *training* fungsi aktivasi dihitung menggunakan fungsi aktivasi *sigmoid biner.* Terdapat 2 *neuron* yang digunakan dalam manualisasi ini. Data yang akan digunakan dalam proses *training* adalah

sebanyak 15 data dari 20 sampel data yang digunakan. Adapun tahapan-tahapan yang dilakukan pada proses *training* antara lain adalah:

Tahap 1 : Melakukan perhitungan *output hidden layer* menggunakan fungsi aktivasi. Tahap awal yang dilakukan adalah menghitung *output hidden layer* berdasarkan Persamaan 2.5. Setelah mendapatkan hasil *output hidden layer,* maka dilakukan perhitungan menggunakan fungsi aktivasi *sigmoid biner.* Pada proses perhitungan manual, *input weight* yang digunakan ada pada Tabel 4.4 fitur 1. Untuk nilai x yang digunakan merupakan data ke-x pada kolom 1 hasil normalisasi yang ditunjukkan Pada Tabel 4.3. Bias yang digunakan ada pada fitur ($x_{1,1}$). yang ditunjukkan pada Tabel 4.6. Adapun contoh perhitungan *output hidden layer* adalah sebagai berikut berdasarkan Persamaan 2.5.

$$H_{init\ ij} = (\sum_{k=1}^{n} w_{jk}.x_{jk}) + b_j$$
$$H_{init\ 1,1} = ((0{,}813 * 0{,}648) + (-0{,}603 * 0{,}409) + (-0{,}919 * 0{,}065) + (0{,}378 * 0{,}016) + 0{,}424 = 0{,}651$$

Hasil perhitungan manual yang didapatkan ada pada Tabel 4.7 fitur ($x_{1,1}$). Adapun perhitungan *output hidden layer* sebelum masuk ke dalam tahap mengitung fungsi aktivasi akan ditunjukkan pada Tabel 4.7.

Tabel 4.7 Matriks *Output Hidden Layer*

	H_{init}	
1	0,651	1,152
2	0,878	1,335
3	0,509	0,970
4	0,581	1,841
5	0,134	1,952
6	0,422	1,588
7	0,658	0,880
8	0,512	1,229
9	0,566	0,805
10	0,577	0,840
11	0,373	1,461
12	0,553	1,108
13	0,694	1,021
14	0,540	0,839
15	0,779	1,283

Tahap 2 : Setelah dilakukan perhitungan *output hidden layer,* maka akan dilakukan perhitungan fungsi aktivasi menggunakan fungsi aktivasi *sigmoid biner* yang akan diberikan simbol $H(x)$. Nilai x merupakan hasil *output layer* yang ditunjukkan pada Tabel 4.7 fitur ($x_{1,1}$).

Berikut akan ditunjukkan mengenai peritungan fungsi aktivasi *sigmoid biner* berdasarkan Persamaan 2.1.

$$H(x)_{1,1} = \frac{1}{1+e^{-x}} = \frac{1}{1+e^{-0.65135}} = 0{,}657$$

Hasil perhitungan manual berada pada Tabel 4.8 yakni fitur ($x_{1,1}$). Adapun hasil perhitungan *output hidden layer* menggunakan fungsi aktivasi akan ditunjukkan pada Tabel 4.8.

Tabel 4.8 Matriks *Output Hidden Layer* Menggunakan Fungsi Aktivasi

	$H(x)$	
1	0,657	0,759
2	0,706	0,791
3	0,624	0,725
4	0,641	0,863
5	0,533	0,875
6	0,604	0,830
7	0,658	0,706
8	0,625	0,773
9	0,638	0,691
10	0,640	0,698
11	0,592	0,811
12	0,634	0,751
13	0,667	0,735
14	0,631	0,698
15	0,685	0,783

Tahap 3 : Dari hasil perhitungan *output hidden layer* menggunakan fungsi aktivasi maka akan digunakan sebagai proses perhitungan *output weight.* Sebelum masuk kedalam perhitungan *output weight,* akan dilakukan perhitungan Matriks H^+ atau disebut dengan *Moore-Penrose Pseudo Inverse* yang berasal dari perhitungan *hidden layer* bersama fungsi aktivasi. Matriks *hidden layer* yang didapatkan sebelumnya kemudian di *transpose.* Pada Tabel 4.9 akan ditunjukkan matriks *transpose hidden layer*.

Tabel 4.9 Transpose Matriks *Output Hidden Layer* Dengan Fungsi Aktivasi

H^T	1	2	…	15
1	0,657	0,706	….	0,685
2	0,759	0,791	….	0,783

Tahap 4: Dari hasil transpose matriks pada Tabel 4.9 maka langkah selanjutnya adalah mengalikan matriks *output hidden layer* dengan matriks *transpose hidden layer.*

Berikut akan dijabarkan mengenai peritungan perkalian matriks *transpose* dengan matriks *output hidden layer* menggunakan fungsi aktivasi *sigmoid biner.*

$$(H^T H)_{1,1} = (0{,}657 * 0{,}657) + (0{,}759 * 0{,}759) + \cdots + (0.685 * 0{,}685) + (0{,}783 * 0{,}783) = 6{,}093$$

Pada Tabel 4.10 akan ditunjukkan hasil dari perkalian matriks *output hidden layer* dengan fungsi aktivasi bersama matriks *transpose hidden layer.*

Tabel 4.10 Perkalian Matriks *Transpose* Dengan Matriks *Output Hidden Layer* Menggunakan Fungsi Aktivasi *Sigmoid Biner*

$(H^T H)$	1	2
1	6,093	7,299
2	7,299	8,861

Tahap 5 : Langkah selanjutnya adalah dilakukan *inverse* terhadap perkalian matriks *transpose hidden layer* dengan matriks *hidden layer* pada Tabel 4.10. Untuk mencari nilai *inverse* digunakan perhitungan Operasi Baris Elementer (OBE). Hasil dari *inverse* matriks akan ditunjukkan pada Tabel 4.11. Berikut akan dijabarkan mengenai perhitungan *inverse* pada matriks:

1. Tahap pertama membuat matriks identitas atau (*I).* Dalam perhitungan *inverse* matriks *R1* merupakan Baris 1, *R2* merupakan baris 2.

   $$[H \mid I] = \begin{bmatrix} 6{,}093 & 7{,}299 \\ 7{,}299 & 8{,}861 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

2. $R1 : 6{,}093 \rightarrow R1$

   $$[H \mid I] = \begin{bmatrix} 1 & 1{,}197 \\ 7{,}299 & 8{,}861 \end{bmatrix} \begin{bmatrix} 0{,}164 & 0 \\ 0 & 1 \end{bmatrix}$$

3. $R2 - 7{,}299 * R1 \rightarrow R2$

   $$[H \mid I] = \begin{bmatrix} 1 & 1{,}197 \\ 0 & 0{,}117 \end{bmatrix} \begin{bmatrix} 0{,}164 & 0 \\ -1{,}197 & 1 \end{bmatrix}$$

4. $R2 : 0{,}117 \rightarrow R2$

   $$[H \mid I] = \begin{bmatrix} 1 & 1{,}197 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 0{,}164 & 0 \\ -10{,}193 & 8{,}509 \end{bmatrix}$$

5. $R1 - 1{,}197 * R2 \rightarrow R1$

   $$[H \mid I] = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 12{,}375 & -10{,}193 \\ -10{,}193 & 8{,}509 \end{bmatrix}$$

Tabel 4.11 Matriks I*nverse*

$(H^T H)^{-1}$	1	2
1	12,375	-10,193
2	-10,193	8,509

Tahap 6 : Setelah didapatkan matriks *inverse* maka langkah selanjutnya adalah mengalikan matriks *inverse* dengan matriks hasil *transpose* agar didapatkan nilai matriks *Moore-Penrose Pseudo Inverse* atau $(H)^+$**.** Berikut akan dijabarkan tentang perhitungan nilai $(H)^+$berdasarkan Persamaan 2.6.

$$H^+ = (H^T * H)^{-1} * H^T$$
$$H^+_{1,1} = \big((12.375 * 0{,}657) + (-10{,}193 * 0{,}759)\big) = 0{,}388$$

Pada Tabel 4.12 akan ditunjukkan mengenai nilai $(H)^+$ yang telah dilakukan perhitungan sebelumya.

Tabel 4.12 Matriks Nilai *Moore-Penrose Pseudo Invers* (H⁺)

H^+	1	2	….	15
1	0,388	0,672	….	0,502
2	-0,234	-0,464	….	-0,325

Tahap 7: Setelah didapatkan nilai matriks $(H)^+$maka langkah selanjutnya adalah menghitung *output weight* dengan cara mengalikan matriks *moore-penrose pseudo inverse* $(H)^+$ dengan nilai prediksi yang telah dinormalisasi (T). Berikut ini akan dijabarkan mengenai perhitungan nilai *output weight* berdasarkan Persamaan 2.7.

$$\beta = H^+T$$
$$\beta = (0{,}382 * 0{,}413) + (0{,}672 * 0{,}596) + \cdots (0{,}502 * 0{,}442) = 1{,}227$$

Pada Tabel 4.13 akan ditunjukkan mengenai matriks hasil keluaran akhir pada proses *training* yaitu *output weight.*

Tabel 4.13 Nilai *Output Weight*

β
1,227
-0,340

5. Proses *Testing*

Pada proses *testing* ini menggunakan data sebanyak 5 yang diambil dari data karakterisasi tanaman kenaf yang telah dilakukan normalisasi. Pada proses *testing* digunakan fungsi aktivasi *sigmoid biner,* kemudian nilai *input weight* serta nilai bias memiliki nilai yang sama dengan proses *training* yang telah dirandom. Pada proses *testing* jumlah *neuron* yang terdapat dalam *hidden layer* adalah sebanyak 2. Pada Tabel 4.14 akan ditunjukkan nilai hasil normalisasi pada data.

Tabel 4.14 Normalisasi Pada Data *Testing*

Data ke-	X1	X2	X3	X4	T
16	0,148	0	0,021	0,033	0,417
17	0,851	0,518	0,038	0,5	0,845
18	0,203	0,317	0,004	0,016	0,551
19	1	0,334	0,033	0	0,476

Tabel 4.14 Normalisasi Pada Data *Testing* (Lanjutan)

Data Ke-	X1	X2	X3	X4	T
20	0,425	0,145	0,024	0,183	0,579

Adapun tahapan-tahapan yang akan dilakukan dalam proses *testing* adalah:
Tahap 1 : Dalam melakukan perhitungan *output hidden layer* menggunakan fungsi aktivasi *sigmoid biner*, langkah awal yang dilakukan adalah menghitung *output hidden layer,* Selanjutnya hasil dari *output hidden layer* tersebut dihitung menggunakan fungsi aktivasi sigmoid *biner*. Berikut perhitungan *output hidden layer* berdasarkan Persamaan 2.5.

$$H_{init\,ij} = (\sum_{k=1}^{n} w_{jk}.x_{ik}) + b_j$$

$$H_{init\,ij} = \big((0{,}813 * 0{,}148) + (-0{,}603 * 0) + (-0{,}919 * 0{,}021) + (0{,}378 * 0{,}033)\big) + 0{,}424 = 0{,}538$$

Pada Tabel 4.15 akan dijabarkan mengenai hasil *output hidden layer.*

Tabel 4.15 Matriks *Output Hidden Layer*

H_{init}	1	2
1	0,538	0,723
2	0,799	1,751
3	0,629	0,845
4	0,730	1,299
5	0,910	1,014

Tahap 2 : Berikut akan diberikan contoh mengenai perhitungan *output hidden layer* dengan menggunakan fungsi aktivasi *sigmoid biner* berdasarkan Persamaan 2.1.

$$= \frac{1}{1+e^{-x}} = H(x)_{1,1} = \frac{1}{1+e^{-0{,}538}} = 0{,}631$$

Pada Tabel 4.16 akan ditunjukkan mengenai hasil dari *output hidden layer* dengan menggunakan fungsi aktivasi *sigmoid biner.*

Tabel 4.16 Perkalian Matriks *Output Hidden Layer* dengan Menggunakan Fungsi Aktivasi *Sigmoid*

$H(x)$	1	2
1	0,631	0,673
2	0,689	0,852
3	0,652	0,699
4	0,713	0,785
5	0,674	0,733

Tahap 3 : Sebelum masuk ke proses denormalisasi data maka akan dilakukan perhitungan *output layer* yang merupakan hasil dari prediksi atau estimasi. Pada contoh dibawah akan ditunjukkan mengenai perhitungan *output layer* berdasarkan Persamaan 2.8.

$$\hat{y} = H\beta$$
$$y_1 = (0{,}631 * 1{,}227) + (0{,}673 * -0{,}340) = 0{,}565$$

Pada Tabel 4.17 akan ditunjukkan mengenai hasil perhitungan dari *output layer.*

Tabel 4.17 Hasil Perhitungan Nilai *Output Layer*

$\hat{y}$
0,545
0,556
0,562
0,608
0,578

6. Melakukan Proses Denormalisasi Data
   Pada proses denormalisasi data dilakukan berdasarkan hasil keluaran *output layer* yang telah dihitung sebelumnya. Berikut akan dijabarkan mengenai proses perhitungan denormalisasi data berdasarkan Persamaan 2.9.

$$d = d' (\max - \min) + min$$
$$d_1 = 0{,}4179 (3006 - 1319) + 1319 = 2239{,}59$$

Pada Tabel 4.18 akan ditunjukkan mengenai hasil perhitungan denormalisasi data.

Tabel 4.18 Denormalisasi Data

Denormalisasi
2239,59
2257,44
2267,48
2345,11
2294,59

7. Melakukan Evaluasi Terhadap Sistem Menggunakan MAPE
   Tahap akhir yang dilakukan adalah memberikan evaluasi terhadap sistem dengan menggunakan perhitungan nilai MAPE. Dalam melakukan estimasi perlu dilakukan evaluasi untuk menghitung seberapa besar nilai *error* yang dihasilkan. Berikut akan dijabarkan mengenai perhitungan MAPE berdasarkan Persamaan 2.10.

$$MAPE = \frac{1}{n} \Sigma_{i=1}^{n} \left| \frac{\hat{y}_i - y_i}{\hat{y}_i} \times 100 \right|$$

$$MAPE = \frac{1}{5}\left(\left|\frac{2024-2239{,}59}{2024}\right|\right) + \left(\left|\frac{2746-2257{,}44}{2746}\right|\right) + \left(\left|\frac{2250-2267{,}48}{2250}\right|\right) + \left(\left|\frac{2123-2345{,}11}{2123}\right|\right) + \left(\left|\frac{2296-2294{,}59}{2296}\right|\right) * 100 = 9{,}944\%$$

4.4 Perancangan Antarmuka

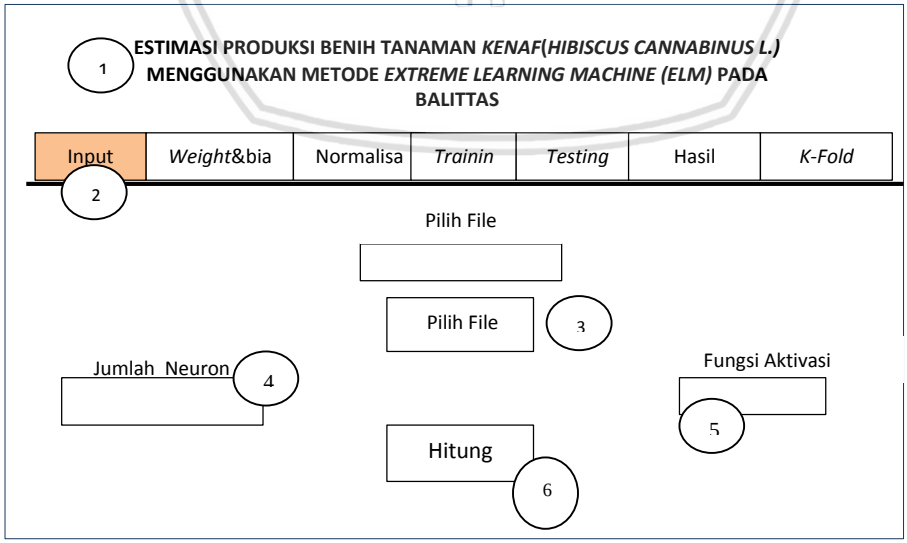
Pada perancangan antarmuka dilakukan agar menggambarkan implementasi sistem yang akan dibuat dalam melakukan estimasi hasil produksi benih tanaman kenaf. Pada perancangan antarmuka ini terdiri dari halaman untuk memasukkan data, inisialisasi *input weight* dan bias, normalisasi, *training, testing,* hasil, dan *k-fold*.

4.4.1 Perancangan Antarmuka Halaman *Input File*

Pada halaman ini akan ditampilkan pada saat pertama kali membuka sistem. Halaman ini merupakan perintah untuk memasukkan data yang akan digunakan. Pada sistem yang akan dibuat, data yang akan dimasukan merupakan data karakterisasi tanaman kenaf yang akan dimasukkan dengan format xls. Perancangan halaman *input file* akan ditunjukkan pada Gambar 4.16.

Dari Gambar 4.16 akan dijelaskan mengenai bagian-bagian dari perancangan halaman *input file*, adalah sebagai berikut.

1. *Header* atau judul dari sistem yang akan dibuat.
2. Menu tab yang diberi warna merupakan halaman yang sedang aktif digunakan.
3. Tombol untuk memilih atau mengunggah file yang akan digunakan pada sistem.
4. *Text Box* yang digunakan untuk memasukkan jumlah *neuron.*
5. *Combo Box* yang akan digunakan untuk memilih fungsi aktivasi yang akan digunakan.
6. Tombol hitung untuk memproses sistem secara keseluruhan.



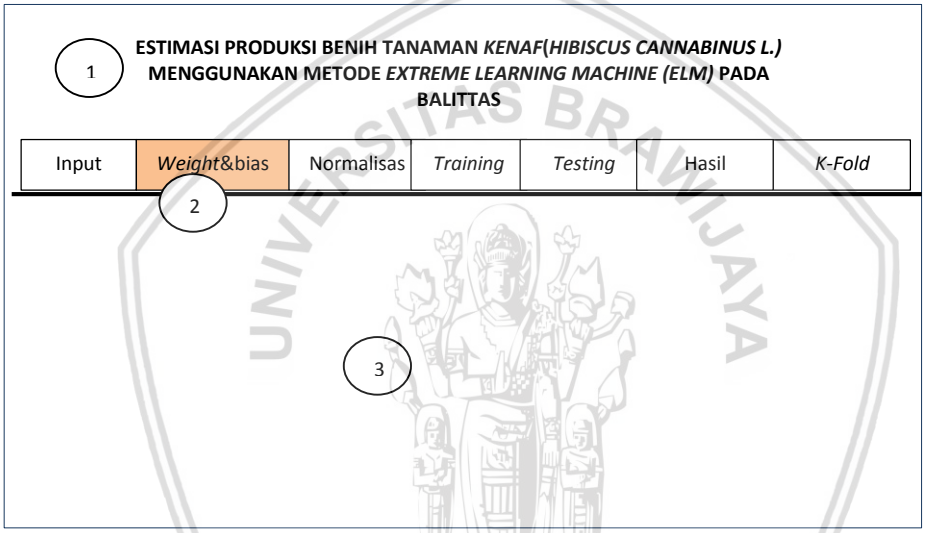
Gambar 4.16 Perancangan Antarmuka Halaman *Input File*

4.4.2 Perancangan Halaman Inisialisasi *Input Weight* Dan Bias

Pada halaman perancangan *weight&*bias merupakan halaman yang akan memperlihatkan jumlah *input weight &* bias yang akan digunakan. Nilai yang didapatkan adalah nilai random dan akan ditampilkan dalam bentuk matriks. Pada Gambar 4.17 akan ditunjukan mengenai halaman *weight&*bias.Perancangan Halaman Inisialisasi *Weight* Dan Bias.

Dari Gambar 4.17 akan dijelaskan mengenai bagian-bagian dari perancangan halaman inisialisasi *weight&*bias, adalah sebagai berikut:

1. *Header* atau judul dari sistem yang akan dibuat.
2. Menu tab yang diberi warna merupakan halaman yang sedang aktif digunakan.
3. *Data View* yang digunakan untuk menampilkan nilai dari *input weight* dan bias.



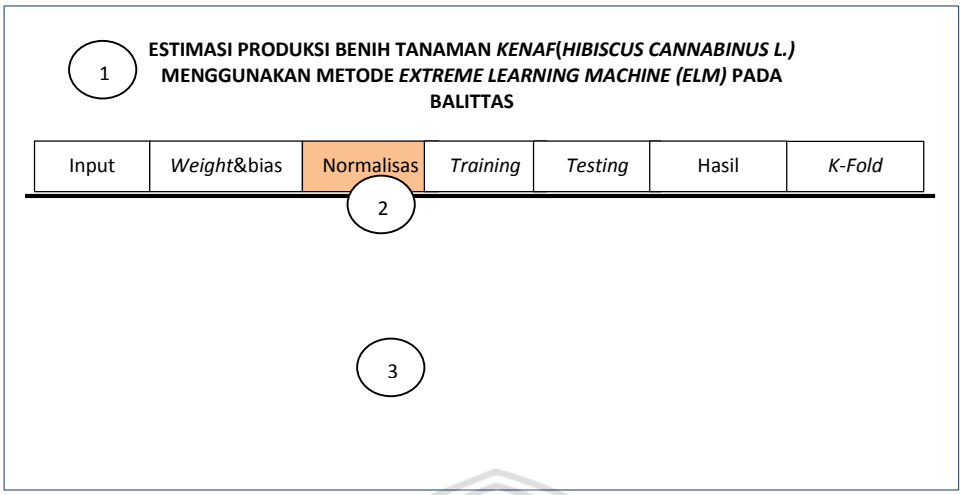
Gambar 4.17 Perancangan Antarmuka Halaman Inisialisasi *Input Weight* dan Bias

4.4.3 Perancangan Antarmuka Halaman Normalisasi

Dalam perancangan halaman normalisasi akan ditampilkan mengenai proses normalisasi pada data. Pada Gambar 4.18 akan ditunjukkan gambar perancangan antarmuka halaman normalisasi.

Dari Gambar 4.18 akan dijelaskan mengenai bagian-bagian dari perancangan halaman inisialisasi *input weight&*bias, adalah sebagai berikut:

1. *Header* atau judul dari sistem yang akan dibuat.
2. Menu tab yang diberi warna merupakan halaman yang sedang aktif digunakan.
3. *Data View* yang digunakan untuk menampilkan nilai normalisasi data.



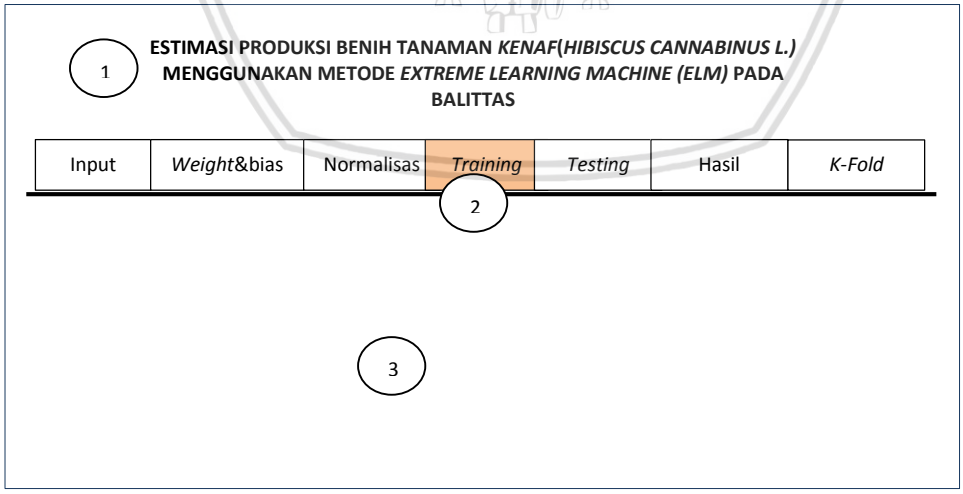
Gambar 4.18 Perancangan Antarmuka Halaman Normalisasi

4.4.4 Perancangan Antarmuka Halaman *Training*

Pada halaman perancangan antarmuka proses *training* ini berisi mengenai perhitungan *training*. Dari hasil perhitungan tersebut maka terdapat nilai *output weight* yang dihasilkan. Pada Gambar 4.19 akan ditunjukkan mengenai perancangan antarmuka halaman *training*.

Dari Gambar 4.19 akan dijelaskan mengenai bagian-bagian dari perancangan antarmuka halaman *training* adalah sebagai berikut:

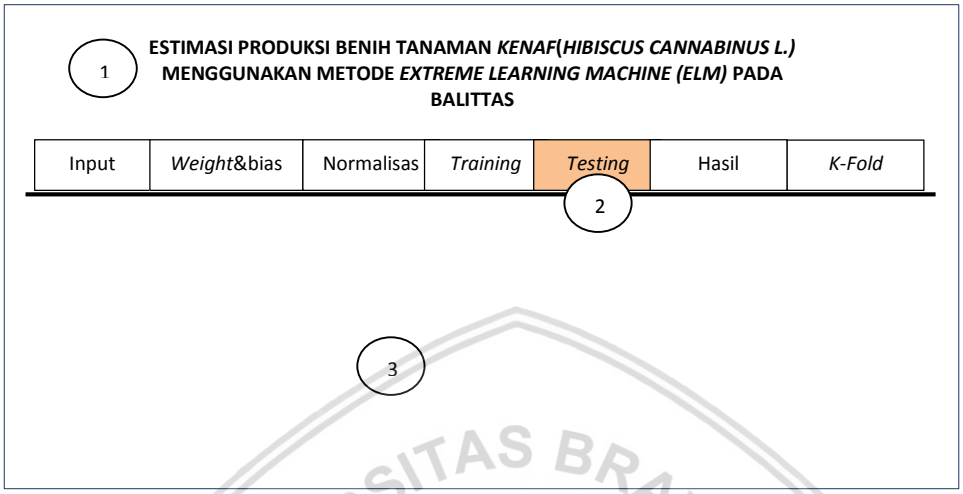
1. *Header* atau judul dari sistem yang akan dibuat.
2. Menu tab yang diberi warna merupakan halaman yang sedang aktif digunakan.
3. *Data View* yang digunakan untuk menampilkan proses *training.*



Gambar 4.19 Perancangan Antarmuka Halaman *Training*

4.4.5 Perancangan Antarmuka Halaman *Testing*

Pada perancangan halaman *testing* akan digunakan untuk menampilkan proses perhitungan *testing* sehingga menghasilkan hasil estimasi. Pada Gambar 4.20 akan ditunjukkan mengenai halaman *testing.*



Gambar 4.20 Perancangan Antarmuka Halaman *Testing*

Dari Gambar 4.20 akan dijelaskan mengenai bagian-bagian dari perancangan halaman *testing* adalah sebagai berikut:

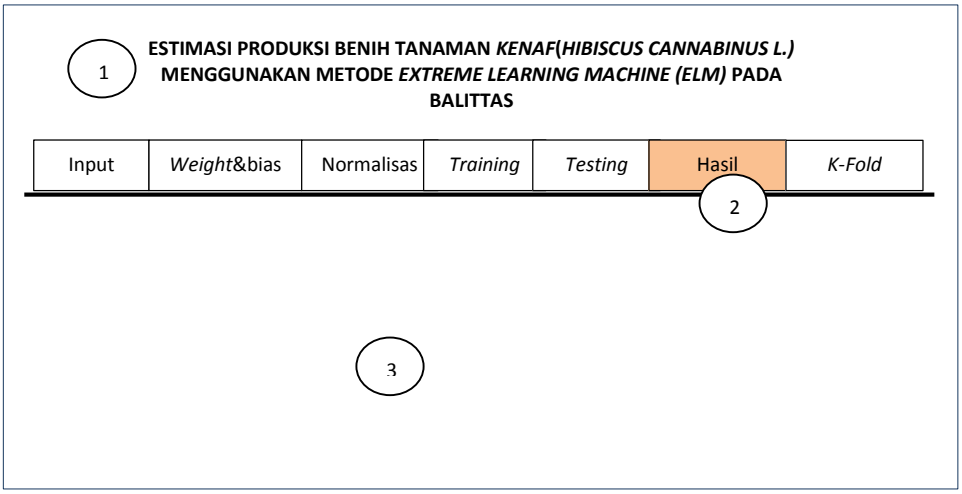
1. *Header* atau judul dari sistem yang akan dibuat.
2. Menu tab yang diberi warna merupakan halaman yang sedang aktif digunakan.
3. *Data View* yang digunakan untuk menampilkan proses *testing.*

4.4.6 Perancangan Antarmuka Halaman Hasil

Perancangan halaman hasil akan ditunjukkan mengenai hasil dari estimasi berupa nilai denormalisasi serta nilai aktual sebelum dilakukan normalisasi. Pada Gambar 4.21 akan ditunjukkan mengenai perancangan antarmuka halaman hasil

Dari Gambar 4.21 akan dijelaskan mengenai bagian-bagian dari perancangan halaman hasil adalah sebagai berikut:

1. *Header* atau judul dari sistem yang akan dibuat.
2. Menu tab yang diberi warna merupakan halaman yang sedang aktif digunakan.
3. *Data View* yang digunakkan untuk menampilkan hasil berupa evaluasi sistem menggunakan MAPE dan nilai estimasi hasil produksi benih tanaman kenaf.



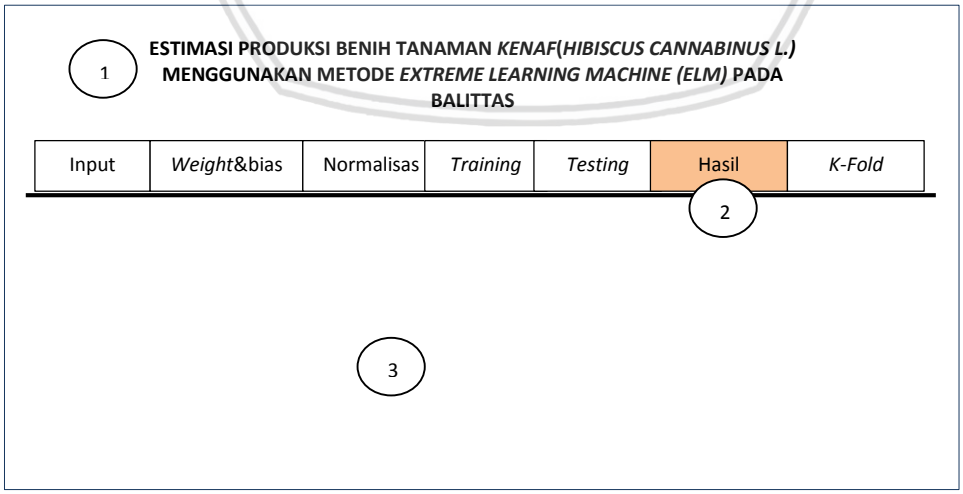
Gambar 4.21 Perancangan Antarmuka Halaman Hasil

4.4.7 Perancangan Antarmuka Halaman *K-Fold Cross Validation*

Pada perancangan halaman *k-fold cross validation* akan dilakukkan pengujian validasi terhadap data serta parameter yang digunakan. Dalam halaman ini akan menunjukkan rata-rata nilai MAPE yang dihasilkan pada k tertentu. Pada Gambar 4.22 akan ditunjukkan mengenai halaman *k-fold cross validation*.

Data View Dari Gambar 4.22 akan dijelaskan mengenai bagian-bagian dari perancangan halaman *k-fold cross validation* adalah sebagai berikut:

1. *Header* atau judul dari sistem yang akan dibuat.
2. Menu tab yang diberi warna merupakan halaman yang sedang aktif digunakan.
3. Data View yang digunakan untuk menampilkan proses pengujian *k-fold cross validation.*



Gambar 4.22 Perancangan Antarmuka Halaman *K-Fold Cross Validation*

4.5 Perancangan Uji Coba Dan Evaluasi Sistem

Pada perancangan proses uji coba ini digunakan agar mengetahui kinerja sistem menggunakan metode ELM. Selain itu dilakukan evaluasi terkait dengan sistem yang dibuat pada penelitian ini. Pada tahap evaluasi sistem menggunakan MAPE. Adapun pengujian yang akan dilakukan pada penelitian kali ini adalah:

1. Pengujian jumlah *neuron* pada *hidden layer.*
2. Pengujian fungsi aktivasi.
3. Pengujian persentse perbandingan jumlah data *training* terhadap *testing.*
4. Pengujian *k-fold cross validation*

4.5.1 Pengujiian Jumlah *Neuron* Pada *Hidden Layer*

Pada pengujian ini, jumlah *neuron* pada *hidden layer* yang akan dihasilkan berjumlah 1 sampai dengan 8 *neuron.* Perbandingan data yang digunakan berdasarkan hasil terbaik yang dilakukan sebelumnya. Fungsi aktivasi yang akan digunakan adalah fungsi aktivasi *sigmoid biner.* Pengujian ini memiliki tujuan untuk mengetahui jumlah *neuron* terbaik dalam melakukan estimasi produksi benih tanaman kenaf. Pada Tabel 4.19 akan ditunjukkan mengenai pengujian jumlah *neuron* pada *hidden layer.*

Tabel 4.19 Pengujian *Input Neuron* Pada *Hidden Layer*

Jumlah *Neuron*	Nilai MAPE (%) Percobaan ke-i										Rata-rata Mape (%)
	1	2	3	4	5	6	7	8	9	10	
1											
2											
3											
4											
5											
6											
7											
8											

4.5.2 Pengujian Fungsi Aktivasi

Pada pengujian fungsi aktivasi ini memiliki manfaat agar mengetahui fungsi aktivasi yang baik untuk sistem sehingga menghasilkan nilai MAPE terkecil. Pada Tabel 4.20 akan ditunjukkan mengenai pengujian fugsi aktivasi.

Tabel 4.20 Pengujian Nilai Fungsi Aktivasi

Fungsi Aktivasi	Nilai MAPE (%) Percobaan ke-i										Rata-rata Mape (%)
	1	2	3	4	5	6	7	8	9	10	
Sigmoid Biner											
Sigmoid Bipolar											
Sin											

4.5.1 Pengujian Persentase Perbandingan Data *Training* Terhadap Data *Testing*

Pengujian ini dilakukan untuk mengetahui jumlah data *training* yang baik untuk proses pengujian dengan jumlah data *testing* sebanyak 10%. Pada pengujian ini jumlah data *training* bervariasi yaitu 90%, 80%, 70%,60%,50%. *Input* yang digunakan merupakan data karakterisasi tanaman kenaf berjumlah 100 data. Jumlah *neuron* yang digunakan adalah sebanyak 2. Fungsi aktivasi yang akan digunakan adalah fungsi aktivasi sigmoid biner. Perbandingan ini dilakukan agar mengetahui persentase data *training* yang memiliki nilai MAPE yang baik. Nilai MAPE yang dihasilkan berbeda-beda karena nilai *input weight* serta bias yang dihasilkan bernilai *random*. Pengujian ini dilakukan sebanyak 10 kali percobaan. Pada Tabel 4.21 akan ditunjukkan mengenai pengujian perbandingan data *training* terhadap data *testing* sebanyak 10%.

Tabel 4.21 Pengujian Peresentase Perbandingan Jumlah Data *Training* dan *Testing*

Persentase Perbandingan Jumlah Data *Training* dan *Testing*	Nilai MAPE (%) Percobaan ke-i										Rata-rata Mape (%)
	1	2	3	4	5	6	7	8	9	10	
90% : 10%											
80% : 10%											
70% : 10%											
60% : 10%											
50% : 10%											

4.5.2 Pengujian *K-Fold Cross Validation*

Pengujian ini dilakukan untuk memvalidasi sistem yang telah dibuat. Pada pengujian ini seluruh data yang dimiliki dapat digunakan sebagai data *training* diuji secara bergantian dan data satu lainya digunakan untuk proses *testing* agar hasilnya dapat optimal. Dengan menggunakan nilai *k* dapat menentukan

banyaknya percobaan data *set.* Pada Tabel 4.22 akan ditunjukkan mengenai pengujian *k-fold cross validation.*

Tabel 4.22 Pengujian *K-Fold Cross Validation*

Percobaan ke-	Subset ke-					Rata-rata (%)
	1	2	3	4	5	
1						
2						
3						
4						
5						
6						
7						
8						
9						
10						

BAB 5 IMPLEMENTASI

Dalam bab ini menjelaskan tentang implementasi serta antarmuka hasil perancangan sistem yang telah dibuat sebelumnya pada Bab 4 yang berfungsi untuk melakukan estimasi terhadap hasil produksi benih tanaman kenaf.

5.1 Implementasi Sistem

Dengan adanya hasil perancangan yang telah dibuat sebelumnya, maka akan dijelaskan mengenai implementasi sistem yang akan digunakan untuk mengestimasi produksi benih. Dalam mengimplementasikan sistem, digunakan bahasa pemrograman *java* menggunakan aplikasi *netbeans* 8.0.1

5.1.1 Implementasi Proses Normalisasi Data

Normalisasi data dilakukan agar data yang digunakkan dapat dipetakkan dengan baik sehingga memiliki *range* yang seimbang. Normalisasi data menggunakan rumus *min-max normalization* berdasarkan Persamaan 2.4. Adapun implementasi sistem normalisasi data akan ditunjukkan pada Kode Program 5.1.

```
1.  public double[][] Normalisasi (double [][] Data){
2.    double [] maxs = getMax ();
3.    double [] min = getMin ();
4.    double data_normal[][]= new double
5.    [Data.length][Data[0].length];
6.       for (int i = 0; i <  Data.length; i++) {
7.         for (int j = 0; j < Data[0].length; j++) {
8.           data_normal[i][j] = (Data[i][j] - min[j])
9.           / (maxs[j]-min[j]);
10.
11.              }
12.          }
13.         return data_normal;
```

Kode Program 5.1 Proses Normalisasi Data

Adapun penjelasan dari Kode Program 5.1 yaitu sebagai berikut:

1. Baris 1 adalah proses inisialisasi terhadap *method* Normalisasi.
2. Baris 2-3 adalah proses pengambilan terhadap nilai maksimal dan minimal yang berasal dari parameter yang di*input*kan.
3. Baris 4-5 adalah proses inisialisasi terhadap variabel data.
4. Baris 6-12 adalah proses perhitungan normalisasi menggunakan *min-max normalization* berdasarkan Persamaan 2.4.
5. Baris 13 adalah proses pengembalian nilai variabel data_normal.

5.1.2 Implementasi Inisialisasi *Input Weight* serta Bias

Untuk mendapatkan nilai *input weight* serta bias, maka dilakukan proses *random* nilai dengan *range -1* sampai dengan 1. Proses untuk menginisialisasi nilai *input weight* akan ditunjukkan pada Kode Program 5.2

Adapun penjelasan dari Kode Program 5.2 adalah sebagai berikut.

1. Baris 1-3 adalah proses inisialisasi terhadap *method* Bobot_Awal.
2. Baris 4-5 adalah proses inisialisasi terhadap variabel bobot
3. Baris 6-16 adalah proses untuk mendapatkan nilai *input weight* secara *random* dengan *range* -1 sampai dengan 1.
4. Baris 17-18 adalah proses untuk pengembalian nilai bobot.
5. Baris 19-20 adalah proses inisialisasi terhadap *method* Bias_Awal.
6. Baris 21-22 adalah proses untuk inisialisasi variabel bias.
7. Baris 23-29 adalah proses untuk mencari nilai bias secara *random* dengan *range* antara -1 sampai dengan 1.
8. Baris 30-32 adalah proses untuk mengembalikan nilai bias.

```
1.    public double [][]Bobot_awal (int
2.    jumlah_hidden, int jumlah_fitur,double
3.    min, double max){
4.          double bobot [][] = new double
5.          [jumlah_hidden][jumlah_fitur];
6.           for (int i = 0; i < jumlah_hidden;
7.           i++) {
8.             for (int j = 0; j < jumlah_fitur;
9.             j++) {
10.              double temp_bobot =
11.              (Math.random() *(max -
12.              min))+min;
13.                bobot [i][j] = temp_bobot;
14.
15.            }
16.
17.          }
18.          return bobot;
19.       }
20.       public double [] Bias_awal (int
21.       jumlah_hidden, double min, double max){
22.          double bias [] = new
23.          double[jumlah_hidden];
24.             for (int i = 0; i < jumlah_hidden;
25.             i++) {
26.                double temp_bias =
27.                (Math.random() * (max-min) +
28.                min);
29.              bias[i] = temp_bias;
30.          }
31.          return bias;
32.    }
```

Kode Program 5.2 Inisialisasi *Input Weight* serta Bias

5.1.3 Implementasi *Output Hidden Layer*

Pada proses perhitungan nilai *output hidden layer* dilakukan pada proses *training* dan *testing.*Fungsi *output layer* adalah untuk mendapatkan nilai akhir

berupa estimasi. Pada Kode Program 5.3 akan ditunjukkan mengenai implementasi *output hidden layer.*

```
1.    public double [][] h_init(double[][] bobot, double [][]
2.    input, double[]bias, int aktivasi){
3.           System.out.println(Arrays.toString(bias));
4.           double[][] wt = transpose_matrix(bobot);
5.           System.out.println("bobot transpose");
6.              for (int i = 0; i < wt.length; i++) {
7.                System.out.println(Arrays.toString(wt[i]));
8.           }
9.                System.out.println("input kali wt");
10.               double[][] input_kali_wt =
11                kali_matrix(input,wt);
12.                  for (int i = 0; i <
13.                  input_kali_wt.length; i++) {
14.                     System.out.println
15.                    (Arrays.toString(input_kali_wt[i]));
16.
17.          }
18.              double [][] hinit_sbl_aktif = new
19.              double[input_kali_wt.length]
20.              [input_kali_wt[0].length];
21.              System.out.println
22.              ("hinit sebelum menggunakan fungsi
23.              aktivasi");
24.                for (int i = 0; i <input_kali_wt
25.                .length; i++) {
26.                   for (int j = 0; j <
27.                   input_kali_wt[0].length; j++) {
28.                    hinit_sbl_aktif[i][j] =
29.                     input_kali_wt[i][j] + bias [j];
30.               }
31.               System.out.println(Arrays.
32.               toString(hinit_sbl_aktif[i]));
33.          }
```

Kode Program 5.3 *Output Hidden Layer*

Adapun penjelasan dari Kode Program 5.3 adalah sebagai berikut.

1. Baris 1-2 adalah proses deklarasi terhadap *method* Bobot_Awal.
2. Baris 3 adalah proses untuk menampilkan *array* menjadi variabel bias bertipe data string.
3. Baris 4 adalah proses untuk memanggil *method transpose_matrix* dan dimasukkan ke dalamvariabel wt.
4. Baris 5 adalah proses untuk mencetak keluaran berupa tulisan bobot transpose.
5. Baris 6-9 proses perhitungan mencari nilai *transpose* dari *input weight.*
6. Baris 10 adalah proses untuk mencetak keluaran berupa tulisan *input* kali wt.
7. Baris 11-12 adalah proses untuk memanggil *method* kali_matriks

8. Baris 13-19 adalah proses untuk melakukan perhitungan perkalian nilai *input* dengan matriks *input weight* yang telah di *transpose.*
9. Baris 20-22 adalah proses untuk melakukan inisialisasi terhadap hinit_sbl_aktif.
10. Baris 23-24 adalah proses untuk mencetak tulisan berupahinit sebelum menggunakan fungsi aktivasi.
11. Baris 25-33 adalah proses perhitungan nilai *output hidden layer* sebelum menggunakan fungsi aktivasi.

5.1.4 Implementasi *Output Hidden Layer* Menggunakan Fungsi Aktivasi

Pada sistem estimasi produksi benih, perhitungan *output hidden layer* dengan fungsi aktivasi yang telah dilakukan perhitungan pada bab 4. Langkah untuk menghitung fungsi aktivasi ini terdapat pada proses *training* dan *testing.* Pada Kode Program 5.4 akan ditunjukkan mengenai implementasi *output hidden layer* menggunakan fungsi aktivasi.

```
1.  double [][] hinit = null;
2.    switch (aktivasi){
3.      case 0:
4.        hinit = aktivasi_biner(hinit_sbl_aktif);
5.      break;
6.      case 1 :
7.        hinit = aktivasi_linear(hinit_sbl_aktif);
8.      break;
9.      case 2:
10.       hinit = aktivasi_sin(hinit_sbl_aktif);
11.     break;
12.     case 3:
13.       hinit = aktivasi_bipolar(hinit_sbl_aktif);
14.     break;
15.         }
16.           System.out.println("Hinit");
17.           for (int i = 0; i < hinit.length; i++) {
18.              System.out.println
19.              (Arrays.toString(hinit[i]));
20.           }
21.            return hinit;
22.       }
23.        double [][] aktivasi_biner(double[][]Hinit){
24.           double[][] hasilOutputHidden = new
25.           double[Hinit.length][Hinit[0].length];
26.           for (int i = 0; i < Hinit.length; i++) {
27.               for (int j = 0; j < Hinit[0].length; j++) {
28.                   hasilOutputHidden[i][j]= 1/(1+Math.exp(-
29.                   Hinit[i][j]));
30.
31.               }
32.
33.           }
34.           return hasilOutputHidden;
35.       }
36.           double[][] aktivasi_sin(double[][] Hinit){
37.           double[][] hasilOutputHidden = new
```

```
38.             double[Hinit.length][Hinit[0].length];
39.             for (int i = 0; i < Hinit.length; i++) {
40.                 for (int j = 0; j < Hinit[0].length; j++) {
41.                     hasilOutputHidden[i][j] =
42.                     Math.sin(Hinit[i][j]);
43.
44.                 }
45.             }
46.             return hasilOutputHidden;
47.         }
48.         double [][] aktivasi_bipolar(double [][] Hinit){
49.             double [][] hasilOutputHidden = new double
50.             [Hinit.length][Hinit[0].length];
51.             for (int i = 0; i < Hinit.length; i++) {
52.                 for (int j = 0; j < Hinit[0].length; j++) {
53.                     hasilOutputHidden[i][j] =
54.                     (1 - Math.exp(-Hinit[i][j]))/ (1 +
55.                      Math.exp(-Hinit[i][j]));
56.
57.                 }
58.
59.             }
60.
61.     return hasilOutputHidden;
62.     }
```

Kode Program 5.4 *Output Hidden Layer* Dengan Fungsi Aktivasi

Adapun penjelasan dari Kode Program 5.4 adalah sebagai berikut.

1. Baris 1 adalah proses deklarasi variabel hinit.
2. Baris 2-15 adalah proses seleksi kondisi untuk masing-masing fungsi aktivasi.
3. Baris 16 adalah proses mencetak tulisan berupa Hinit.
4. Baris 17-20 adalah proses menghitung h init menggunakan fungsi aktivasi.
5. Baris 21-22 adalah proses mengembalikan nilai variabel hinit.
6. Baris 23 adalah proses inisialisasi method aktivasi_biner
7. Baris 24-35 adalah proses untuk menghitung *output hidden layer* menggunakan fungsi aktivasi sigmoid biner.
8. Baris 36 adalah proses inisialisasi *method* aktivasi_sin.
9. Baris 37-46 adalah proses untuk menghitung *output hidden layer* menggunakan fungsi aktivasi sin.
10. Baris 48 adalah proses inisialisasi *method* aktivasi_bipolar.
11. Baris 49-62 adalah proses untuk menghitung *output hidden layer* menggunakan fungsi aktivasi *sigmoid bipolar*.

5.1.5 Implementasi Matriks *Moore-Penrose Pseudo Inverse*

Pada perhitungan matriks *Moore-Penrose Pseude Inverse* dilakukan dalam tahap *training.* Pada Kode Program 5.5 akan ditunjukkan mengenai implementasi matriks *moore-penrose pseudo inverse.*

```
1.     double [][] get_moore_penrose(double hinit [][]){
2.             double Htranspose[][] = transpose_matrix(hinit);
3.             double temp_HtH [][]  = kali_matrix(Htranspose,
4.             hinit);
5.             double temp_Hinverse[][] =
6.             inverse_obe(temp_HtH);
7.             System.out.println("inverse");
8.             for (int i = 0; i < temp_Hinverse.length; i++) {
9.                System.out.println(Arrays.
10.               toString(temp_Hinverse[i]));
11.            }
12.
13.            double moore_penrose[][] =
14.            kali_matrix(temp_Hinverse, Htranspose);
15.            System.out.println("Moore penrose pseudo
16.            inverse");
17.            for (int i = 0; i < moore_penrose.length; i++) {
18.               System.out.println(Arrays.
19.               toString(moore_penrose[i]));
20.            }
21.            return moore_penrose;
22.    }
```

Kode Program 5.5 Matriks *Moore Penrose Pseudo Inverse*

Adapun penjelasan dari Kode Program 5.5 adalah sebagai berikut.

1. Baris 1 adalah inisialisasi *method* get_moore_penrose.
2. Baris 2 adalah proses memanggil method transpose_matrix yang disimpan dalam variabel Htranspose
3. Baris 3-4 adalah proses memanggil method kali_matrix yang disimpan dalam variabel temp_HtH.
4. Baris 5-6 adalah proses memanggil method inverse_obe yang disimpan pada variabel temp_Hinverse.
5. Baris 7 adalah proses mencetak tulisan inverse.
6. Baris 8-12 adalah proses untuk menghitung matriks *moore-penrose pseudo inverse.*
7. Baris 13-14 memanggil *method* kali_matriks dan disimpan pada *method moore_penrose*
8. Baris 15-16 adalah proses mencetak tulisan berupa Moore Penrose Pseudo Inverse.
9. Baris 17 adalah melakukan perulangan matriks *moore-penrose pseudo inverse.*

10. Baris 18-19 adalah proses untuk menampilkan variabel moore_penrose dalam bentuk string.
11. Brs 20-22 adalah proses untuk mengembalikan nilai variabel moore_penrose.

5.1.6 Implementasi *Inverse* Matriks OBE

Pada proses *inverse* matriks dalam *moore-penrose pseudo inverse* digunakan *inverse* matriks OBE. Proses tersebut guuna menghitung matriks *moore-penrose pseudo inverse.* Pada Kode Program 5.6 akan ditunjukkan mengenai implementasi *inverse* matriks OBE.

```
1.    public double[][] inverse_obe(double[][] data) {
2.       int baris = data.length, kolom = data[0].length;
3.       double[][] hasil_inverse = new double[baris][kolom];
4.       double[][] temp = new double[baris][kolom * 2];
5.       double[][] temp1 = new double[baris][kolom * 2];
6.            //identitas, dan tambah gabung
7.         for (int i = 0; i < baris; i++) {
8.           for (int j = 0; j < kolom * 2; j++) {
9.            if (j - i == baris) {
10.               temp[i][j] = 1;//nambah identitas
11.                  }
12.                  if (j < kolom) {
13.                     temp[i][j] = data[i][j];
14.                  }
15.               }
16.           }
17.           //mengubah dibawah diagonal jadi 0
18.           for (int i = 0; i < baris; i++) {
19.             for (int j = 0; j < baris; j++) {
20.               for (int k = 0; k < kolom * 2; k++) {
21.                  if (i == j) {
22.                     temp1[i][k] = temp[i][k] / temp[i][i];
23.                       } else {
24.                            temp1[j][k] = temp[j][k];
25.                       }
26.                  }
27.               }
28.               temp = pindah(temp1);
29.               for (int j = i + 1; j < baris; j++) {
30.                  for (int k = 0; k < kolom * 2; k++) {
31.                    temp1[j][k] = temp[j][k] - temp[i][k] *
32.                    temp[j][i];
33.                  }
34.               }
35.               temp = pindah(temp1);
36.           }
37.           //mengubah diatas diagonal jadi 0
38.           for (int i = kolom - 1; i > 0; i--) {
39.             for (int j = i - 1; j >= 0; j--) {
40.               for (int k = 0; k < kolom * 2; k++) {
41.                 temp1[j][k] = temp[j][k] - temp[i][k] *
42.                 temp[j][i];
43.                   }
44.               }
45.               temp = pindah(temp1);
```

```
46.             }
47.             //ambil matrix sesuai ukuran awal
48.             for (int i = 0; i < baris; i++) {
49.               for (int j = kolom; j < kolom * 2; j++) {
50.                 hasil_inverse[i][j - kolom] = temp1[i][j];
51.                 }
52.             }
53.             return hasil_inverse;
54.         }
55.         public double[][] pindah(double [][] data){
56.            double[][] hasil = new
57.            double[data.length][data[0].length];
58.            for (int i = 0; i < data.length; i++) {
59.               System.arraycopy(data[i], 0, hasil[i], 0,
60.               data[0].length);
61.            }
62.            return hasil;
63.    }
```

Kode Program 5.6 *Inverse* Matriks OBE

Adapun penjelasan dari Kode Program 5.6 adalah sebagai berikut.

1. Baris 1 adalah inisialisasi *method* inverse_obe.
2. Baris 2 adalah inisialisasi variabel baris dan kolom.
3. Baris 3 adalah inisialisasi variabel hasil_inverse.
4. Baris 4-5 adalah untuk menyimpan variabel baris dan kolom.
5. Baris 7-16 adalah proses untuk menghasilkan matriks identitas serta menghitung dan mengalikan matriks asal dengan matriks identitas.
6. Baris 18-36 adalah proses untuk mengubah nilai di bawah diagonal menjadi 0.
7. Baris 38-46 adalah proses untuk mengubah nilai di atas diagonal menjadi 0.
8. Baris 48-54 adalah proses untuk mengambil nilai matriks sesuai dengan ukuran awal.
9. Baris 55 adalah inisialisasi *method* pindah.
10. Baris 56-57 adalah inisialisasi variabel hasil.
11. Baris 58 adalah proses perulangan terhadap length dari variabel data.
12. Baris 59-61 adalah proses untuk meng*copy* variabel data ke dalam variabel hasil.
13. Baris 62-63 adalah proses untuk mengembalikan nilai variabel hasil.

5.1.7 Implementasi *Output Weight*

Pada proses perhitungan nilai *output weight* dilakukan perkalian antara matriks *moore-penrose pserudo inverse* dengan matriks target berdasarkan hasil perhitungan normalisasi. Pada Kode Program 5.7 akan ditunjukkan mengenai implementasi *output weight.*

```
1.      double []output_weight(double [][] moore_penrose,
2.      double [] target){
3.         double [] hasil_Beta = new
4.         double[moore_penrose.length];
5.              for (int i = 0; i < moore_penrose.length; i++) {
6.                for (int k = 0; k < target.length; k++) {
7.                  hasil_Beta[i]+= moore_penrose[i][k] *
8.                  target[k];
9.                  }
10.             }
11.             System.out.println("Output Weight");
12.             System.out.println(Arrays.toString(hasil_Beta));
13.             return hasil_Beta;
14.     }
```

Kode Program 5.7 *Output Weight*

Adapun penjelasan dari Kode Program 5.7 adalah sebagai berikut.

1. Baris 1-2 adalah proses inisialisasi method *Output_weight*.
2. Baris 3-4 adalah poses inisialisasi variabel hasil_Beta.
3. Baris 5-10 adalah proses untuk menghitung nilai *output weight.*
4. Baris 11 adalah proses untuk mencetak tulisan berupa *Output Weight*.
5. Baris 12 adalah untuk membaca variabel hasil_Beta dalambentuk string.
6. Baris 13-14 adalah untuk mengembalikan nilai variabel hasil_Beta.

5.1.8 Implementasi Proses Perhitungan Nilai Estimasi

Proses perhitungan nilai estimasi merupakan proses yang dilakukan setelah mendapatkan nilai *output weight.* Perhitungan tersebut dilakukan dengan cara mengalikan nilai *output hidden layer* dengan *output weight.* Pada Kode Program 5.8 akan ditunjukkan mengenai implementasi *output* perhitungan nilai peramalan.

```
1.      double [] y(double [][] H,double[] beta){
2.         double[] hasil_y = new double[H.length];
3.            for (int i = 0; i < H.length; i++) {
4.               for (int k = 0; k < beta.length; k++) {
5                  hasil_y[i] += H[i][k] * beta[k];
6.                 }
7.             }
8.             return hasil_y;
9.
10.     }
```

Kode Program 5.8 Perhitungan Nilai Estimasi

Adapun penjelasan dari Kode Program 5.8 adalah sebagai berikut.

1. Baris 1 adalah proses inisialisasi terhadap *method* y.
2. Baris 2 adalah proses inisialisasi terhadap variabel hasil_y.
3. Baris 3-10 adalah proses perhitungan nilai estimasi.

5.1.9 Implementasi Denormalisasi Data

Setelah dilakukan perhitungan estimasi, maka tahap selanjutnya adalah melakukan denormalisasi data. Pada Kode Program 5.9 akan ditunjukkan mengenai implementasi denormalisasi data.

```
1.    public double []denormalisasi(double data []){
2.       double temp_min[] = getMin();
3.       double temp_max[] = getMax();
4.       double min = temp_min[temp_min.length - 1];
5.       double max = temp_max[temp_max.length - 1];
6.       double []data_asli = new double [data.length];
7.       System.out.println("min" +min);
8.       System.out.println("max" +max);
9.       for (int i = 0; i < data.length; i++) {
10.         data_asli[i] =Math.round(data[i] * (max - min) +
11.         min);
12.          }
13.          return data_asli;
14.   }
```

Kode Program 5.9 Denormalisasi Data

Adapun penjelasan dari Kode Program 5.9 adalah sebagai berikut.

1. Baris 1 adalah proses inisialisasi terhadap *method* denormalisasi.
2. Baris 2-3 adalah proses pengambilan nilai variabel min dan max yang akan disimpan pada variabel temp.
3. Baris 4-5 adalah inisialisasi variabel min dan max yang disimpan pada variabel temp.
4. Baris 6 adalah proses untuk inisialisasi variabel data_asli.
5. Baris 7-8 adalah proses untuk mencetak nilai max dan min.
6. Baris 9-12 adalah proses perhitungan denormalisasi data.
7. Baris 13-14 adalah proses untuk mengembalikan nilai variabel data_asli.

5.1.10 Implementasi Evaluasi Sistem Menggunakan MAPE

Pada proses estimasi produsi benih langkah yang dilakukan terakhir kali adalah melakukan evaluasi terhadap sistem yaitu menggunakan MAPE. Pada Kode Program 5.10 akan ditunjukkan mengenai implementasi denormalisasi data.

Adapun penjelasan dari Kode Program 5.10 adalah sebagai berikut.

1. Baris 1 adalah proses inisialisasi terhadap *method* hitung_MAPE.
2. Baris 2 adalah deklarasi variabel tot dengan tipe data *double.*
3. Baris 3-6 adalah proses perhitungan nilai MAPE.
4. Baris 7 adalah proses untuk mencari nilai rata-rata untuk menghasilkan nilai MAPE keseluruhan.
5. Baris 8 adalah proses untuk mengembalikan nilai variabel mape.

```
1.    double hitung_MAPE(double [] t, double[] y){
2.       double tot = 0;
3.         for (int i = 0; i < t.length; i++) {
4.           double temp = Math.abs(t[i] - y[i]) / t[i] * 100;
5.              tot += Math.abs(temp);
6.           }
7.           double mape = tot / t.length;
8.           return mape;
9.         }
```

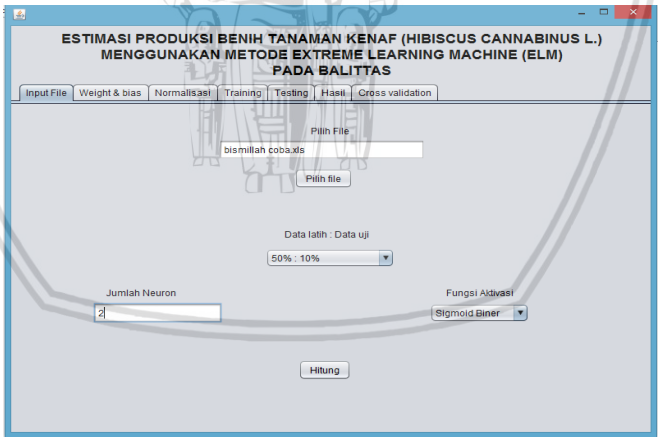
Kode Program 5.10 Evaluasi Sistem Menggunakan MAPE

5.2 Implementasi Antarmuka

Pada bab 4 telah dilakukan perancangan antarmuka, sehingga pada bab ini akan ditunjukkan mengenai hasildari rancangan tersebut. Berdasarkan perancangan antarmuka sebelumnya, sistem ini memiliki 7 buah tab. Diantara tab tersebut adalah *Input* File, *Weight*&Bias, Normalisasi, Training, *Testing*, Hasil dan K-Fold.

5.2.1 Implementasi Antarmuka Halaman *Input File*

Pada antarmuka halaman *Input File* berfungsi untuk meng*inputkan file* yang akan digunakan pada sistem estimasi hasil produksi benih tanaman kenaf. File yang digunakan merupakan data karakterisasi tanaman kenaf. Pada halaman tersebut akan di*input*kan jumlah *neuron* yang akan digunakan, fungsi aktivasi serta perbandingan data latih dengan data uji. Pada Gambar 5.1 akan ditunjukkan mengenai implementasi antarmuka halaman *input file.*

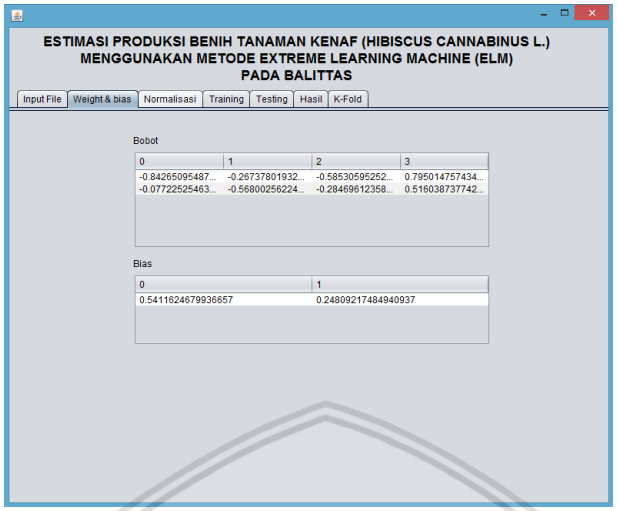


Gambar 5.1 Implementasi Antarmuka Halaman *Input File*

5.2.2 Implementasi Antarmuka Halaman*Weight&*Bias

Pada antarmuka halaman *weight&*bias berfungsi untuk mendapatkan nilai *weight* dan bias secara *random.* Didalam panel halaman tersebut terdapat Tabel *weight* dan bias. Jumlah *weight* dan bias sama dengan jumlah *input* pada *neuron*.

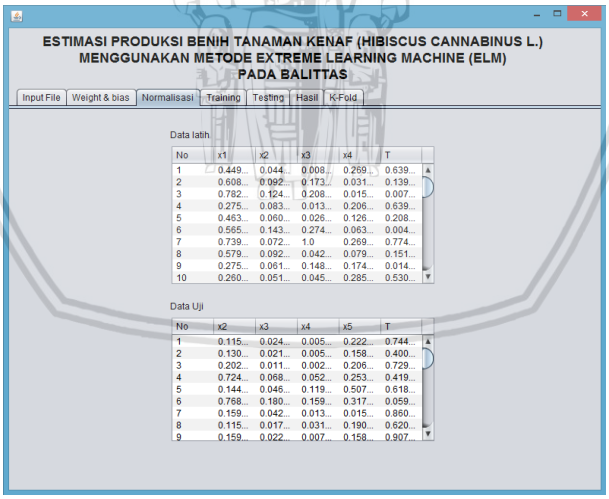
Pada Gambar 5.2 akan ditunjukkan mengenai implementasi antarmuka halaman *weight* dan bias.



Gambar 5.2 Implementasi Antarmuka Halaman *Weight*&Bias

5.2.3 Implementasi Antarmuka Halaman Normalisasi

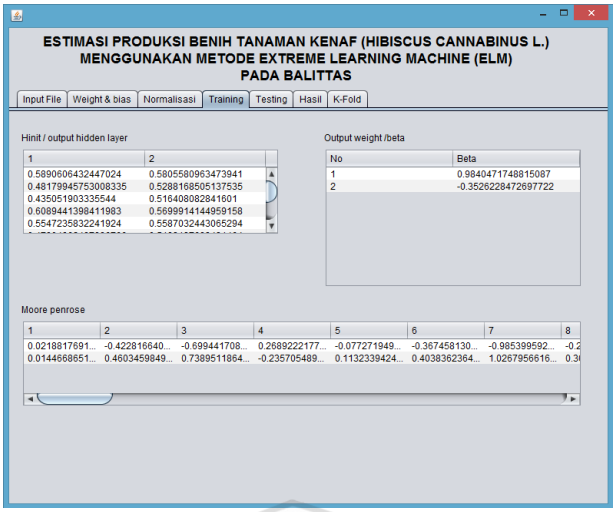
Pada halaman antarmuka normalisasi terdapat hasil perhitungan normalisasi menggunakan *min-max* normalization. Dalam halaman ini data sudah dibagi menjadi data yang digunakan untuk proses *training* dan *testing* sesuai dengan persentase pembagian data sebelumnya. Pada Gambar 5.3 akan ditunjukkan mengenai implementasi halaman antarmuka normalisasi.



Gambar 5.3 Implementasi Antarmuka Halaman Normalisasi

5.2.4 Implementasi Antarmuka Halaman *Training*

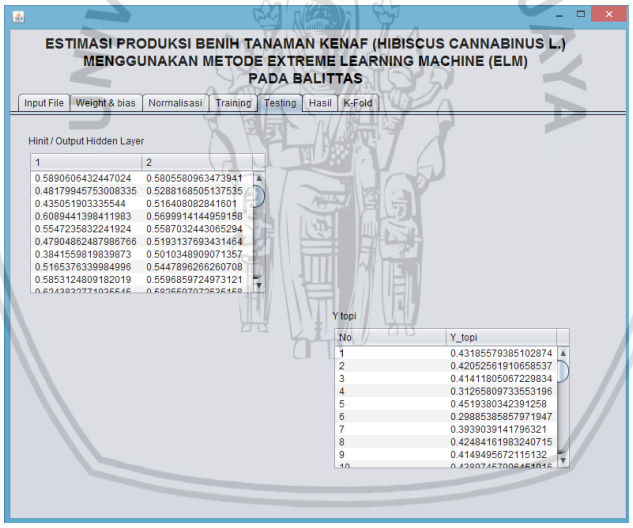
Pada halaman *training* terdapat proses perhitungan untuk melakukan pelatihan. Diantara proses tersebut antara lain adalah mencari nilai *output hidden layer/hinit,* matriks *moore penrose pseudo inverse, output weight/beta.* Pada Gambar 5.4 akan ditunjukkan mengenai Gambar implementasi antarmuka halaman *training.*



Gambar 5.4 Implementasi Antarmuka Halaman *Training*

5.2.5 Implementasi Antarmuka Halaman *Testing*

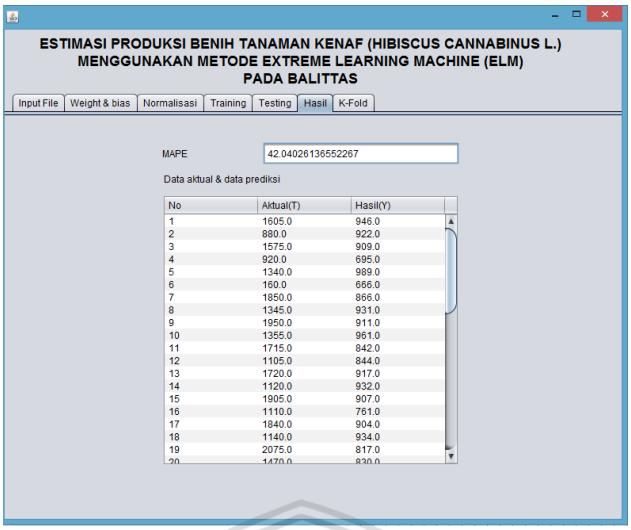
Pada halaman *testing* terdapat proses perhitungan untuk mencari nilai *output hidden layer/hinit,* serta mencari nilai estimasi atau y. Pada Gambar 5.5 akan ditunjukkan mengenai Gambar implementasi antarmuka halaman *testing*.



Gambar 5.5 Implementasi Antarmuka Halaman *Testing*

5.2.6 Implementasi Antarmuka Halaman Hasil

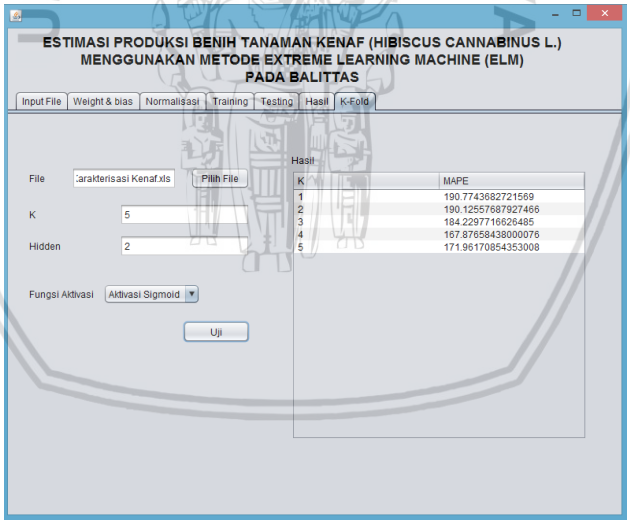
Pada halaman hasil terdapat evaluasi MAPE serta nilai hasil estimasi yang sudah dilakukan denormalisasi data serta perbandingan dengan nilai aktual sebelum dilakukan normalisasi. Pada Gambar 5.6 akan ditunjukkan mengenai implementasi antarmuka halaman hasil.



Gambar 5.6 Implementasi Antarmuka Halaman Hasil

5.2.7 Implementasi Antarmuka Halaman *K-Fold Cross Validation*

Pada halaman ini terdapat pengujian untuk melakukan validasi terhadap data serta parameter-parameter yang digunakan sebelumnya. Pada halaman *K-Fold* terdapat *input*an berupa *input file,* nilai k, jumlah *neuron,* fungsi aktivasi, serta tombol uji untuk melakukan pengujian. Pada Gambar 5.7 akan ditunjukkan mengenai implementasi antarmuka halaman *k-fold.*



Gambar 5.7 Implementasi Antarmuka Halaman *K-Fold Cross Validation*

BAB 6 PENGUJIAN DAN ANALISIS

Pada bab ini akan melakukan pengujian mengenai sistem yang telah dibuat dan dilakukan perancangan pada Bab 4. Setelah dilakukan pengujian, akan dilakukan analisis terhadap hasil dari pengujian tersebut. Sesuai dengan perancangan pengujian pada Bab 4 terdapat beberapa pengujian yang akan dilakukan diantaranya adalah pengujian persentase perbandingan data *training* terhadap data *testing*, jumlah *neuron* pada *hidden layer,* pengujian fungsi aktivasi dan pengujian validasi menggunakan *k-fold cross validation.*

6.1 Pengujian dan Analisis Jumlah *Neuron* Pada *Hidden Layer*

Pengujian ini dilakukan untuk mengetahui pengaruh jumlah *neuron* pada *hidden layer* terhadap nilai MAPE. Hasil dari pengujian tersebut bernilai bervariasi ketika sistem dijalankan, dikarenakan *input weight* serta bias yang dihasilkan bernilai *random.* Pada pengujian jumlah *neuron* pada *hidden layer* ini dilakukan sebanyak 10 kali uji coba. Hasil evaluasi nilai MAPE kemudian dirata-ratakan.

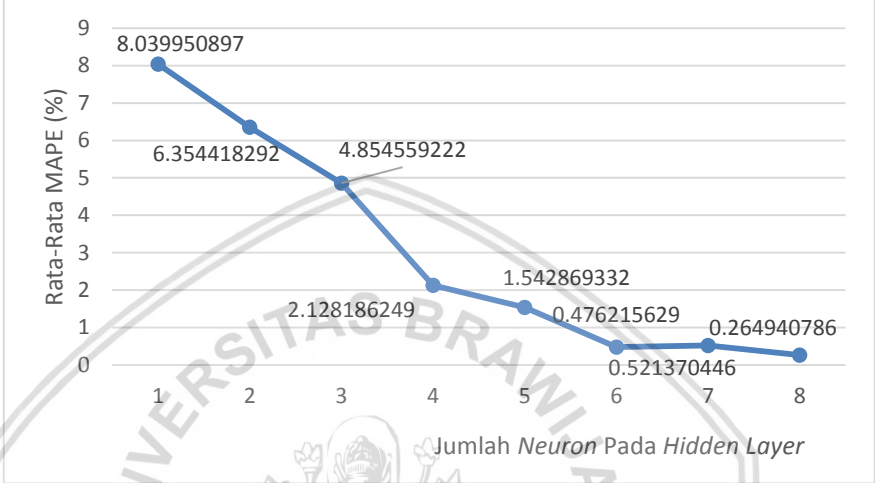
Pada proses pengujian ini, jumlah *neuron* pada *hidden layer* yang akan diuji bervariasi, yakni bernilai 1 hingga 8. Dalam melakukan pengujian jumlah *neuron* menggunakan persentase perbandingan data latih dengan data uji sebesar 90%:10%, berdasarkan hasil terbaik pada pengujian yang dilakukan sebelumnya. Fungsi aktivasi yang digunakan pada proses pengujian ini adalah fungsi aktivasi *sigmoid biner.* Pada Tabel 6.1 akan ditunjukkan mengenai hasil uji coba jumlah *neuron* pada *hidden layer* dan pada Gambar 6.1 merupakan grafik pengujian tersebut.

Pada Tabel 6.1 dan Gambar 6.1 menunjukkan bahwa dengan semakin banyaknya jumlah *neuron* pada *hidden layer* yang dihasilkan maka semakin kecil nilai MAPE yang didapatkan. Dapat dilihat bahwa, pada saat menggunakan jumlah *neuron* sebesar 8 maka menghasilkan rata-rata nilai MAPE sebesar 0,26%. Kemudian sebaliknya, apabila jumlah *neuron* yang dihasilkan semakin sedikit, maka nilai MAPE yang dihasilkan pun semakin besar. Dengan adanya hasil tersebut dapat disimpulkan bahwa dengan adanya jumlah *neuron* pada *hidden layer* yang semakin banyak maka penghubung antara proses *input layer* menuju *output layer* akan semakin banyak sehingga sistem dapat melakukan pengenalan terhadap pola pembelajaran dengan baik.

Tabel 6.1 Pengujian Jumlah *Neuron* Pada *Hidden Layer*

Jumlah *Neuron*	Nilai MAPE (%) Percobaan ke-i										Rata-rata Mape (%)
	1	2	3	4	5	6	7	8	9	10	
1	8,41	9,44	7,91	8,14	7,90	8,01	6,89	9,700	6,69	7,25	8,03
2	5,48	7,54	6,66	6,88	6,28	7,44	9,36	4,14	4,09	15,61	6,35
3	6,34	2,97	2,39	4,62	5,98	4,02	4,65	4,20	4,07	9,27	4,85

4	2,99	2,61	1,80	1,82	1,92	2,05	4,23	1,31	1,28	1,27	2,12
5	0,80	0,93	1,48	1,62	2,20	2,95	2,34	1,13	1,51	0,43	1,54
6	0,15	0,37	0,25	0,29	0,47	1,16	10,40	0,61	0,39	0,62	0,47
7	0,65	0,20	0,50	1,19	0,14	0,36	0,46	0,65	0,93	0,09	0,52
8	0,12	0,28	2,80	0,10	0,05	0,32	0,25	0,26	0,05	0,37	0,26



Gambar 6.1 Grafik Pengujian Jumlah *Neuron* Pada *Hidden Layer*

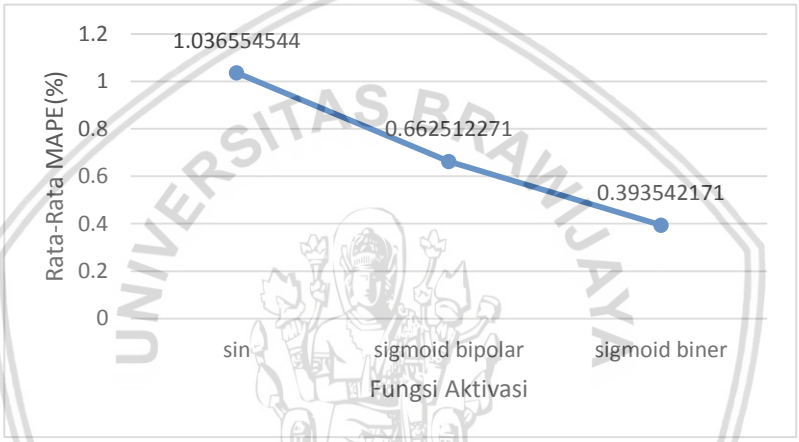
6.2 Pengujian dan Analisis Fungsi Aktivasi

Pengujian fungsi aktivasi dilakukan agar mengetahui seberapa besar pengaruh fungsi aktivasi tersebut dalam mendapatkan nilai rata-rata MAPE. Pada proses pengujian ini dilakukan dengan cara mengubah fungsi aktivasi yang akan digunakan pada saat menjalankan sistem. Fungsi aktivasi yang akan diuji sebanyak 3, yaitu fungsi aktivasi *sigmoid biner,* fungsi aktivasi *sigmoid bipolar* dan fungsi aktivasi *sin.* Dalam pengujian kali ini persentase perbandingan data yang digunakan adalah 90% data *training* dan 10% data *testing.* Jumlah *neuron* pada *hidden layer* yang digunakan adalah sebanyak 8.

Pada proses pengujian ini menghasilkan rata-rata nilai MAPE yang bervariasi. Hal tersebut dikarenakan *input weight* dan bias yang digunakan bernilai *random.* Pada Tabel 6.2 akan ditunjukkan mengenai hasil pengujian fungsi aktivasi dan pada Gambar 6.2 akan ditunjukkan mengenai grafik hasil pengujian tersebut.

Tabel 6.2 Pengujian Fungsi Aktivasi

Fungsi Aktivasi	Nilai MAPE (%) Percobaan ke-i										Rata-rata Mape (%)
	1	2	3	4	5	6	7	8	9	10	
Sin	0,30	0,24	2,88	0,57	0,88	0,73	1,99	1,32	0,17	1,24	1,03
Sigmoid Bipolar	0,19	0,65	0,17	0,16	0,53	0,04	3,77	0,64	0,15	0,28	0,66
Sigmoid Biner	0,23	0,47	0,16	0,22	0,24	0,12	0,57	0,08	1,62	0,16	0,39



Gambar 6.2 Grafik Pengujian Fungsi Aktivasi

Pada Tabel 6.2 dan Gambar 6.2 dapat dilihat bahwa tiap-tiap fungsi aktivasi hanya menghasilkan nilai rata-rata MAPE yang tidak berbeda jauh. Masing-masing fungsi aktivasi memiliki fungsi yang berbeda. Akan tetapi terlihat bahwa fungsi aktivasi *sigmoid biner* memiliki tingkat akurasi rata-rata nilai MAPE terkecil yakni sebesar 0,393%. Hal tersebut dikarenakan rentan nilai pemetaan berada pada nilai 0,1. Pada dasarnya ketiga fungsi aktivasi tersebut memiliki pengaruh yang cukup baik terhadap sistem, dikarenakan nilai MAPE yang dihasilkan rendah. Sehingga kita dapat menggunakan ketiga fungsi aktivasi tersebut dalam implementasi metode ELM.

6.3 Pengujian dan Analisis Persentase Perbandingan Jumlah Data *Training* Terhadap Data *Testing*

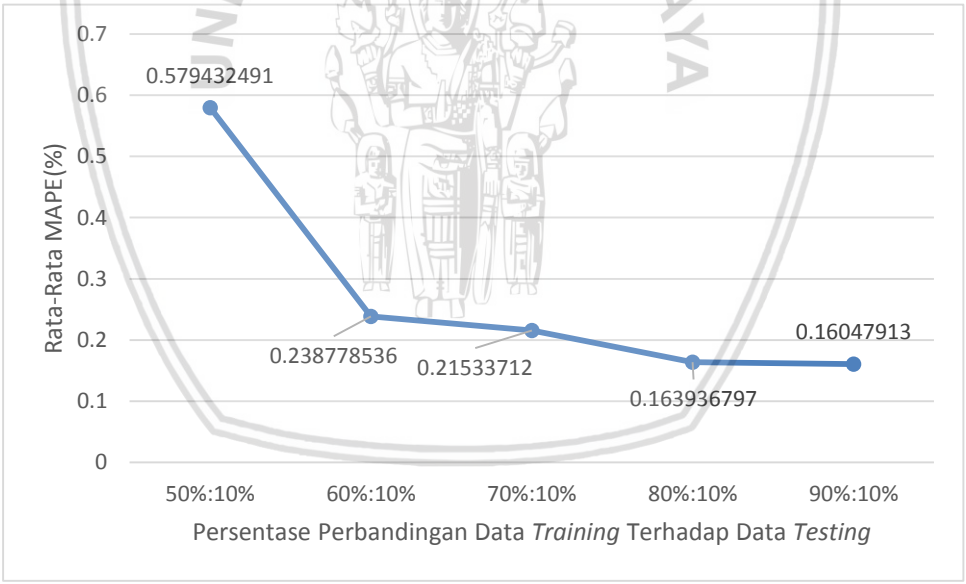
Pada pengujian ini dilakukan agar mengetahui seberapa besar pengaruh jumlah data *training* terhadap data *testing* sebesar 10% . Hasil rata-rata nilai MAPE bervariasi dikarenakan jumlah *input weight* serta bias bernilai *random.* Pengujian ini dilakukan sebanyak 10 kali uji coba dan didapatkan nilai rata-rata MAPE.

Pada saat melakukan pengujian,jumlah data *training* bervariasi yaitu 90%, 80%, 70%, 60%, 50%. Jumlah *neuron* pada hidden *layer* yang digunakan adalah

sebesar 8 berdasarkan hasil terbaik pada pengujian sebelumnya. Kemudian fungsi aktivasi yang digunakan pada pengujian ini adalah fungsi aktivasi *sigmoid biner*. Pada Tabel 6.3 akan ditunjukkan mengenai hasil pengujian data *training* terhadap data *testing* sebesar 10% dan pada Gambar 6.3 akan ditunjukkan mengenai grafik pengujian tersebut.

Tabel 6.3 Pengujian Persentase Perbandingan Jumlah Data *Training* Terhadap Data *Testing*

Persentase Perbandingan Jumlah Data *Training* Terhadap Data *Testing*	Nilai MAPE (%) Percobaan ke-i										Rata-rata Mape (%)
	1	2	3	4	5	6	7	8	9	10	
50% : 10%	0,21	0,23	0,59	0,19	0,36	0,56	0,57	2.14	0,49	0,42	0,57
60% : 10%	0,19	0,29	0,19	0,18	0,20	0,45	0,19	0,21	0,15	0,30	0,23
70% : 10%	0,37	0,23	0,11	0,06	0,27	0,30	0,10	0,33	0,13	0,20	0,21
80% : 10%	0,16	0,16	0,09	0,14	0,29	0,09	0,23	0,03	0,23	0,16	0,16
90% : 10%	0,10	0,21	0,25	0,11	0,13	0,17	0,18	0,16	0,17	0,08	0,16



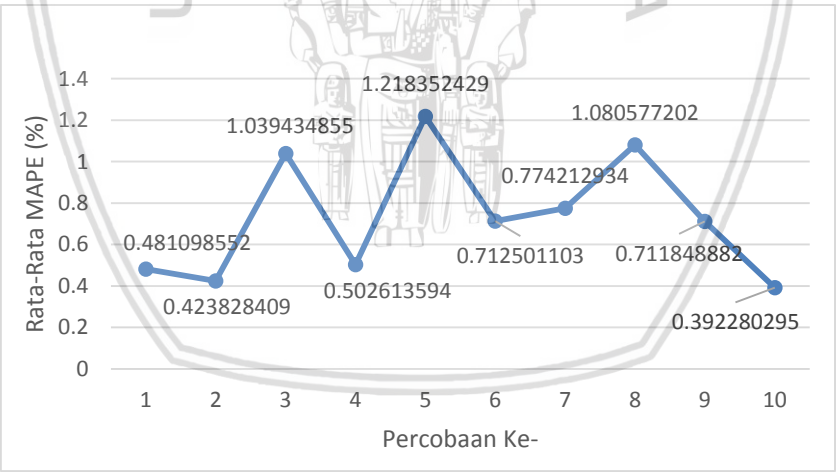
Gambar 6.3 Grafik Pengujian Persentase Perbandingan Data *Training* Terhadap Data *Testing*

Berdasarkan Tabel 6.3 dan Gambar 6.3 menunjukan bahwa dengan menggunakan data *training* yang semakin banyak maka nilai rata-rata MAPE yang dihasilkan semakin kecil. Hal tersebut ditunjukkan ketika menggunakan data *training* sebanyak 90% maka menghasilkan nilai MAPE yang kecil yaitu sebesar 0,16%. Akan tetapi sebaliknya, apabila menggunakan data *training* yang berjumlah sedikit, maka akan menghasilkan nilai MAPE yang lebih besar. Hal tersebut ditunjukkan apabila menggunakan data *training* sebanyak 50% maka nilai rata-

rata MAPE yang dihasilkan adalah sebesar 0,57%. Nilai MAPE yang tinggi disebabkan adanya *underfitting.* Pada proses pengujian ini *underfitting* terjadi karena data *training* berjumlah sedikit, sehingga pada saat proses pembelajaran tidak dapat menangkap pola dengan baik. Dengan adanya hal tersebut dapat disimpulkan bahwa metode ELM merupakan metode yang mengedepankan proses pembelajaran, sehingga semakin banyak data *training* yang digunakan maka akan semakin baik pula nilai estimasi yang dihasilkan. Selain itu nilai MAPE yang dihasilkan pun akan semakin baik.

6.4 Pengujian *K-Fold Cross Validation*

Pengujian *K-Fold Cross Validation* digunakan sebagai bentuk untuk melakukan validasi terhadap data serta parameter yang digunakan. Pada pengujian ini nilai data diambil secara random bergantung dengan nilai k yang akan diuji. Hal ini yang akan menimbulkan nilai evaluasi MAPE yang berbeda-beda. Agar mendapatkan hasil yang terbaik maka dilakukan 10 kali percobaan untuk masing-masing nilai k. Pada pengujian kali ini akan dilakukan 2 skenario pengujian. Pengujian pertama menggunakan nilai k sebesar 5 dan pengujian ke-2 menggunakan nilai k sebesar 10. Pada Tabel 6.4 akan ditunjukkan mengenai hasil pengujian *k-fold cross validation* dengan nilai k sebanyak 5 dan grafik pengujian akan ditunjukkan pada Gambar 6.4. Pada Tabel 6.5 akan menunjukkan Tabel pengujian dengan nilai k sebanyak 10 dan grafik pengujian akan ditunjukkan pada Gambar 6.5.



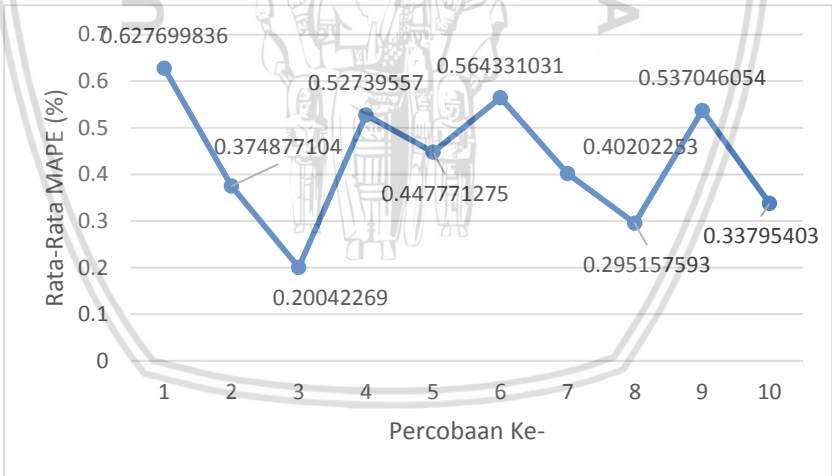
Gambar 6.4 Grafik Pengujian *K-Fold Cross Validation K=5*

Tabel 6.4 Pengujian *K-Fold Cross Validation* dengan K=5

Percobaan ke-	Subset Ke-					Rata-rata MAPE (%)
	1	2	3	4	5	
1	0,11	0,19	1,65	0,27	0,16	0,48
2	0,56	0,13	0,76	0,19	0,46	0,42
3	2,06	0,31	1,82	0,68	0,30	1,03
4	0,76	0,40	0,50	0,46	0,36	0,50

Percobaan ke-	Subset Ke-					Rata-rata MAPE (%)
	1	2	3	4	5	
5	0,58	0,12	3,71	0,24	1,42	1,21
6	0,34	0,15	2,06	0,48	0,50	0,71
7	0,58	0,37	1,32	0,72	0,81	0,77
8	1,89	0,05	1,02	0,97	2,33	1,08
9	0,63	0,30	1,86	0,46	0,46	0,71
10	0,80	0,40	0,15	0,36	0,36	0,39
Rata Rata MAPE (%)						0,73

Berdasarkan Tabel 6.4 dan Gambar 6.4 menunjukkan bahwa dengan menggunakan nilai k sebesar 5 menunjukkan hasil yang baik yaitu menghasilkan nilai rata-rata MAPE sebesar 0,73%. Dengan menggunakan nilai k=5 maka data akan dikelompokkan sebanyak 20 pada masing-masing fold. Nilai rata-rata MAPE yang dihasilkan bervariasi dikarenakan *input weight* dan bias yang dihasilkan adalah bernilai *random*. Selain itu, urutan data pun dapat mempengaruhi nilai MAPE yang dihasilkan. Data dengan pengurutan yang baik akan menghasilkan nilai MAPE yang baik pula.



Gambar 6.5 Grafik Pengujian K-Fold Cross Validation K=10

Tabel 6.5 Pengujian *K-Fold Cross Validation* dengan K=10

Percobaan ke-	Subset Ke-										Rata-rata MAPE (%)
	1	2	3	4	5	6	7	8	9	10	
1	1,90	0,19	0,33	0,21	0,50	1,41	0,23	0,19	0,26	1,02	0,62
2	0,58	0,27	0,07	0,58	0,21	0,68	0,62	0,08	0,28	0,33	0,37
3	0,45	0,22	0,04	0,50	0,06	0,11	0,06	0,59	0,20	0,10	0,20

Percobaan ke-	Subset ke-										Rata-Rata MAPE (%)
	1	2	3	4	5	6	7	8	9	10	
4	0,21	0,07	0,29	0,21	0,28	0,11	0,23	0,27	1,05	0,68	0,52
5	1,50	0,25	0,21	0,06	0,42	0,61	0,25	0,02	0,09	1,04	0,44
6	1,76	0,25	0,38	0,28	0,15	1,56	0,43	0,37	0,11	0,46	0,56
7	0,95	0,25	0,10	0,42	0,19	0,53	0,50	0,28	0,37	0,76	0,40
8	1,19	0,18	0,18	0,15	0,40	0,20	0,10	0,14	0,27	0,09	0,29
9	1,21	0,11	0,09	0,34	0,30	1,45	1,07	0,12	0,23	0,40	0,53
10	0,50	0,28	0,25	0,27	0,34	0,25	0,63	0,37	0,09	0,35	0,33
Rata-Rata Nilai MAPE (%)											0,43

Berdasarkan Tabel 6.5 dan Gambar 6.5 yang menunjukan grafik pengujian *k-fold cross validation,* nilai rata-rata MAPE yang dihasilkan dengan menggunakan k 10 adalah sebesar 0,43%. Hal tersebut menunjukkan hasil yang baik bila dibandingkan dengan pengujian *k-fold* menggunakan k sebesar 5 dikarenakan jumlah data *training* yang diuji lebih banyak jika dibandingkan dengan k sebanyak 5. Urutan data yang digunakan dapat mempengaruhi nilai MAPE yang dihasilkan. Sehingga data dengan pengurutan nilai yang baik akan menghasilkan nilai *MAPE* yang baik pula. Jumlah *neuron* pada *hidden layer, input weight* dan bias secara *random* juga dapat mempengaruhi nilai MAPE yang dihasilkan.

Analisis hasil pengujian *k-fold cross validation* dengan k=5 dan k=10, membuktikan bahwa data yang digunakan dan sistem yang dibangun dapat dinyatakan valid. Data dan sistem dapat dinyatakan valid karena hasil pengujian menunjukkan bahwa nilai dari rata-rata MAPE yang didapatkan stabil. Perbedaan nilai rata-rata MAPE yang didapatkan dikarenakan perbedaan urutan data latih dan data uji yang digunakan.

BAB 7 PENUTUP

7.1 Kesimpulan

Berdasarkan pengujian serta pembahasan dari estimasi produksi benih menggunakan metode *Extreme Learning Machine* (ELM) maka didapatkan kesimpulan sebagai berikut.

1. Berikut merupakan proses dari implementasi dengan menggunakan metode *Extreme Learning Machine* (ELM) dalam melakukan estimasi hasil produksi benih tanaman kenaf.
    a. Melakukan normalisasi terhadap data menggunakan *min-max normalization.*
    b. Melakukan proses perhitungan *training* untuk mendapatkan nilai *output weight.*
    c. Melakukan proses perhitungan *testing* untuk mendapatkan nilai *output layer.*
    d. Melakukan denormalisasi pada data untuk mendapatkan hasil estimasi produksi benih tanaman kenaf.
    e. Melakukan evaluasi sistem menggunakan MAPE.
2. Dalam melakukan evaluasi terhadap sistem estimasi produksi benih tanaman kenaf, sistem ini menggunakan MAPE (*Mean Absolute Percentage Error).* Berikut adalah hasil evaluasi terhadap pengujian yang dilakukan:
    a. Pada pengujian jumlah *neuron* pada *hidden layer* didapatkan hasil bahwa jumlah *neuron* pada *hidden layer* mempengaruhi perhitungan dari suatu permasalahan. Hal tersebut dikarenakan jumlah *neuron* mengolah seluruh *inputan* yang nantinya akan menghasilkan *output.* Jumlah *neuron* terbaik yang dihasilkan adalah sebesar 8 dengan menghasilkan nilai rata-rata MAPE sebesar 0,264%.
    b. Pada pengujian fungsi aktivasi didapatkan hasil bahwa fungsi aktivasi yang terbaik pada sitem dengan metode ELM adalah fungsi aktivasi *sigmoid biner.* Rata-rata nilai MAPE yang dihasilkan oleh fungsi aktivasi *sigmoid biner* adalah sebesar 0,395%. Ketiga fungsi aktivasi yakni *sigmoid biner, sigmoid bipolar* dan *sin* merupakan fungsi aktivasi yang baik digunakan untuk perhitungan ELM karena menghasilkan nilai MAPE kurang dari 10%. Sehingga ketiganya dapat digunakan pada sistem.
    c. Pada pengujian persentase perbandingan data *training* terhadap data *testing* didapatkan hasil bahwa data *training* yang digunakan pada sistem dengan hasil terbaik adalah 90%, dengan nilai rata-rata MAPE sebesar 0,160%. Semakin banyak data *training* yang digunakan maka semakin baik pola yang dihasilkan.

d. Pada pengujian validasi sistem menggunakan *k-fold cross validation,* sistem estimasi produksi benih menggunakan ELM ini menggunakan 2 nilai k pada proses pengujian *k-fold* yakni k=5 dan k=10. Akurasi sistem terbaik terdapat pada nilai k dengan jumlah 10. Pada nilai k=10 terdapat pembagian subset dengan nilai masing-masing 10 di setiap *fold* nya.Nilai rata-rata MAPE yang dihasilkan adalah sebesar 0,43%.

7.2 Saran

Berdasarkan penelitian estimasi produksi benih tanaman kenaf menggunakan metode ELM, terdapat saran yang dapat dikembangkan untuk penelitian selanjutnya yaitu untuk penelitian berikutnya, peneliti dapat menambahkan parameter-parameter pendukung lainnya dalam melakukan estimasi terhadap produksi benih tanaman kenaf. Hal tersebut guna didapatkan hasil yang lebih obyektif dalam melakukan estimasi. Karena semakin banyak parameter yang dihasilkan, maka hasil yang didapatkan pun akan semakin baik.

DAFTAR PUSTAKA

Balittas, 2015. *Balai Penelitian Tanaman Pemanis dan Serat.* [Online] Available at: http://balittas.litbang.pertanian.go.id/index.php/tentang-pui [Accessed 5 12 2017].

Cahyanti, D. D., 2017. Penerapan Regresi Linier Berganda Untuk Penelitian Tanaman Kenaf Di Balai Penelitian Tanaman Pemanis Dan Serat. *Jurnal Statistika,* p. 1.

Chang, P. C., Wang, Y. W. & Liu, C. H., 2007. The Development of a Weighted Envolving Fuzzy Neural Network for PCB Sales Forecasting. *Expert System with Applications,* Volume 32, pp. 86-89.

D., S., S. & D., 2009. *Kenaf : Monograf Kenaf.* Karang Ploso Malang: Balai Penelitian Tembakau dan Serat.

Hasanah, M., 2002. Peran Mutu Fisiologik Benih dan Pengembangan Industri Benih Tanaman Industri. *Jurnal Litbang Pertanian,* Volume 21(3), pp. 84-91.

Hasanah, M., 2002. Peran Mutu Fisiologik Benih Dan Pengembangan Industri Benih Tanaman Industri. *Jurnal pertanian litbang,* Volume 21(3), pp. 85-90.

Hidayatullah, N. U. & Umar, Y., 2014. Estimasi Radiasi Matahari Perjam Pada Permukaan Horizontal dengan Extreme Learning Machine (Studi Kasus di Surabaya). *Jurnal Teknik POMITS,* Volume 2, pp. 2301-9271.

Hossain, M. D., Hanafi, M. M., Jol, J. & Hazandy, A. H., 2011. Growth, yield and fiber morphology of kenaf (Hibiscus Cannabinus L.) grown on sandy bris soil as ilfluenced by different levels of carbon. *African Journal of Biotechnology ,* Volume 10(50), pp. 10087-10094.

Huang, G. B., Zhu, Q. Y. & Siew, C. K., 2006. Extreme Learning Machine : A New Learning Scheme Of Feedforward Neural Networks. *Proedings of International Joint Conference on Neural Networks.*

Huang, G. B., Zhu, Q. Y. & Siew, C. K., 2006. Extreme Learning Machine : Theory and Application. *Elsevier (Scient Direct),* Volume 70, pp. 489-501.

Jain, Y. K. & Bhandare, S. K., 2011. Min Max Normalization Based Data Pertubation Method For Privacy Proection. *Computer Science & Engineering,* 2(VIII), pp. 45-50.

Kim, S. & Kim, H., 2016. A New Metric Of Absolute Percentage Error For Intermittent Demand Forecast. *International Journal of Forecasting,* Volume 32, pp. 669-679.

Kohavi, R., 2015. A Study of Cross Validation and Bootstrap for Accuracy Estimation and Model Selection. *Computer Science.*

Liang, N. Y., Huang , G. B., Saratchandran, P. & Sundararajan, N., 2006. A Fast And Accurate Online Sequential Learning Algorithm For Feeforward Networks. *IEEE Transactions on Neural Network,* Volume 17(6), pp. 1411-1423.

Marjani, 2017. *Kendala Pada BalittasDalam Memprediksi Produksi Benih Kenaf* [Interview] (20 December 2017).

Naji, S., Keivani, A., Shamshirband, S. & Alengaram, J., 2015. Estimating Building Energi Consumption Using Extreme Learning Machine. *Elsevier,* Volume 97, pp. 506-516.

Nurdjajati, E., Sunaryuni, E., Utami, D. A. & Bahri, S., 2009. *Monograf Kenaf.* Malang: Balai Tanaman Tembakau dan Serat.

Pati, J. C., Mishra, K. K. & Patnaik, S. K., 2013. A Comparative study on Short term load forecasting using BPNN and Extreme Learning Machine. *InternationalJournal of Advanced Research in Science and Technology,* 2(1), pp. 30-34.

Shamshirband, S., Mohammadi, K., Tong, C. W. & Petkovic, D., 2015. Application of Extreme Learning Machine For Estimation of Wind Speed Distribution. *Journal of Estimation Method,* Volume 46, pp. 1893-1907.

Sheela, K. G. & Deepa, S. N., 2013. Review on Method to FIx Number of Hidden Neurons in Neural Networks. *Mathematical Problems in Engineering,* Volume 23, p. 11.

Singh, R. & Balasundaram, S., 2007. Application of Extreme Learning Machine Method for Time Series Analysis. *Journal of Computer and Information Engineering,* Volume 1(11).

Sinuhaji, F., 2009. Jaringan Syaraf Tiruan Untuk Prediksi Keputusan Medis Pada Penyakit Asma. *Skripsi S1.*

Srimuang, W, Intarashothonchun & S, 2015. Classification Model of Network Intrusion Using Weighted Extreme Learning Machine. *International Joint Conference on Computer Scine And Software Engineering (JCSSE),* Volume 12.

Sudjindro, 2007. Kenaf (Hibicus cannabinus) Sebagai Alternatif Bahan Baku Pulp. *Tabloid Sinar Tani,* pp. 1-3.

Suryadi, S., 2000. Kuantfikasi Metabolisme Benih. *Jurnal Litbang Pertanian,* Volume 20, pp. 34-35.

Wuryandari, M. D. & Afrianto, I., 2012. Perbandingan Metode Jaringan Syaraf Tiruan Backpropagation dan Learnong Vector Quantization Pada Pengenalan Wajah. *Jurnal Komputer dan Informatika,* Volume 1, pp. 45-50.