PARTICLE SWARM OPTIMIZITATION UNTUK OPTIMASI BOBOT *EXTREME LEARNING MACHINE* DALAM MEMPREDIKSI PRODUKSI GULA KRISTAL PUTIH PABRIK GULA CANDI BARU-SIDOARJO

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:
Eka Yuni Darmayanti
NIM: 145150201111012

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

PENGESAHAN

PARTICLE SWARM OPTIMIZITATION UNTUK OPTIMASI BOBOT *EXTREME LEARNING MACHINE* DALAM MEMPREDIKSI PRODUKSI GULA KRISTAL PUTIH PABRIK GULA CANDI BARU-SIDOARJO

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh :
Eka Yuni Darmayanti
NIM : 145150201111012

Skripsi ini telah diuji dan dinyatakan lulus pada
07 Juni 2018
Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I	Dosen Pembimbing II
Budi Darma Setiawan, S.Kom, M.Cs	Dr.Eng. Fitra Abdurrachman Bachtiar, S. T, M.Eng
NIP: 19841015 201404 1 002	NIK: 201201 840628 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 disuatu perguruan tinggi, dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali 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-undnagan yang berlaku (UU No. 20 Tahun 2003, Pasal 25 ayat 2 dan pasal 70).

Malang, 28 Juni 2018

Eka Yuni Darmayanti

NIM: 145150201111012

KATA PENGANTAR

Puji syukur penulis panjatkan kepada Tuhan Yang Maha Esa telah melimpahkan rahmat-Nya sehingga laporan skripsi yang berjudul "*PARTICLE SWARM OPTIMIZATION* UNTUK OPTIMASI BOBOT *EXTREME LEARNING MACHINE* DALAM MEMPREDIKSI PRODUKSI GULA KRISTAL PUTIH PABRIK GULA CANDI BARU-SIDOARJO" ini dapat terselesaikan.

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

1. Budi Darma Setiawan, S.Kom, M.Cs, selaku dosen pembimbing I yang telah memberikan bimbingan, arahan ilmu, dan masukan kepada penulis dalam menyelesaikan skripsi ini.
2. Dr.Eng. Fitra Abdurrachman Bachtiar, S. T, M.Eng, selaku dosen pembimbing II yang telah memberikan bimbingan, memberikan saran dan memberikan arahan kepada penulis selam penyusunan skripsi ini.
3. Bapak Wayan Firdaus Mahmudy, S.Si, M.T, Ph.D., Bapak Heru Nurwasito Ir., Bapak M.Kom, marji, Drs., M. T, Bapak Edy Santoso , S.Si, M.Kom, selaku Dekan, Wakil Dekan I, Wakil Dekan II, Wakil Dekan II Fakultas Ilmu Komputer Universitas Brawijaya.
4. Bapak Abdullah selaku bapak kandung penulis, Ibu Faridaningsih selaku ibu kandung penulis, Wais Qurnain dan Mutia Nayla Nurida selaku adik kandung penulis, serta seluruh keluarga besar atas nasehat, perhatian, kasih sayang serta kesabaran dalam membesarkan penulis, serta tiada henti-hetinya memberikan doa dan semangat demi terselesaikannya skripsi ini.
5. Seluruh teman-teman Teknik Informatika angkatan 2014 dan khususnya sahabat penulis Anggita Mahardika S.Kom, Yane Marita F. S.Kom, Amelia Achsandini yang segera mendapat gelar S.Kom-nya, Vina Meilia yang juga akan segera mendapat gelar S.Kom-nya, Pratomo Adinegoro S.Kom, Nur Afifah Sugianto S.Kom, Regina Anky Chandra S.Kom, Firdaus Rahman S.Kom, Maria Sartika Tambun S.Kom yang selalu memberikan semangat, motivasi dan dukungan kepada penulis selama menyelesaikan skripsi ini.
6. Seluruh civitas akademika Teknik Informatika Universitas Brawijaya yang telah banyak memberi bantuan dan dukungan selama penulis menempuh studi di Teknik Informatika Universitas Brawijaya dan selama penyelesaian skripsi ini.
7. Kakak-kakak, teman-teman, dan adik-adik pengurus himpunan yang tidak bisa penulis sebutkan satu persatu, yang telah memberikan banyak pelajaran non-akademik selama masa perkuliahan yang tidak pernah penulis dapatkan sebelumnya, sehingga selama masa perkuliahan penulis tidak hanya mendapatkan ilmu dalam bidang akademik. Teman-teman

yang senantiasa memberikan dukungan, semangat dan motivasi serta menjadi keluarga kedua bagi penulis.

8. Kakak-kakak, teman-teman, adik-adik anggota KpopKdrama_World yang selalu memberikan dorongan, motivasi dan semangat kepada penulis dalam menyelesaikan skripsi ini.
9. Teman-teman seperantauan yang sedang berjuang mengerjakan tugas akhir Pretty Septiana dan One Grahita D.L yang telah memberikan semangat kepada penulis.

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

Malang, 28 Juni 2018

Penulis
ekayunidarmayanti23@gmail.com

ABSTRAK

Permintaan gula akan meningkat seiring dengan peningkatan jumlah penduduk, pendapatan masyarakat, dan pertumbuhan industri pengolahan makanan dan minuman. Oleh karena itu agar proses produksi gula selalu meningkat sesuai dengan kebutuhan gula itu sendiri, maka diperlukannya perencanaan produksi. Peramalan yang akurat dapat membantu perusahaan dalam mengambil keputusan untuk menentukan jumlah gula yang akan diproduksi, bahan yang dibutuhkan dan menetukan harga barang. Salah satu metode yang dapat digunakan untuk melakukan prediksi adalah algoritme *Extreme Learning Machine*. Akan tetapi metode tersebut dalam pemilihan *input weight* dan bias dipilih secara acak, hal ini dapat menyebabkan hasil yang didapat dalam perhitungan kurang maksimal. Diperlukannya kombinasi dengan algoritme *Particle Swarm Optimization* yang dapat melakukan optimasi nilai *input weight* dan bias secara optimal. Penelitian ini menggunakan 45 data giling produksi gula dengan 5 fitur. Berdasarkan penelitian yang telah dilakukan, didapatkan parameter yang optimal yaitu jumlah ukuran populasi 50, perbandingan data training 80% (36 data), jumlah hidden neuron 10, bobot inersia 0.5, dan iterasi maksimal 250. dari parameter tersebut didapatkan nilai rata-rata MAPE sebesar 0.59%. Dari hasil rata-rata MAPE yang didapat, menunjukkan bahwa penambahan algoritme PSO pada ELM dapat menentukan nilai *input weight* dan bias yang optimal.

Kata kunci: prediksi, optimasi, *Extreme Learning Machine*, *Particle Swarm Optimization*

ABSTRACT

Sugar demand will increase in line with increase in population, income, and growth in food and beverage processing industry. Therefore, in order for the sugar production process is always increasing in accordance with needs of the sugar itself, hence need for production planning. Accurate forecasting can help companies in taking decisions to determine the amount of sugar to be produced, the materials needed and determine the price of the goods. One method that can be used to do the prediction algorithm is Extreme Learning Machine. But that method in selection of input and weight bias is chosen randomly, this can lead to the results obtained in the calculation less maximum. This need for a combination with Particle Swarm Optimization algorithms that can perform optimization the input value weight and bias optimally. This research uses data 45 milled sugar production with 5 features. Based on the research that has been performed, the obtained optimal parameters, namely the number of population size 50, 80% training data comparison (36), the number of hidden neuron 10, weigh inertia 0.5, and a maximum of iterations 250. The parameter value is obtained from the average MAPE of 0.59%. From the average MAPE results obtained, shows that the addition of the PSO algorithm on ELM can determine the value of input of weight and optimal bias.

Keywords: *prediction, optimization, Extreme Learning Machine, Particle Swarm Optimization*

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR LAMPIRAN

BAB 1 PENDAHULUAN

1.1 Latar belakang

Salah satu komoditas yang cukup strategis dan memegang peranan penting di sektor pertanian khususnya sub sektor perkebunan dalam hal perekonomian nasional adalah komoditas gula. Gula sendiri merupakan kebutuhan pokok rakyat yang cukup strategis sebagai bahan pangan sumber kalori yang menempati urutan keempat setelah padi – padian, pangan hewani, serta minyak dan lemak, dengan pangsa sebesar 6,7 persen. Selain itu, gula juga merupakan salah satu sumber pemanis utama yang telah digunakan secara luas baik untuk keperluan konsumsi rumah tangga maupun bahan baku industri pangan. Hal ini terjadi dikarenakan gula mengandung kalori sehingga dapat menjadi salah satu sumber energi dan gula juga digunakan untuk bahan pengawet dan tidak membahayakan kesehatan.

Seiring dengan meningkatnya salah satu bahan pangan pokok konsumsi gula selalu mengalami peningkatan dari tahun ke tahun. Ketergantungan akan konsumsi gula cukup besar karena kecilnya atau lemahnya kecenderungan untuk mensubstitusikannya dengan gula buatan atau pemanis lain. Permintaan gula secara nasional akan terus meningkat seiring dengan peningkatan jumlah penduduk, pendapatan masyarakat, dan pertumbuhan industri pengolahan makanan dan minuman. Sebagai negara berpenduduk besar dengan tingkat pendapatan yang terus meningkat, maka Indonesia sangat potensial menjadi salah satu konsumen gula terbesar di dunia (Sugiyanto, 2007).

Salah satu Pabrik Gula yang ada di Indonesia khususnya Jawa Timur, yang memproduksi gula dalam jumlah banyak adalah Pabrik Gula Candi Baru yang berlokasi di kota Sidoarjo, Jawa Timur. Pabrik Gula Candi Baru merupakan perusahaan penghasil gula SHS (*Superior Hooft Sulker*) atau biasa disebut GKP (Gula Kristal Putih). Untuk menghasilkan GKP yang berkualitas baik ditinjau dari proses produksi yang dilakukan. Pada masa giling yang dilakukan oleh pabrik setiap tahunnya mengalami hasil produksi yang naik turun. Sedangkan permintaan akan gula selalu mengalami perubahan.

Agar proses produksi gula selalu meningkat sesuai dengan kebutuhan akan gula itu sendiri, maka di perlukannya perencanaan produksi. Perencanaan produksi adalah sebuah aktivitas untuk menetapkan produk yang diproduksi, jumlah yang dibutuhkan, kapan produk tersebut harus selesai dan sumber – sumber yang dibutuhkan lainnya. Dalam kegiatan memproduksi gula, PT Pabrik Gula Candi Baru sudah melakukan perencanaan produksi untuk dapat mengoptimalkan strategi produksi. Perencanaan produksi adalah suatu kegiatan untuk menetapkan produk yang diproduksi diantaranya jumlah yang dibutuhkan, kapan produk harus selesai dan sumber-sumber yang dibutuhkan. Namun perencanaan produksi ini masih

dilakukan secara manual yang memungkinkan terjadinya kesalahan dalam perhitungannya (Siwi, et al., 2016).

Peramalan yang akurat dan efektif dapat membantu pengambil keputusan dalam perusahaan untuk menentukan jumlah barang yang akan diproduksi, bahan baku yang dibutuhkan serta menentukan harga terhadap barang jadi sehingga perusahaan memiliki tingkat inventory rendah. Serta mampu merespon permintaan dari konsumen secara cepat. Banyak metode telah dikemukakan untuk mendapatkan hasil ramalan yang akurat. Salah satunya Jaringan Syaraf Tiruan (JST) yang mengadopsi dari kinerja sistem pembelajaran otak manusia. Banyak penelitian menyimpulkan bahwa metode JST lebih baik daripada metode-metode peramalan konvensional (Sun, et al., 2008).

Salah satu metode baru dalam JST adalah *Extreme Learning Machine* (ELM). ELM merupakan jaringan syaraf tiruan *feedforward* dengan satu *hidden layer* atau lebih dikenal dengan istilah *single hidden layer feedforward neural network* (SLNFs). Metode ELM sendiri mempunyai kelebihan dalam *learning speed*, serta mempunyai tingkat akurasi yang lebih baik dibandingkan dengan metode konvensional lainnya. Sehingga dengan diterapkannya metode ELM diharapkan mampu menghasilkan ramalan yang lebih efektif (Agustina, et al., 2010).

Seperti pada penelitian yang sudah dilakukan sebelumnya, penelitian yang dilakukan oleh Irwin Dwi Agustina (2010) yang berjudul "Penerapan Metode *Extreme Learning Machine* Untuk Peramalan Permintaan" dalam penelitian ini, peneliti membahas tentang metode ELM dalam melakukan peramalan permintaan terhadap perencanaan produksi toko "Cak Cuk Shop" pada penelitian ini meghasilkan tingkat kesalahan rendah dengan mengguakan MAPE yaitu sebesar 0.0042% pada produk kaos dan 0.0095% pada produk pin. Selain itu ada penelitian lain yang dilakukan oleh Iga Permata Siwi (2016) yang juga menggunakan algoritma ELM dengan penelitian yang berjudul "Peramalan Produksi Gula Pasir Menggunakan ELM pada PG Candi Baru Sidoarjo" pada penelitian ini tingkat error yang juga dihitung dengan nilai MAPE sebesar 0,74 %.

Salah satu penelitian yang menggunakan *Particle Swarm Optimization* (PSO) sebagai metode optimasi adalah penelitian dari Zhao, Zhang dan Han (2014). Pada penelitian ini peneliti menggunakan metode ELM lalu mengembangkannya dengan *Adaptive Grwoth of Hidden Nodes* (AG) dan PSO. Pada penelitian ini peneliti menggunakan PSO untuk menghitung input weight, output weight dan bias yang selanjutnya digunkan pada AG-ELM. Pada penelitian ini peneliti menyimpulkan bahwa pengembangan PSO-AG-ELM ini memiliki tingkat akurasi yang tinggi jika dibandingankan dengan AG-ELM dan ELM biasanya.

Penelitian yang juga menggunakan *hybrid algorithm* adalah penelitian dari Nur Afifah Sugianto (2017). Pada penelitian nya peneliti menggunakan metode ELM sebagai metode klasifikasi untuk pemilihan keminatan pada mahasiswa Informatika.

Akan tetapi pada metode ELM tidak memiliki kemampuan untuk melakukan seleksi fitur sehingga dikombinasikan metode ELM dengan metode PSO sebagai metode untuk menyeleksi fitur yang digunakan agar dapat menyeleksi fitur dengan otomatis dan optimal. Pada penelitian ini menghasilkan nilai akurasi sebesar 94.44% dibandingkan dengan hanya menggunakan metode ELM biasa yakni menghasilkan nilai sebesar 66,67%.

Selain keuntungan-keuntungan yang ada, ELM sendiri memilliki kelemahan yaitu, jumlah dari *hidden nodes* yang ada ditentukan dengan cara *try and error* , sehingga tidak dapat diketahui jumlah dari *hidden nodes* yang tepat untuk mendapatkan hasil peramalan yang tepat dengan menggunakan metode ELM. Selain permasalahan *hidden nodes* dalam melakukan pemilihan *input weight*, bias dalam metode ELM nilai tersebut dipilih secara acak atau *random*, maka hal ini dapat menyebabkan hasil yang didapat dalam perhitungan juga kurang maksimal (Handika, I P.S (2016)).

Oleh karena itu diperlukan metode optimasi untuk menghitung input weight dan bias yang akan digunakan pada peramalan dengan metode ELM. Berdasarkan permasalahan yang ada, dengan menggunakan *hybrid algorithm* ELM dan PSO dapat menghasilkan hasil yang lebih optimal dengan pembelajaran yang cepat. Sehingga pada penelitian kali ini peneliti mengusulkan penelitian tentang optimasi bobot *Extreme Learning Machine* dengan *Particle Swarm Optimization* dalam memprediksi produksi gula kristal putih agar dapat memberikan prediksi yang optimal dengan tingkat *error* yang rendah.

1.2 Rumusan masalah

Berdasarkan uraian latar belakang permasalahan diatas, maka dapat dirumuskan permasalahan yang akan dijadikan Sebuah rumusan masalah adalah sebagai berikut:

1. Bagaimana pengaruh parameter *Particle Swarm Optimization* (PSO) dalam optimasi bobot *Extreme Learning Machine* (ELM)?
2. Berapa tingkat kesalahan yang diukur dengan perhitungan *Mean Absolute Precentage Error* (MAPE) pada hasil produksi gula kristal putih dengan menggunakan metode *Extreme Learning Machine* (ELM) dan *Particle Swarm Optimization* (PSO)?
3. Bagaimana perbandingan tingkat kesalahan (MAPE) pada perhitungan peramalan ELM dan perhitungan peramalan ELM-PSO?

1.3 Tujuan

Tujuan dari penelitian ini adalah sebagai berikut:

1. Mengetahui pengaruh dari parameter Particle Swarm Optimization (PSO) dalam optimasi bobot *Extreme Learning Machine* (ELM).

2. Mengetahui tingkat kesalahan dengan *Mean Absolute Precentage Error* (MAPE) pada hasil produksi gula kristal putih dengan menggunakan metode *Extreme Learning Machine* (ELM) dan *Particle Swarm Optimization* (PSO).
3. Mengetahui perbandingan error pada perhitungan peramalan ELM dan perhitungan peramalan ELM-PSO

1.4 Manfaat

Manfaat yang diharapkan dari penelitian ini adalah sebagai berikut:

1. Membantu perhitungan optimasi hasil produksi gula kristal putih pada pabrik gula.
2. Sebagai salah satu sarana referensi untuk mengoptimasi metode.

1.5 Batasan masalah

Berdasarkan penjelasan – penjelasan sebelumnya, ruang lingkup atau batasan yang akan digunakan dalam perancangan untuk mengoptimasi produksi gula kristal putih adalah sebagai berikut:

1. Implementasi hybrid algorithm *Extreme Learning Machine* (ELM) dan *Particle Swarm Optimization* (PSO).
2. Data yang digunakan diambil dari data yang ada pada Pabrik Gula Candi Baru pada masa giling dengan data dari tahun 2010 – 2015.
3. Jumlah parameter yang digunakan ada 5 diantaranya sebagai berikut:
   a. jumlah lama giling
   b. rendemen tebu
   c. luas areal tanaman tebu
   d. jumlah pekerja
   e. jumlah bahan baku/tebu yang digunakan
4. Perhitungan tingkat kesalahan pada penelitian ini menggunakan *Mean Absolute Precentage Error* (MAPE).

1.6 Sistematika pembahasan

Sistematika pembahasan/laporan ini dibuat untuk memberikan gambaran secara umum dari pembahasan yang akan dibangun dari penelitian ini, adapun sistematika dari penelitian ini adalah :

BAB I PENDAHULUAN

Pada bab ini berisi latar belakang, rumusan masalah, tujuan penelitian manfaat penelitian, batasan penelitian, sistematika pembahasan dan jadwal penelitian dan menjelaskan penelitian secara umum.

BAB II DASAR TEORI

Pada bab ini berisi teori – teori pendukung yang digunakan untuk mendukung penelitian. Dasar teori yang diperlukan untuk mendukung penelitian ini adalah yang pertama definisi gula itu sendiri yang berisi tentang penjelasan mengenai permintaan gula oleh Industri, permintaan gula oleh Rumah Tangga dan faktor – faktor yang mempengaruhi, lalu ada pengertian Metode ELM dan Metode PSO.

BAB III METODOLOGI

Pada bab ini berisi langkah – langkah yang akan dilakukan dalam penelitian ini, yang meliputi: studi literatur, pengumpulan data, pengolahan data, pengujian dan analisis hasil, dan kesimpulan.

BAB IV PERANCANGAN

Pada bab ini berisi perancangan sistem, yang terdiri dari perancangan antarmuka, perancangan pengujian dan evaluasi akhir.

BAB V IMPLEMENTASI

Pada bab ini menyajikan implementasi dari sistem dengan metode hybrid algorithm yang sudah di rancang pada bab sebelumnya, yakni bab perancangan.

BAB VI PENGUJIAN DAN ANALISIS

Pada bab ini menjelaskan analisis sistem yang sudah dikembangkan berdasarkan hasil optimasi.

BAB VII PENUTUP

Pada bab ini berisi kesimpulan hasil yang didapat berdasarkan pengujian dan berisi saran terhadap sistem yang dibangun agar dapat dikembangkan lagi pada penelitian selanjutnya.

BAB 2 LANDASAN KEPUSTAKAAN

Pada bab ini berisi landasan kepustakaan, yang berisi tentang dasar teori – teori yang mendukung pada penelitian ini. Teori yang mendukung diantaranya tentang produksi gula kristal putih dan dasar teori tentang algoritma *Extreme Learning Machine* (ELM) dan *Particle Swarm Optimization* (PSO) sebagai metode yang digunakan dalam penelitian kali ini.

1.1 Kajian Pustaka

Berdasarkan penelitian yang akan dibangun mengenai optimasi produksi gula kristal putih dengan menggunakan metode *hybrid algorithm* yaitu metode ELM - PSO, penulis menggunakan beberapa penelitian yang sudah ada sebelumnya untuk mendukung penelitian ini. Berikut beberapa penelitian yang telah menggunakan metode ELM sebagai peramalan maupun menggunakan metode *hybrid algorithm* ELM - PSO sebagai optimasi. Kajian pustaka tersebut akan dibahas pada Tabel 2.1 .

Tabel 2. 1 Kajian Pustaka

No	Judul	*Input* (Obyek)	Metode (Proses)	*Output* (Hasil)
1	Peramalan Produksi Gula Pasir Menggunakan *Extreme Learning Machine (ELM)* pada PG Candi Baru Sidoarjo (Siwi, et al., 2016)	Data jumlah produksi gula pada Pabrik Gula Candi Baru Sidoarjo selama 5 tahun dari tahun 2010-2015	*Extreme Learning Machine* (ELm)	Hasil dari penelitian ini berupa nilai peramalan produksi gula dan nilai error terendah yang diukur dengan MAPE sebesar 0.74%.
2	Penerpan Metode *Extreme Learning Machine* untuk Peramalan Permintaan (Agustina, et al., 2010)	Data produksi kaos dan pin selama 2 tahun dari tahun 2008-2009 di toko “Cak Cuk Shop”	*Extreme Learning Machine* (ELM)	ELM menghasilkan *output* dengan tingkat kesalahan yang rendah yaitu MAPE 0.0042% pada produk kaos, 0.0095% pada produk pin. *Training time* yang dibutuhkan rata-rata

				0.0059 detik
3	*Classification Of ECG Signals Using Hybrid Particle Swarm Optimization In Extreme Learning Machine* (Karpagachelvi, 2014)	Sinyal ECG	*Hybrid algorithm* ELM – PSO	Pengklasifikasian sinyal ECG dengan menggunakan metode *hybrid algorithmI* ELM – PSO lebih unggul dan akurat
4	Klasifikasi Keminatan Menggunakan Algoritme Extreme Learning Machine dan Particle Swarm Optimization untuk Seleksi Fitur (studi Kasus: Program Studi Teknik Informatika FILKOM UB) (Sugianto, et al., 2017)	Nilai Akademik Mahasiswa Teknik Informatika dari semester 1-5 dan jenis kelamin	*Extreme Learning Machine* (ELM) dan *Particle Swarm Optimization* (PSO)	Penelitian ini menghasilkan hasil klasifikasi dalam pemilihan keminatan yang akan diambil dengan tingkat akurasi tinggi sebesar 94.44%
5	*An Improved Extreme Learning Machine with Adaptive Growth of Hidden Nodes based on Particle Swarm Optimization* (Zhao, et al., 2014)	*Seven Benchmark Datasets, Delta ailerons, Delta elevators, Machine CPU* (diambil dari UCI *Machine*	*Extreme Learning Machine with Adaptive Growth of Hidden Nodes, Particle Swarm Optimization*	Menggunakan metode PSO-AG-ELM yang memiliki nilai deviasi *Root Mean Square Error* (RMSE) lebih kecil daripada metode AG-ELM biasa, hal ini menunjukkan bahwa PSO-AG-ELM lebih stabil.

		Learniing Repository)		

Beberapa penelitian sebelumnya memilki metode yang sama dengan penelitian yang peneliti kerjakan , peneliti menggunakan penelitian diatas guna mendukung penelitian skripsi yang akan dilakukan penulis. Pada penelitian pertama, yang sudah dilakukan oleh Irwin Dwi Agustina (2010) dalam penelitian ini, peneliti menggunakan metode baru dari JST yaitu ELM untuk membandingkan tingkat akurasi metode ini dengan metode konvensional lain seperti metode Moving Average dan Exponential. Hasil yang di dapatkan pada penelitian ini adalah, tingkat akurasi ELM menghasilkan tingkat kesalahan yang rendah yaitu MAPE 0.0042% pada produk kaos dan 0.0095% pada produk pin. Sedangkan pada metode peramalan *Moving Average* tingkat kesalahan 19.19% pada produk kaos dan 54.43% pada produk pin. Lalu pada metode peramalan yang terakhir yaitu metode Exponential Smoothing mempunyai tingkat kesalahan 32.93% pada produk kaos dan 111.39% pada produk pin.

Lalu pada penelitian yang lain yang dilakukan oleh Iga Permata Siwi (2016) yang juga menggunakan algoritma ELM pada penelitian ini tingkat error yang juga dihitung dengan nilai MAPE dengan nilai sebesar 0,74 %. Pada penelitian ini peneliti juga menyimpulkan bahwa metode Extreme Learning Machine (ELM) dapat digunakan dalam sistem yang menghasilkan nilai erorr kecil dengan memasukkan jumlah hidden layer dan random input weight. Selain itu ternyata perbedaan input weight juga berpengaruh terhadap nilai MAPE yang dihasilkan. Semakin lebar nilai interval range yang digunakan maka bukan tidak mungkin untuk mendapatkan nilai input weight terbaik berdasarkan hasil random dan tidak lupa perbandingan jumlah data training dan data testing serta penambahan hidden layer berpengaruh terhadap output peramalan yang dihasilkan.

Pada penelitian yang lain, yang dilakukan oleh S. Karpagachelvi (2014) pada penelitian ini peneliti membahas pengklasifikasian sinyal ECG dengan menggunakan algoritma *hybrid*, yaitu algoritma ELM dan PSO. Dalam penelitian ini hasil yang di peroleh peneliti dengan menggabungkan kedua algoritma tersebut ELM dan PSO memiliki nilai yang lebih unggul dan akurat jika dibandingkan dengan pengklasifikasian sinyal ECG secara tradisional. Dalam penelitian ini peneliti juga menyimpulkan bahwa metode hybrid ELM - PSO ini membuat metode ELM mengalami peningkatan performa dalam dua aspek yaitu dalam aspek seleksi fitur dan parameter optimasi. Metode PSO sendiri digunakan untuk mengoptimasi seleksi fitur dan parameter kernel ELM.

Penelitian lain yang juga menggunakan dua algoritma adalah penelitian yang dilakukan oleh Nur Afifah Sugianto (2017). Pada penelitian ini peneliti menggunakan metode ELM sebagai sebagai metode klasifikasi untuk pemilihan keminatan pada mahasiswa Informatika. Akan tetapi pada metode ELM tidak memiliki kemampuan untuk melakukan seleksi fitur sehingga dikombinasikan metode ELM dengan metode

PSO sebagai metode untuk menyeleksi fitur yang digunakan agar dapat menyeleksi fitur dengan otomatis dan optimal. Pada penelitian ini menghasilkan nilai akurasi sebesar 94.44% dibandingkan dengan hanya menggunakan metode ELM biasa yakni menghasilkan nilai sebesar 66,67%.

Pada penelitian yang lain, penelitian yang dilakukan oleh Zhao, Zhang dan Han (2014). Pada penelitian ini peneliti menggunakan metode ELM lalu mengembangkannya dengan *Adaptive Grwoth of Hidden Nodes* (AG) dan menggabungkannya dengan *Particle Swarm Optimization* (PSO). Pada penelitian ini peneliti menggunakan metode PSO guna menghitung bobot dan bias yang paling optimal yang selanjutnya nilai tersebut akan digunakan pada metode AG-ELM. Pada penelitian ini peneliti menyimpulkan bahwa pengembangan PSO-AG-ELM ini memiliki tingkat akurasi yang tinggi jika dibandingankan dengan AG-ELM dan ELM biasanya dan menunjukkan bahwa PSO-AG-ELM lebih stabil dan juga memiliki tingkat akurasi yang lebih baik.

Lalu pada penelitian yang lain yang dilakukan oleh Iga Permata Siwi (2016) yang juga menggunakan algoritma ELM, pada penelitian ini tingkat *error* yang juga dihitung dengan nilai MAPE dengan nilai sebesar 0,74 %. Padap penelitian ini peneliti menggunakan variabel gabungan dimana variabel pada data ke-4 akan digunakan pada variabel data pertama sehingga variabel yang digunakan berjumlah dua kali lipat. Pada penelitian ini peneliti juga menyimpulkan bahwa metode *Extreme Learning Machine* (ELM) dapat digunakan dalam sistem yang menghasilkan nilai *error* kecil dengan memasukkan jumlah *hidden layer* dan *random input weight*. Selain itu ternyata perbedaan *input weight* juga berpengaruh terhadap nilai MAPE yang dihasilkan. Semakin lebar nilai *interval range* yang digunakan maka bukan tidak mungkin untuk mendapatkan nilai *input weight* terbaik berdasarkan hasil random dan tidak lupa perbandingan jumlah data *training* dan data *testing* serta penambahan *hidden layer* berpengaruh terhadap *output* peramalan yang dihasilkan.

Pada penelitian kali ini, peneliti melakukan penelitian tentang optimasi dari *input weight* dan bias pada algoritma ELM yang tidak dilakukan oleh penelitian sebelumnya. Pada penelitian sebelumnya peneliti hanya menggunakan metode ELM saja untuk memprediksi suatu produksi dan menggunakan variabel gabungan. Untuk *input weight* dan bias yang digunakan didapat secara *random* atau acak. Sedangkan pada penelitian kali ini *input weight* dan bias didapat dari hasil optimasi oleh metode PSO. Variabel yang digunakan pada penelitian ini berjumlah 5 variabel dan tanpa variabel gabungan, yakni jumlah lama giling, rendemen tebu, luas areal tanaman tebu, jumlah pekerja, dna jumlah bahan baku.

1.2 Gula

1.2.1 Pengertian Gula

Gula adalah suatu karbohidrat sederhana yang menjadi sumber energi dan komoditi perdagangan utama. Gula digunakan untuk mengubah rasa menjadi manis dalam keadaan makanan ataupun minuman. Gula sederhana seperti glukosa menyimpan energi yang akan digunakan oleh sel. Gula yang paling banyak diperdagangkan adalah gula dalam bentuk kristal sukrosa padat. Gula diperoleh dari nira tebu, bit gula atau aren. Meski demikian, terdapat sumber – sumber gula lainnya, seperti kelapa, umbi dahlia, anggur, ataupun bulir jagung. Sedangkan proses untuk menghasilkan gula mencakup tahap ekstrasi (pemerasan) dan pemurnian melalui distalasi (penyulingan)(Siwi, et.al., 2016).

1.2.2 Permintaan Gula oleh Industri

Seiring dengan pesatnya pertumbuhan industri makanan dan minuman, penggunaan gula pasir oleh industri juga mengalami peningkatan lebih cepat jika dibandingkan dengan konsumsi langsung oleh rumah tangga. Permintaan gula pasir oleh kelompok industri pangan skala sedang dan besar di Indonesia adalah konsumen terbesar berbentuk berbagai bahan pemanis sehingga kelompok industri ini dapat mepresentasikan dinamika permintaan seluruh kelompok industri. Pada Tabel 2.2 berikut dapat menunjukkan beberapa jenis industri yang permintaannya dominan terhadap konsumsi gula. Selama periode 1980 – 1994 permintaan terhadap gula cenderung mengalami peningkatan (Sugiyanto, 2007).

Tabel 2. 2 Permintaan Gula Pasir oleh Industri Sedang dan Besar

Jenis Industri / Tahun	Jml. Perusahaan (Unit)	Total (Ton)	Rata – rata (Ton)
Susu			
- 1980	6	49.962	8.327
- 1985	19	38.142	2.007
- 1990	23	54.735	2.380
- 1994	19	90.337	4.755
Minuman Kemasan			
- 1980	94	14.784	157
- 1985	126	28.119	223
- 1990	163	45.858	281
- 1994	231	77.919	337

Makanan Olahan			
- 1980	542	20.645	38
- 1985	1.983	33.580	17
- 1990	1.390	62.318	45
- 1994	2.529	133.630	53

(Sugiyanto, 2007).

1.2.3 Permintaan Gula oleh Rumah Tangga

Survei ICBS (1997) menunjukkan hal yang sama dengan studi yang ada bahwa presentase konsumsi gula secara langsung mengalami penurunan dari 80% total kebutuhan pada tahun 1984 menjadi 60,2% gula nasional pada tahun 1990. Penurunan angka tersebut mencerminkan 3 hal. Pertama, pergeseran pola konsumsi gula di dalam rumah tangga menjadi konsumsi di luar rumah tangga sebagai akibat dari meningkatnya jumlah rumah makan *fast food*. Kedua, peningkatan konsumsi penduduk atas produk makanan atau minuman olahan yang mengandung gula sehingga pengguna gula sebagai bahan baku dalam industri meningkat. Ketiga, tingkat penghasilan masyarakat.

Selain itu, pola konsumsi gula untuk rumah tangga di kota dan di desa memiliki pola konsumsi yang berbeda. Konsumsi gula untuk rumah tangga di kota jauh lebih tinggi daripada konsumsi gula di desa. Seperti ditunjukkan pada Tabel 2.3 , dalam periode 1984-2003 konsumsi langsung gula di daerah perkotaan dan perdesaan cenderung meningkat dengan laju kenaikan 0,24% dan 1,97% per tahun atau secara nasional meningkat 1,51% per tahun (Sugiyanto, 2007).

Tabel 2. 3 Permintaan Tebu Desa dan Kota

Tahun	Pedesaan	Perkotaan	Jumlah
1987	7,03	9,26	7,62
1990	7,36	9,06	7,88
1993	7,57	9,29	8,14
196	8,47	9,41	9,04
1996	8,88	9,26	9,04
Growth per tahun (%)	1,92	0,24	1,51

(Sugiyanto, 2007)

1.2.4 Faktor-Faktor Yang Mempengaruhi Produksi Gula

Dalam memproduksi gula diperlukan beberapa faktor pendukung agar jumlah produksi memenuhi permintaan pasar. Berikut beberapa faktor yang mempengaruhi jumlah produksi gula, khususnya di Indonesia:

1. Tebu

Tebu (Saccharum Officinarum) termasuk kedala satu keluarga dengan rumput – rumputan. Mulai dari pangkal hingga ujung batang mengandung air gula dengan kadar 20%. Air gula inilah yang nantinya akan dibuat kristal – kristal gula putih. Disamping untuk pembuatan gula pasir, tebu juga digunakan sebagai bahan baku utama pembuatan gula merah (Napitupulu, et al., 2013).

2. Luas Areal (Tanah)

Lahan sebagai sarana produksi merupakan bagian dari faktor produksi. Dalam suatu usaha tani, pemilikan atau penguasaan lahan sempit sudah pasti kuran efisien dibanding lahan yang lebih luas. Luas areal atau lahan tebu mampu mempengaruhi jumlah produksi gula. Semakin luas lahan yang ditanami tebu, maka semakin banyak jumlah tebu yang akan diproduksi. Jumlah produksi tebu yang dihasilkan dalam jumlah besar dan memiliki kualitas tebu yang baik akan mampu menghasilkan jumlah produksi gula yng tinggi dan juga berkkualitas. Jenis dan tipe lahan yang akan ditanami tebu harus sesuai dengan jenis varietas tanaman tebu agar menghasilkan gula dalam jumlah yang esar dan berkualitas tinggi pula (Siwi, et al., 2016).

3. Rendemen tebu

Rendemen tebu merupakan kandungan yang terdapat ada tebu. Dalam prosesnya rendemen dihasilkan oleh tanaman yang dipengaruhi oleh keadaan tanaman dan proses penggilingan tanaman di pabrik. Sedangkan untuk mendapatkan rendemen tebu yang tinggi, tanaman harus bermutu baik dan ditebang pada saat yang tepat. Namun sebaik apapun mutu dari tebu, apabila pabrik dalam mengolah tebu tidak baik, maka hablur yang didapat akan berbeda dengan kandungan sukrosa yang ada pada batang tebu.

Pengertian rendemen yang lain adalah dapat diartikan sebagai kadar kandungan gula di dalam batang tebu yang dinyatakan dalam hitungan persen. Contohnya apabila dikatakan rendemen tebu 10% yang memiliki arti bahwa dari 100 kg tebu yang digiling di pabrik , akan diperoleh gula sebanyak 10 kg. Hubungan rendemen terhadap jumlah produksi gula sangat mempengaruhi. Disebabkan karena rendemen sendiri merupakan kandungan gula dalam batang tebu yang nantinya akan mempengaruhi jumlah produksi gula yang dihasilkan. Rendemen sendiri sangat dipengaruhi oleh bagaimana proses penggilingan tebu, penebangan, dan tentunya varietas tebu yang ditanam. Jika proses penggilingan =, penebangan dan penanaman varietas tebu yang sesuai dan bekualitas, maka rendemen yang dihasilkan pun akan tinggi jumlahnya sehingga gula yang akan diproduksi akan lebih banyak dan lebih berkualitas (Napitupulu, et al., 2013).

4. Tenaga Kerja Manusia

Tenaga kerja manusia merupakan faktor penting untuk memproduksi gula. Para tenaga kerja membantu proses pembuatan gula yang tidak bisa dilakukan oleh mesin. Tenaga kerja yang ada pada Pabrik Gula Candi Baru dibedakan atas tiga bagian yaitu, *staff, non-staff,* dan pekerja PKWT. Pekerja staff dan non-staff adalah pekerja tetap. Sedangkan yang dimaksud dengan pekerja PKWT adalah pekerja yang bekerja secara musiman atau dengan sistem kontrak, yang artinya dia hanya bekerja pada saat musim giling tebu saja (Siwi, et al., 2016).

5. Modal Biaya Produksi

   Biaya produksi adalah biaya – biaya yang terjadi untuk mengolah bahan baku menjadi produk jadi yang siap untuk dijual (Mulyadi, 2004). Unsur biaya dalam harga pokok produksi diklasifikasikan atas 3 biaya , yaitu:

   1. Biaya bahan baku (*Material Cost*)
   2. Biaya tenaga kerja (*Labour Cost*)
   3. Biaya *overhead* pabrik (*Factory Overhead Cost*)

6. Jam mesin

   Mesin merupakan salah satu faktor penting dalam memproduksi gula. Berdasarkan sifat produksi gula yang bersifat kontinyu, apabila terjadi kerusakan atau kemacetan pada mesin, dapat mengakibatkan berhentinya produksi gula secara keseluruhan sehingga lama jam mesin dapat beroperasi akan sangat mempengaruhi hasil yang didapatkan dari kegiatan produksi gula tersebut (Siwi, et al., 2016).

7. Lama giling

   Lama giling adalah lama waktu yang dibutuhkan dalam mengolah atau menggiling tebu menjadi gula dalam satu musim giling. Lama giling juga sangat berkaitan dengan jam mesin. Apabila mesin mengalami kerusakan, maka akan berpengaruh terhadap lama giling. Proses penggilingan yang efektif dapat ditentukan dari tersedianya bahan baku tebu dan mesin giling dalam memproduksi gula (Siwi, et al., 2016).

1.3 Peramalan

Peramalan adalah metode untuk memperkirakan suatu nilai dimasa depan dengan menggunakan data masa lalu. Peramalan juga dapat diartikan sebagai seni dan ilmu memperkirakan kejadian pada masa yang akan dating, sedangkan aktivitas peramalan merupakan suatu fungsi bisnis yang berusaha memperkirakan penjualan dan penggunaan suatu produk sehingga produk-produk tersebut dapat dibuat dalam kuantitas yang tepat (Gaspersz. 2002).

Peramalan yang dibuat selalu diupayakan agar dapat (Subagyo, 1986):

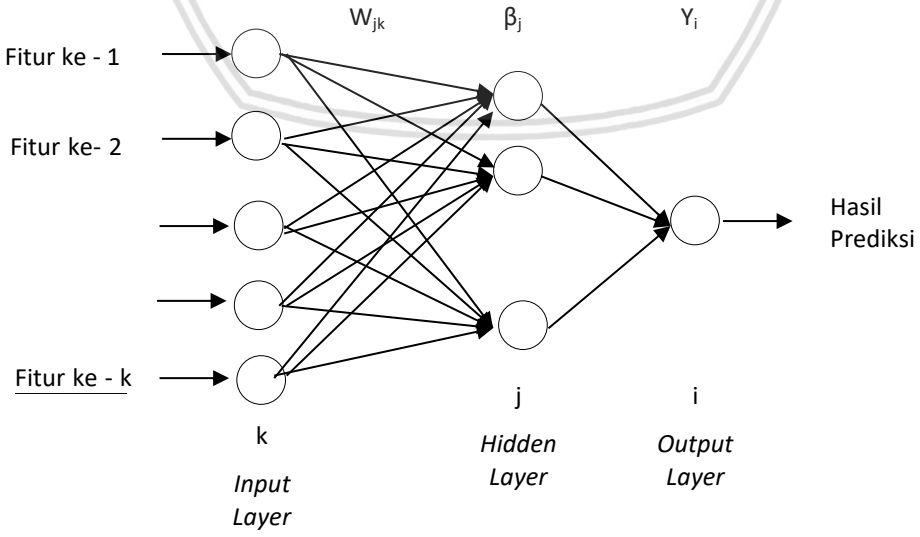
1. Meminimumkan pengaruh ketidakpastian terhadap perusahaan
2. Peramalan bertujuan mendapatkan peramalan (*forceast*) yang bisa meminimumkan kesalahan meramal (*forecast error*) yang biasanya diukur dengan MSE (*Mean Absolute Error*), MAE (*Mean Absolute Error*) dan sebagainya.

1.4 Extreme Learning Machine

Extreme Learning Machine merupakan metode pembelajaran baru dari jaringan saraf tiruan. Metode ini pertama kali dikenalkan oleh Huang (2004). Metode ELM sendiri merupakan jaringan saraf tiruan *feedforward* dengan *single hidden layer* atau dengan kata lain *Single Hidden Layer Feedforward Neural Networks (SLFNs)*. Huang *et al* (2004) mengemukakan dua alasan mengapa JST *feedforward* lain mempunyai learning speed rendah, yaitu:

1. Menggunakan *slow gradient based learning algorithm* untuk melakukan *training*.
2. Semua parameter pada jaringan ditentukan secara *iterative* dengan menggunakan metode pembelajaran tersebut.

Pada ELM parameter–parameter seperti input weight dan hidden bias dipilih secara random, sehingga ELM memiliki *learning speed* yang cepat dan mampu menghasilkan generalisasi dengan performa yang baik. Berikut adalah struktur dari metode ELM pada data (x_i , y_i) (Cholissodin, et al., 2017).



Gambar 2. 1 Arsitektur ELM

1.4.1 Cara Kerja Algoritma ELM

Dalam perhitungannya ELM sendiri dibedakan menjadi 2 proses, proses yang pertama adalah proses *training* dan proses yang kedua adalah proses *testing*.

1.4.1.1 Proses Pelatihan

Pada proses training ELM yang pertama kali dilakukan adalah menginputkan nilai input weight dan bias secara random, nilai tersebut didapat dari hasil perhitungan PSO. Langkah – langkah *process training* ELM adalah sebagai berikut (Cholissodin, 2017):

1. Inisialisasi *Input weight* (W_{jk}) da Bias (b) secara acak.
2. Menghitung Matriks Keluaran *Hidden Layer*

   Menghitung matriks keluaran dari *hidden layer* (H) dengan fungsi aktivasi *sigmoid biner*, fungsi aktivasi yang sering digunakan dalam jaringan syaraf tiruan untuk memetakan nilai matriks keluaran dari *hidden layer* pada interval 0 sampai 1. Persamaan *sigmoid biner* dapat dituliskan pada persamaan 2.1 berikut:

$$H = \frac{1}{(\,1 + \exp\,(-(X.W^T + ones(Ntrain, 1) * b)))} \qquad (2.1)$$

   Keterangan :

   H = Keluaran Hidden Layer
   exp = Eksponensial
   X = Matriks Data Masukan *(Input)*
   W^T = Matriks *Transpose Input Weight*
   Ones(Ntrain, 1) = Matriks yang bernilai 1 dengan ukuran baris sama dengan jumlah *training* dan 1 kolom
   b = Bias

3. Menghitung Matriks *Moore-Penrose Pseudo Inverse*

   Menghitung matriks *Moore-Penrose Inverse* yang didapatkan dari perhitungan perkalian matriks *inverse* dan *trasnpose* keluaran *hidden layer*. Persamaan Matriks *Moore-Penrose Pseudo Inverse* dapat dituliskan pada persamaan 2.2 berikut:

$$H^+ = (H^T * H)^{-1} * H^T \qquad (2.2)$$

   Keterangan :

   H^+ = matriks *Moore – Penrose Pseudo Inverse*
   H = matriks keluaran *hidden layer*

H^T = transpose matriks keluaran *hidden layer*

4. Menghitung *Output weight* ($\hat{\beta}$)

   Menghitung *Ouput weight* yang dihasilkan oleh *hiden layer* dan *output layer*. Persamaan dapat dituliskan seperti persamaan 2.3 berikut ini:

$$(\hat{\beta}) = H^+ * Y \quad (2.3)$$

   Keterangan :

   $\hat{\beta}$ = bobot keluaran *hidden layer*
   H^+ = matriks *Moore – Penrose Pseudo Inverse*
   Y = matriks data *output* atau target

5. Menghitung hasil presdiksi $(\hat{Y})$

   Menghitung hasil prediksi didapatkan dari proses perkalian antara matriks keluaran *hidden layer* dengan bobot keluaran *hidden layer* atau *Output weight hidden layer*. Persamaan dapat dituliskan seperti persamaan 2.4 berikut:

$$(\hat{Y}) = H * \hat{\beta} \quad (2.4)$$

   Keterangan :

   $\hat{Y}$ = hasil prediksi
   H = matriks keluaran *hidden layer*
   $\hat{\beta}$ = matriks bobot keluaran *hidden layer* / *Output weight hidden layer*

1.4.1.2 Proses Pengujian

Langkah–langkah proses *testing* tidak jauh berbeda dengan langkah-langkah pada proses *training*. Hanya saja pada proses *testing* tidak perlu menghitung *input weight* dan *output weight*. *Input weight* dan *output weight* pada proses *testing* menggunakan bobot yang telah dihitung pada proses *training.* Proses *training* pada metode ELM dugunakan untuk membuat pola model ELM sedangkan pada proses *testing* metode ELM digunakan untuk melakukan evaluasi kemampuan ELM itu sendiri sebagai *forecasting tool* (Siwi, et al., 2016). Langkah-langkah pada proses *testing* adalah sebagai berikut:

1. Menggunakan *input weight* (W_{jk}) dan *output weight* ($\hat{\beta}$) yang didapatkan pada proses *training*
2. Menghitung matriks keluaran *hidden layer* (H) menggunakan Persamaan 2.1 seperti pada proses training.

3. Menghitung hasil prediksi ($\hat{Y}$) menggunakan Persamaan 2.4 seperti pada proses training.
4. Menghitung nilai evaluasi menggunakan MAPE.

1.5 Optimasi

Optimasi adalah suatu pendekatan normatif untuk mengidentifikasikan penyelesaian terbaik dalam pengambilan keputusan dari suatu permasalahan. Penyelesaian permasalahan dalam teknik optimasi diarahkan untuk mendapatkan titik maksimum atau titik minimimum dari fungsi yang dioptimumkan. Tujuan dari optimasi adalah untuk meminimumkan usaha yang diperlukan atau biaya operasional dan memaksimumkan hasil yang diinginkan. Jika usaha yang diperlukan atau hasil yang diharapkan dapat dinyatakan sebagai fungsi dari peubah keputusan, maka optimasi dapat didefinisikan sebagai proses pencapaian kondisi maksimum dan minimum dari fungsi tersebut. Unsur penting dalam masalah optimasi adalah fungsi tujuan, yang sangat bergantung pada sejumlah peubah masukan. Peubah-peubah ini dapat tidak saling bergantung atau saling bergantung melalui satu atau lebih kendala (Witary, et al., 2013).

1.6 Particle Swarm Optimization

Algoritma PSO pertama kali diusulkan oleh J. Kennedy dan R. C. Eberhart (Kennedy, 1995). Particle Swarm Optimization (PSO) merupakan salah satu metode *Particle Swarm Optimization* (PSO), merupakan sebuah metode optimasi yang didasarkan pada perilaku sebuah kawanan serangga misalnya semut, rayap, lebah atau burung. Suatu partikel dalam ruang memiliki posisi dan setiap posisi dalam ruang pencarian merupakan alternatif solusi yang dapat dievaluasi menggunakan fungsi objektif. Setiap partikel dapat menyesuaikan posisi dan kecepatan masing-masing dengan cara setiap partikel menyampaikan informasi terbaiknya kepada partikel yang lain. Oleh karena itu, setiap partikel memiliki kecenderungan untuk terbang menuju posisi yang dianggap terbaik (Handika, et al., 2016).

1.6.1 Cara Kerja Algoritma PSO

Langkah – langkah pada *Particle Swarm Optimization* (PSO) dapat diuraikan seperti berikut (Cholissodin, et al., 2016):

1. Proses Inisialisasi

a. Inisialisasi Kecepatan Awal Partikel

Pada iterasi awal atau iterasi ke-0, nilai kecepatan awal semua partikel di atur dengan memberi nilai 0.

b. Inisialisasi Posisi Awal Partikel

Pada iterasi awal atau iterasi ke-0, posisi awal partikel dibangkitkan secara acak atau *random.*

c. Menghitung Nilai *Fitness*

d. Inisialisasi Nilai *Pbest* dan *Gbest*

2. *Update* kecepatan Partikel

Untuk melakukan *update* kecepatan partikel, digunakan persamaan seperti persamaan 2.5 berikut ini:

$$v_{i,j}^{t+1} = w.v_{i,j}^{t} + c_1.r_1\left(Pbest_{i,j}^{t} - x_{i,j}^{t}\right) + c_2.r_2\left(Gbest_{g,j}^{t} - x_{i,j}^{t}\right) \quad (2.5)$$

Keterangan:

$v_{i,j}^{t+1}$ = kecepatan partikel i dimensi j pada iterasi

$v_{i,j}^{t}$ = kecepatan partikel i dimensi j pada iterasi t (t sebelumnya)

w = bobot inersia

$c1$ = konstanta kecepatan 1

$c2$ = konstanta kecepatan 2

$r1, r2$ = nilai acak $\in [0,1]$

$Pbest_{i,j}^{t}$ = posisi terbaik dari partikel i dimensi j pada iterasi t

$Gbest_{g,j}^{t}$ = global optimal dari partikel g dimensi j pada iterasi t

$x_{i,j}^{t}$ = posisi partikel i dimensi j pada iterasi t

3. *Update* Posisi Partikel

- Menentukan *update* posisi dengan menggunakan persamaan 2.7 seperti persamaan berikut ini pada PSO *realcode*:

$$x_{i,j}^{t+1} = x_{i.j}^{t} + v_{i.j}^{t+1} \quad (2.7)$$

Keterangan:

$x_{i.j}^{t}$ = nilai posisi awal

$v_{i.j}^{t+1}$ = nilai update kecepatan

4. Update Nilai *Pbest* dan *Gbest*

   Untuk mendapat nilai *Pbest* dapat dilakukan dengan cara membandingkan antara nilai *Pbest* pada iterasi sebelumnya dengan hasil dari perhitungan *update* posisi. Nilai *fitness* yang lebih rendah dari keduanya akan menjadi nilai *Pbest* yang baru. Sedangkan untuk mendapatkan nilai *Gbest* terbaru, dapat dilakukan perhitungan dengan cara melihat nilai *Pbest* yang memiliki nilai *fitness* yang paling rendah. Nilai *fitness* yang digunakan dalam penelitian ini didapat dari nilai MAPE pada perhitungan ELM sehingga dipilihlah nilai *fitness* terendah.

5. Kondisi berhenti

   Mengulang langka ke 2-4 sampai kondisi berhenti terpenuhi (*Termination condition*) terpenuhi. Ada beberapa kondisi berhenti yang digunakan, diantaranya adalah:

a. Ketika iterasi sudah mencapai *Maximum*
b. Iterasi berhenti tetapi tidak ada perubahan yang signifikan
c. Ketika telah mencapai waktu *Maximum*

1.7 Extreme Learning Machine (ELM) & *Particle Swarm Optimization* (PSO)

Extreme Learning Machine merupakan metode yang paling cocok digunakan untuk sistem peramalan. Akan tetapi karena dalam perhitungan awal dari ELM nilai input weight dan bias dipilih secara random, maka tidak menutup kemungkinan hasil yang didapat dari perhitungan ELM kurang efektif. Oleh karena itu, untuk lebih mengoptimalkan nilai input weight dan bias diperlukan metode lain guna menghasilkan nilai peramalan ELM yang lebih tepat.

Untuk menyelesaikan permasalahan diatas, metode optimasi yang dapat digunakan salah satunya adalah metode *Particle Swarm optimization* (PSO). Pada umumnya *hybrid algorithm* PSO-ELM sering kali digunakan untuk 2 tujuan, yaitu seleksi fitur dan mendapatkan bias terbaik (Sugianto, et al., 2017). Akan tetapi pada penelitian kali ini, metode PSO yang peneliti gunakan adalah unutk mendapakan nilai input weight dan bias yang optimal sehingga dapat menghasilkan peramalan yang lebih optimal juga.

Langkah – langkah pengerjaan ELM dengan menggunakan optimasi PSO secara umum sperti berikut:

1. Menentukan panjang dimensi *particle* dan banyaknya iterasi maksimum yang akan digunakan. Panjang particle didapat dari fitur dikalikan jumlah *hidden neuron* dan ditambah bias.
2. Inisialisasi kecepatan *particle*. Seperti pada langkah – langkah PSO pada proses inisialisasi diatas, bahwa pada iterasi ke-0, kecepatan *particle* bernilai 0.

3. Inisialisasi posisi *particle* secara acak dengan panjang dimensi *particle* seperti yang sudah dijelaskan pada langkah 1 diatas.
4. Melakukan proses *training* pada ELM dan melakukan proses *testing* seperti yang sudah dipaparkan dengan nilai input weight yang didapat dari PSO untuk mendapat nilai MAPE yang nantinya digunakan sebagai nilai *fitness* pada posisi partikel, *update* Pbest dan Gbest.
5. Melakukan langkah-langkah pengerjaan PSO seperti yang sudah dipaparkan diatas hingga kondisi berhenti dan mendapatkan nilai *Gbest* dengan *fitness* yang paling rendah.

1.8 Normalisasi Data

Dalam melakukan proses perhitungan dalam peramalan, perlu dilakukan proses *preprocessing* terlebih dahulu yang berupa normalisasi data. Normalisasi data bertujuan untuk standarisasi data yang akan digunakan dalam perhitungan sehingga data berada pada jarak tertentu. Proses normalisasi sendiri diperlukan untuk menghasilkan nilai output dengan *range* data [0,1] atau [-1,1]. Persamaan yang digunakan untuk melakukan normalisasi adalah seperti persamaan 2.8 berikut ini:

$$x' = \frac{x - min}{max - min} \qquad (2.8)$$

Keterangan:

x' = Nilai data setelah normalisasi

x = Nilai data sebelum normalisasi

max = Nilai maksimum dari pada data set

min = Nilai minimum dari pada data set

1.9 Denormalisasi Data

Denormalisasi data adalah proses dimana untuk mengembalikan nilai data menjadi nilai yang sebenarnya berdasarkan hasil peramalan. Persamaan denormalisasi data dapat dituliskan seperti persamaan 2.9 berikut:

$$x = (x' \times (\max - \min)) + min \qquad (2.9)$$

Keterangan:

x' = Nilai data setelah normalisasi

x = Nilai data sebelum normalisasi

max = Nilai maksimum dari pada data set

min = Nilai minimum dari pada data set

1.10 Nilai Evaluasi

Nilai evaluasi dilakukan dengan mencari nilai *error Mean Absolute Precentage Error* (MAPE) dengan membandingkan selisih antara nilai ramalan dengan nilai aktual. Persamaan untuk nilai MAPE dapat ditulis seperti berikut ini:

$$\text{MAPE} = \frac{1}{n}\sum_{i=1}^{n}\left|\left(\frac{\text{y}_i - \hat{\text{y}}_i}{\text{y}_i}\ x\ 100\right)\right| \qquad (2.10)$$

Keterangan:

n = jumlah data

$\hat{\text{y}}_i$ = nilai hasil ramalan

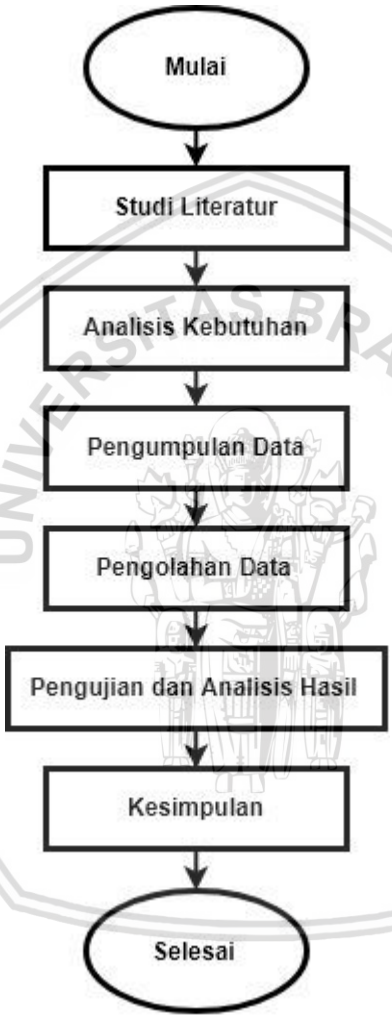
y_i = nilai aktual

Lebih kecil nilai MAPE yang didapat, maka nilai dari prediksi akan semakin baik. Berikut adalah tabel kriteria dari MAPE (Chang, 2007) :

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

BAB 3 METODOLOGI

Dalam penelitian kali ini yang berjudul "Particle Swarm Optimization Untuk Optimasi Bobot Extreme Learning Machine Dalam Memprediksi Produksi Gula Kristal Putih Pabrik Candi Baru-Sidoarjo" berikut adalah gambar diagram alirnya:



Gambar 3. 1 Diagram Alir Metode Penelitian

1.1 Studi Literatur

Studi literatur merupakan dasar teori yang digunakan untuk menunjang suatu penelitian. Studi literatur juga digunkan untuk memahami konsep–konsep yang akan dipakai pada suatu penelitian. Berikut adalah studi literatur yang digunakan pada penelitian ini.

a. Metode *Extreme Learning Machine*
b. Metode *Particle Swarm Optimization*
c. Faktor – faktor produksi gula

1.2 Analisis Kebutuhan

Analisis kebutuhan digunakan untuk menentukan kebutuhan perangkat lunak maupun perangkat keras yang akan di gunakan oleh peneliti untuk proses penelitian.

1.3 Pengumpulan Data

Pengumpulan data pada penelitian kali ini didapatkan dari Pabrik Gula Candi Baru Sidoarjo. Data yang diperoleh merupakan data dari tahun 2010 hingga 2015. Jumlah data sebanyak 45, data yang didapat meliputi bulan penggilingan yang dimulai dari bulan mei sampai desember tiap tahunnya. Selain bulan ada hari giling, yang artinya berapa hari dalam satu bulan itu proses penggilingan terjadi. Lalu ada rendemen, luas areal dalam hektar (ha), jumlah pekerja , tebu yang di giling dalam kwintal (kw) dan jumlah produksi gula.

1.4 Pengolahan Data

Berdasarkan penjelasan diatas, data yang didapat akan di olah sesuai dengan langkah – langkah yang sudah dijelaskan sebelumnya. Langkah-langkah dari proses perhitungan ini dimulai dari perhitungan PSO. Dimana pada proses menentukan panjang *particle* didapat dari perkalian antara jumlah fitur dan *hidden neuron* yang akan di pakai dan jumlah bias. Jumlah dari fitur itu sendiri adalah 5 dan *hiden neuron* yang dipakai adalah 3 dan nilai biasnya didapat dari 1 dikalikan *hidden neuron* sehingga jumlah total panjang *particle* yang akan digunakan adalah 18. Setelah itu mulai proses inisialisai pada PSO dimana ketika mencapai inisialisasi posisi yang membutuhkan nilai fitness, kemudian nilai fitness tersebut didapat dari proses perhitungan ELM dan setelah mencapai tahap testing pada ELM lalu di hitung nilai kesalahannya dengan MAPE dan nilai MAPE itu yang akan menjadi nilai fitness dari masing-masing ukuran populasi. Setelah masing-masing nilai fitness terisi barulah masuk ke tahap selanjutnya pada metode PSO. Setelah inisialisasi Pbest dan Gbest, dimana nilai Gbest adalah nilai yang memiliki fitness terkecil karena nilai fitness merupakan hasil dari MAPE. Setelah mencapai iterasi tertentu barulah dapat di

hasilkan nilai *input weight* , bias dan hasil kesalahan yang paling rendah. Dari 45 total data yang didapat, 15 data digunakan dalam perhitungan manualisasi. Untuk data *training* menggunakan 12 data, sedangkan untuk data *testing* menggunakan 3 data sisanya.

1.5 Pengujian dan Analisis Hasil

Tahap pengujian pada penelitian ini adalah mengujikan sistem yang dibuat terhadap algoritma Particle Swarm Ootimization. Sedangkan untuk tahap analisis kebutuhan bertujuan untuk menganalisis hasil dari pengujian. Menganalisis apakah hasil yang diperoleh sudah dapat menentukan produksi gula kristal putih yang berkualitas dan optimal.

1.6 Penarikan Kesimpulan

Setelah tahap analisis kebutuhan, pengumpulan data, pengolahan data, dan pengujian telah selesai dilakukan, maka tahap selanjutnya adalah pengambilan kesimpulan dan saran. Kesimpulan diperoleh dari hasil pengolahan data terhadap sistem yang dibangun. Setelah ditarik kesimpulan, maka langkah terakhir dari penulisan ini adalah saran. Saran ini berguna untuk memperbaiki kesalahan–kelasalahan yang terjadi baik pada sistem maupun pada penulisan. Selain itu saran juga dapat digunakan untuk menyempurnakan sistem dan sebagai bahan pertimbangan untuk penelitian selanjutnya.

BAB 4 PERANCANGAN

Pada bab ini menjelaskan tentang permasalahan, arsitektur perancangan untuk sistem, diagram alir pada sistem, siklus penyelesaian optimasi *input weight* dan bias yang menggunakan *Particle Swarm Optimization* (PSO) pada prediksi produksi gula dengan metode *Extreme Learning Machine* (ELM).

4.1 Analisis Kebutuhan

4.1.1. Kebutuhan Sistem yang Digunakan

Pada penelitian ini, bahasa pemrograman yang digunakan dalam mengimplementasikan metode ELM dan PSO adalah bahasa pemrograman *Java*. Sedangkan perangkat lunak yang digunakan untuk implementasi adalah Netbeans IDE 8.0.2 dan spesifikasi dari Netbeans IDE 8.0.2 dapat dilihat pada tabel 4.2 berikut.

Tabel 4. 1 Spesifikasi Netbeans 8.0.2

Sistem Operasi	Processor	RAM	Kapasitas Harddisk
Windows 10 Professional	800MHz Intel Pentium III atau setara	512 MB	750 MB

Sumber: netbeans.org

4.1.2. Kebutuhan Data

Pada penelitian kali ini proses pengumpulan data untuk melakukan identifikasi didapatkan dari data Pabrik Gula Candi Baru Sidoarjo. Data yang didapat sejumlah 45 data dan 7 fitur, akan tetapi yang digunakan hanya 5 fitur diantaranya, lama giling, rendemen tebu, jumlah pekerja, berat tebu yang digiling dan luas areal tanaman tebu. Data yang didapat adalah data dari bulan Mei 2010 hingga bulan November 2015. Adapun data yang digunakan pada penelitian ini ditunjukkan pada lampiran A.

4.2 Analisis Kebutuhan

Permasalah yang akan diselesaikan pada penelitian kali ini adalah permasalahan tentang prediksi atau peramalan produksi gula pasir yang bertujuan agar produksi gula yang dihasilkan dapat memenuhi kebutuhan konsumen akan permintaan pada gula itu sendiri. Prediksi gula pada penelitian ini dilakukan dengan metode *Extreme Learning Machine* yang akan di optimasi terlebih dahulu dengan metode *Particle Swarm Optimization* sehingga hasil peramalan pada penelitian ini memiliki tingkat akurasi yang lebih baik dibanding peramalan dengan metode yang konvensional lain. Hasil peramalan yang didapat berdasarakan nilai tingkat kesalahan (*forecast error*)

yang artinya semakin kecil nilai kesalahan yang di dapat maka data peramalan semakin akurat.

Peramalan ini memiliki 5 nilai *input* data antara lain, hari giling, rendemen gula, luas areal, jumlah pekerja, dan jumlah tebu yang digiling (kw) dimana data tersebut dalam format .xls yang bernama data produksi gula. Pada penelitian ini selain terdapat 5 nilai masukan data atau *input weight* yang nantinya akan di produksi dengan 3 *hidden neuron* sehingga diperoleh matrik dengan ordo 3 x 5 = 15 partikel dan nilai bias yang didapat yaitu dari perkalian 1 x *hidden neuron* dengan hasil nilai matriks yang memiliki ordo 1 x 3 = 3 *particle*. Sebelum memulai mengerjakan permasalahan dengan metode ELM, input weight dan bias yang jumlah nya 18 *particle* tadi, akan di optimasi terlebih dahulu nilai nya dengan metode PSO guna mendapatkan nilai *input weight* dan bias yang optimal dan menghasilkan nilai prediksi yang baik.

Selanjutnya, masuk ke dalam perhitungan PSO yang mana dalam permasalahan ini jumlah *particle* dalam PSO ada 18, nilai 18 *particle* ini adalah 15 *particle* dari nilai *input weight* dan 3 *particle* dari nilai bias. Input weight dan bias masing-masing memiliki nilai rentan yang berbeda, yakni rentan [-1, 1] untuk nilai *input weight* dan [0,1] untuk nilai bias yang kedua nilainya di dapat secara acak. Jumlah ukuran populasi pada PSO sudah di tentukan terlebih dahulu yaitu 3. Adapun contoh data produksi gula yang digunakan pada penelitian ini ada 15 data dari 45 data yang ada dengan data sebagai berikut. Data pada tabel 4.2

Tabel 4. 2 Data Analisis Produksi Gula

Data Analisis Produksi Gula PG Candi Baru Sidoarjo							
No	**Bulan**	**Hari Giling**	**Rendemen**	**Luas Areal (ha)**	**Pekerja**	**Tebu Digiling (ton)**	**Produksi Gula (ton)**
1	Mei-10	18	5,6	4300	794	411223	22878
2	Jun-10	30	5,6	4300	794	685381	38167
3	Jul-10	31	5,6	4300	794	708226	39414
4	Agu-10	30	5,6	4300	794	685360	38152
5	Sep-10	23	5,6	4300	794	525461	29244
6	Okt-10	31	5,6	4300	794	708200	39409
7	Nov-10	29	5,6	4300	794	662530	36875
8	Des-10	9	5,6	4300	794	205587	11444
9	Mei-11	22	7,29	4400	792	500805	36916
10	Jun-11	30	7,29	4400	792	682920	49197
11	Jul-11	31	7,29	4400	792	705694	51979
12	Agu-11	26	7,29	4400	792	591864	41494
13	Sep-11	27	7,29	4400	792	614628	43182
14	Okt-11	31	7,29	4400	792	705684	51769

15	Nov-11	20	7,29	4400	792	483672	34636

(Sumber: Pabrik Gula Candi Baru-Sidoarjo)

Keterangan setiap parameter data pada Tabel 4.1 adalah sebagai berikut:

1. Bulan : menerangkan masa giling atau masa kerja produksi gula (integer)
2. Hari Giling : menerangkan jumlah hari giling atau hari kerja untuk memproduksi gula (double)
3. Rendemen tebu : menerangkan kadar kandungan gula yang ada di dalam batang tebu yang dinyatakan dalam persen (integer)
4. Luas areal : menerangkan luas areal tanaman tebu (integer)
5. Jumlah pekerja : menerangkan banyaknya tebu yang akan digiling dalam satuan ton (integer)
6. Tebu digiling : menerangkan banyaknya tebu yang akan digiling dalam satuan ton (integer)
7. Produksi gula : menerangkan banyaknya produksi gula yang dihasilkan dalam satuan ton (integer)

4.3 Perancangan Algoritma

4.3.1. Siklus Perhitungan Metode Extreme Learning Machine (ELM) dan Particle Swarm Optimization (PSO)

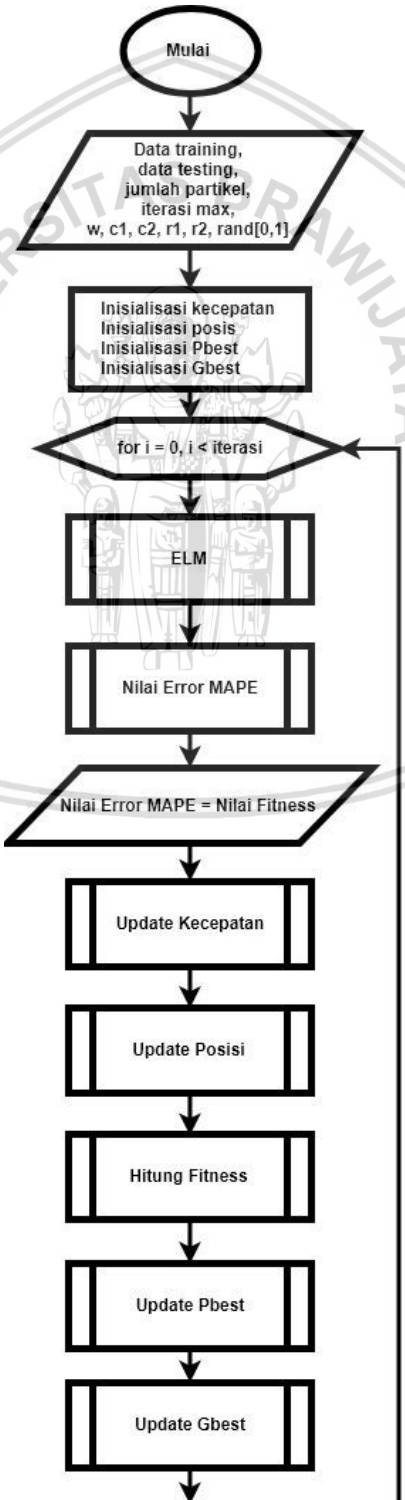
Siklus dari metode hybrid algorithm antara ELM dan PSO adalah suatu penyelesaian dalam menyelesaikan masalah yakni untuk mencari nilai input weight dan bias yang optimal. Langkah-langkah hybrid algorithm ELM dan PSO adalah sebagai berikut:

Langkah-langkah proses hybrid algoritma PSO-ELM :

1. Inisialisasi kecepatan awal, inisilisasi posisi awal dengan nilai random [-1,1] setelah itu mencari nilai fitness pada ELM. Inisialisasi awal pbest bernilai

sama dengan inisialisasi posisi awal dengan nilai fitness yang sama. Inisialisasi gbest didapat dari inisialisasi pbest dengan nilai fitness terendah.

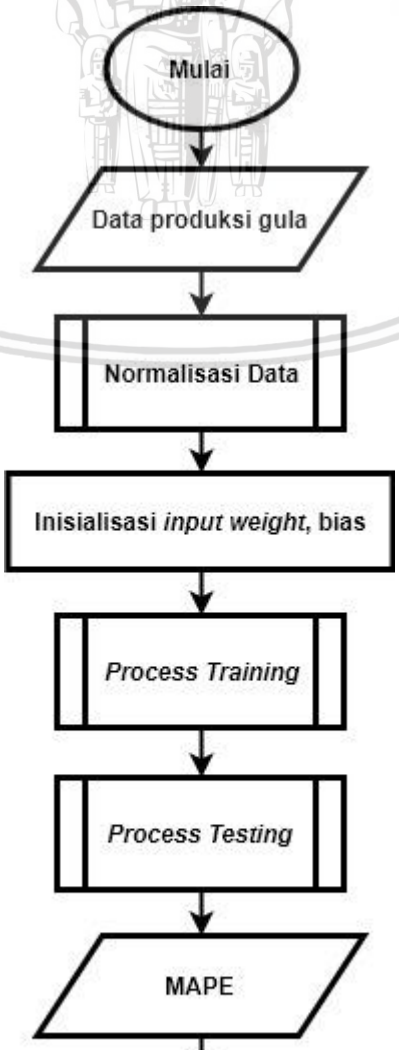
2. Melakukan update kecepatan, posisi, pbest dang best samapi iterasi maksimal.



4.3.2. Si

Gambar 4. 1 Diagram Alir ELM PSO

Siklus metode ELM sendiri, merupakan suatu tahapan penyelesaian masalah secara sekuensial untuk mendapatkan hasil prediksi yang akan dijelaskan pada sub bab berikutnya. Diagram alir proses prediksi produksi gula menggunakan metode ELM ditunjukkan pada Gambar 4.2 berikut.



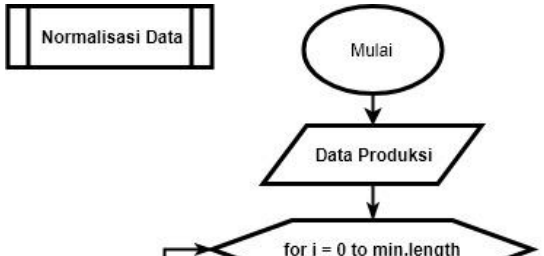
Gambar 4. 2 Diagram Alir Metode ELM

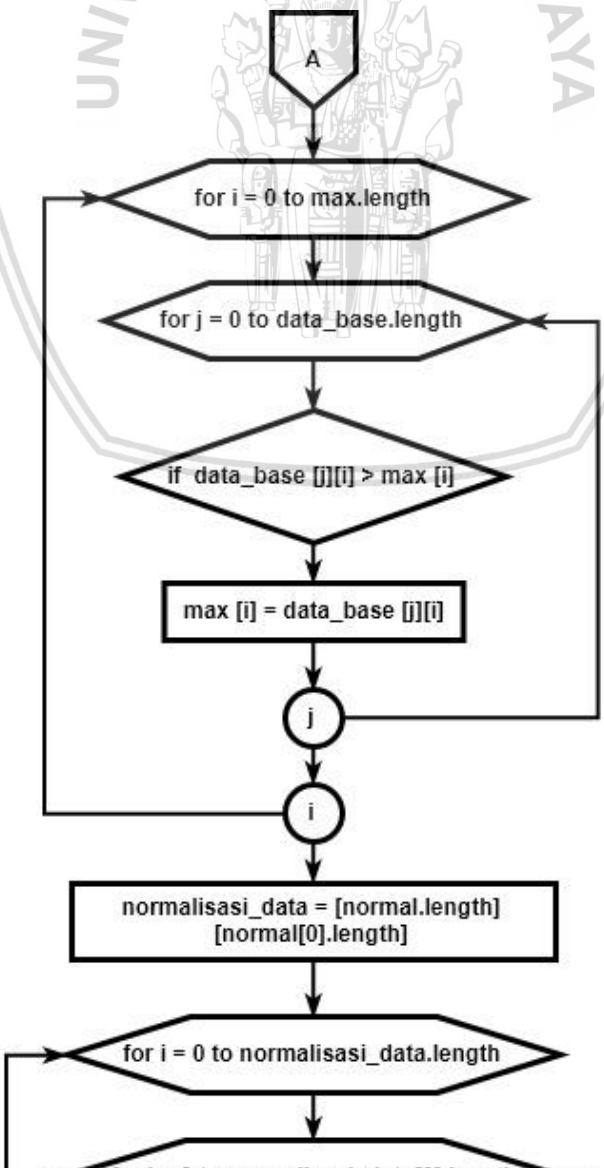
Langkah-langkah proses optimasi bobot menggunakan metode ELM berdasarkan Gambar 4.2 adalah sebagai berikut:

1. Sistem menerima input berupa data hasil produksi gula dari bulan mei 2010 sampai bulan November bulan 2015.
2. Normalisasi data masukan dengan rumus yang sudah ditentukan.
3. Inisialisasi *input weight* pada *range* [-1, 1] dan bias pada *range* [0,1] secara *random*.
4. Melakukan proses training dari data yang telah diterima sistem. Diagram alir untuk *process training* akan ditunjukkan pada Gambar 4.4
5. Melakukan process testing untuk menghasilkan prediksi. Diagram alir untuk proses testing akan ditunjukkan pada Gambar 4.11
6. Sistem mengeluarkan output berupa nilai MAPE.

4.3.2.1 Proses Normalisasi Data

Proses normalisasi data bertujuan untuk standarisasi data yang akan digunakan dalam perhitungan sehingga data berada pada rentan tertentu. Tujuan dari proses normalisasi data sendiri adalah untuk menjamin struktur data yang konsisten dan meminimalkan kerangkapan data sehingga data yang dihasilkan baik. Normalisasi data yang digunakan pada penelitian ini adalah *Min-Max Normalization.* Diagram alir normalisasi data dapat dilihat pada Gambar 4.3 berikut:



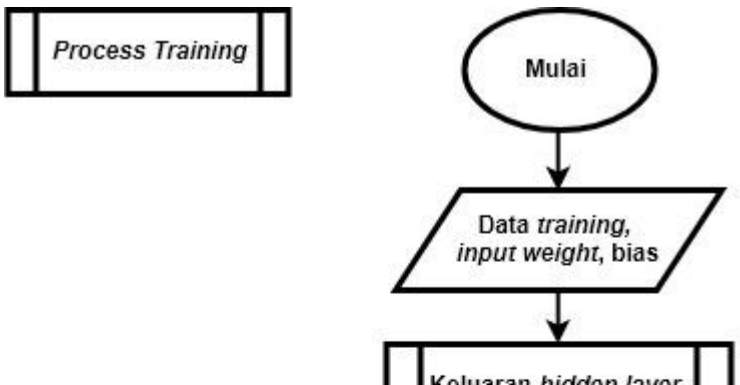

A
for i = 0 to max.length
for j = 0 to data_base.length
if data_base [j][i] > max [i]
max [i] = data_base [j][i]
j
i
normalisasi_data = [normal.length]
[normal[0].length]
for i = 0 to normalisasi_data.length

L **Gambar 4. 3 Diagram Alir Normalisasi Data** *Min-Max Normalization* adalah sebagai berikut:

1. Memasukkan data produksi gula yang akan diinputkan dalam format .xls dalam sistem.
2. Mencari nilai max dan min dari masing-masing parameter data.
3. Melakukan proses perhitungan normalisasi dengan persamaan 2.9.
4. Hasil normalisasi berupa data dengan rentan [0,1].

4.3.2.2 *Process Training*

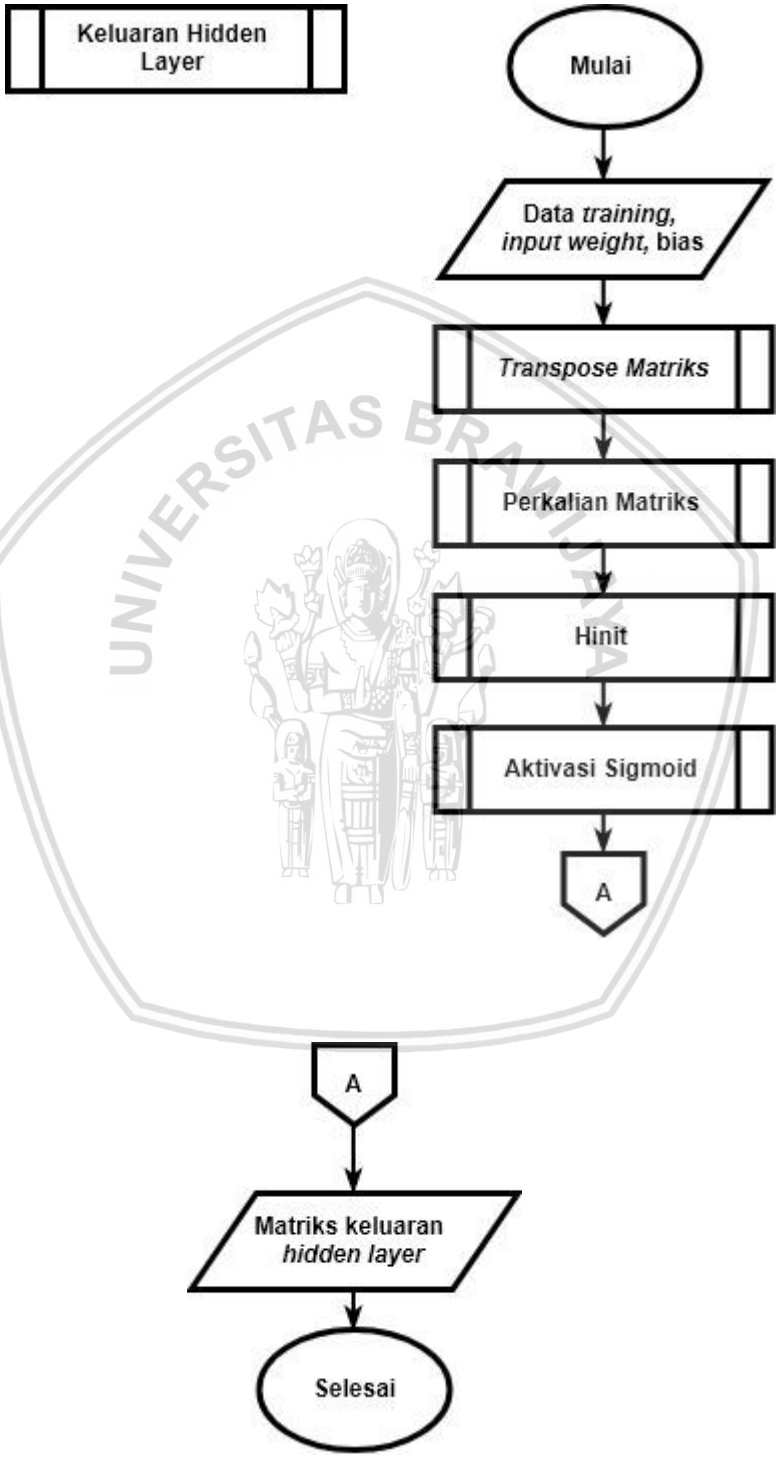
Proses training dilakukan untuk memperoleh nilai *output weight* dimana output weight itu sendiri didapatkan dari beberapa perhitungan. Diagram alir dari process training untuk mendapatkan nilai *output weight* dapat ditunjukkan seperti Gambar 4.4 berikut.



Langkah-langkah yang ada pada *process training* metode ELM adalah sebagai berikut:

1. Sistem menerima input data training berupa nilai data produksi gula yang sudah dinormalisasi. Serta menerima input weight dan bias yang didapatkan secara random pada posisi awal pada metode PSO .
2. Menghitung keluaran hidden layer menggunakan Persamaan 2.1 sedangakan diagram alir untuk hidden layer sendiri akan ditunjukkan pada Gambar 4.5
3. Menghitung matriks-penrose pseudo inverse menggunakan Persamaan 2.2 dan diagram untuk menghitung keluaran matriks-penrose pseudo inverse akan ditunjukkan pada Gambar 4.10
4. Menghitung output weight yang merupakan keluaran sistem menggunakan Persamaan 2.3. Hasil dari ouput weight yang telah didapatkan nantinya akan digunakan kembali untuk menentukan hasil

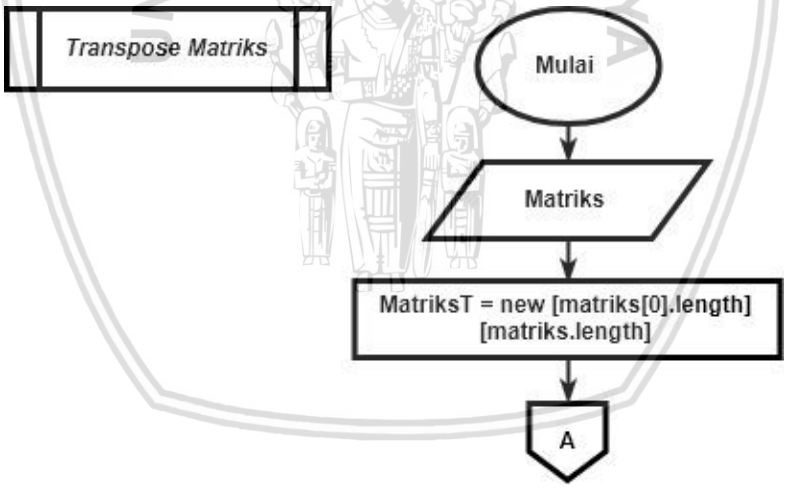
prediksi pada proses testing. Diagram alir untuk menghitung *output weight* akan ditunjukkan pada Gambar 4.11

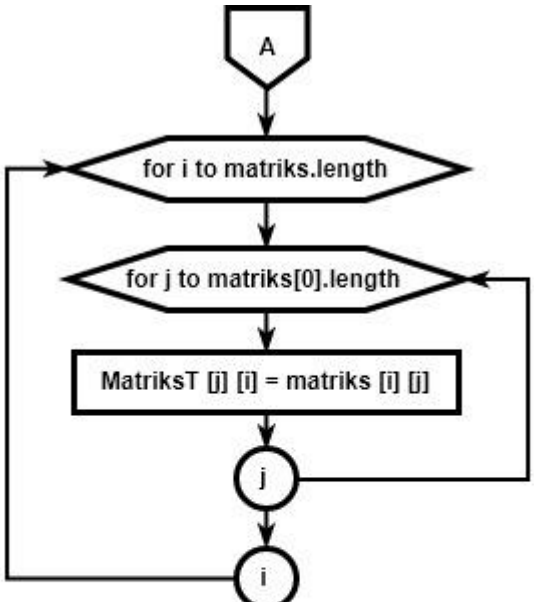


Gambar 4. 5 Diagram Alir Hidden Layer

Langkah-langkah proses perhitungan keluaran hidden layer adalah sebagai berikut:

1. Sistem menerima input berupa data training, input weight dan bias.
2. Melakukan *transpose* untuk *input weight*. Diagram alir untuk proses *transpose* matriks akan ditunjukkan pada gambar 4.6.
3. Melakukan perkalian matriks data training dengan input weight. Diagram alir untuk perkalian matriks ditunjukkan pada Gambar 4.7.
4. Setelah mendapatkan nilai perkalian matriks selanjutnya melakukan perhitungan $H_{init.}$ Perhitungan H_{init} didapat dari penjumlahan hasil perklaian matriks dengan matriks bias. Diagram alir Hinit ditunjukkan pada gambar 4.8.
5. Menghitung keluaran hidden layer beserta fungsi aktivasi menggunakan persamaan 2.2

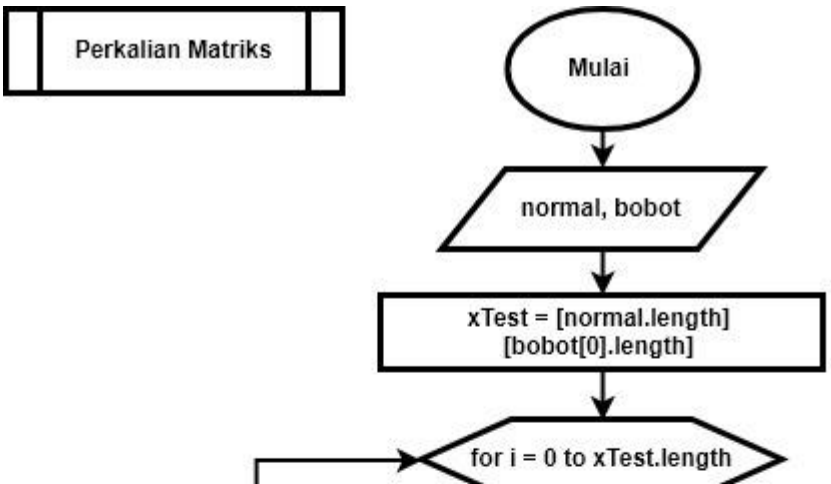




Gambar 4. 6 Diagram Alir Transpose Matriks

Langkah-langkah dalam melakukan proses *transpose* matriks adalah sebagai berikut:

1. Sistem menerima input berupa suatu matriks
2. Membuat array baru dengan panjang baris sama dengan panjang kolom matriks input dan panjang kolom sama dengan panjang baris matriks input.
3. Melakukan transpose matriks dengan cara, mengubah baris dan kolom matriks input yang berordo baris x kolom menjadi kolom dan baris baru yang nantinya akan disimpan pada array baru dengan ordo kolom baris.

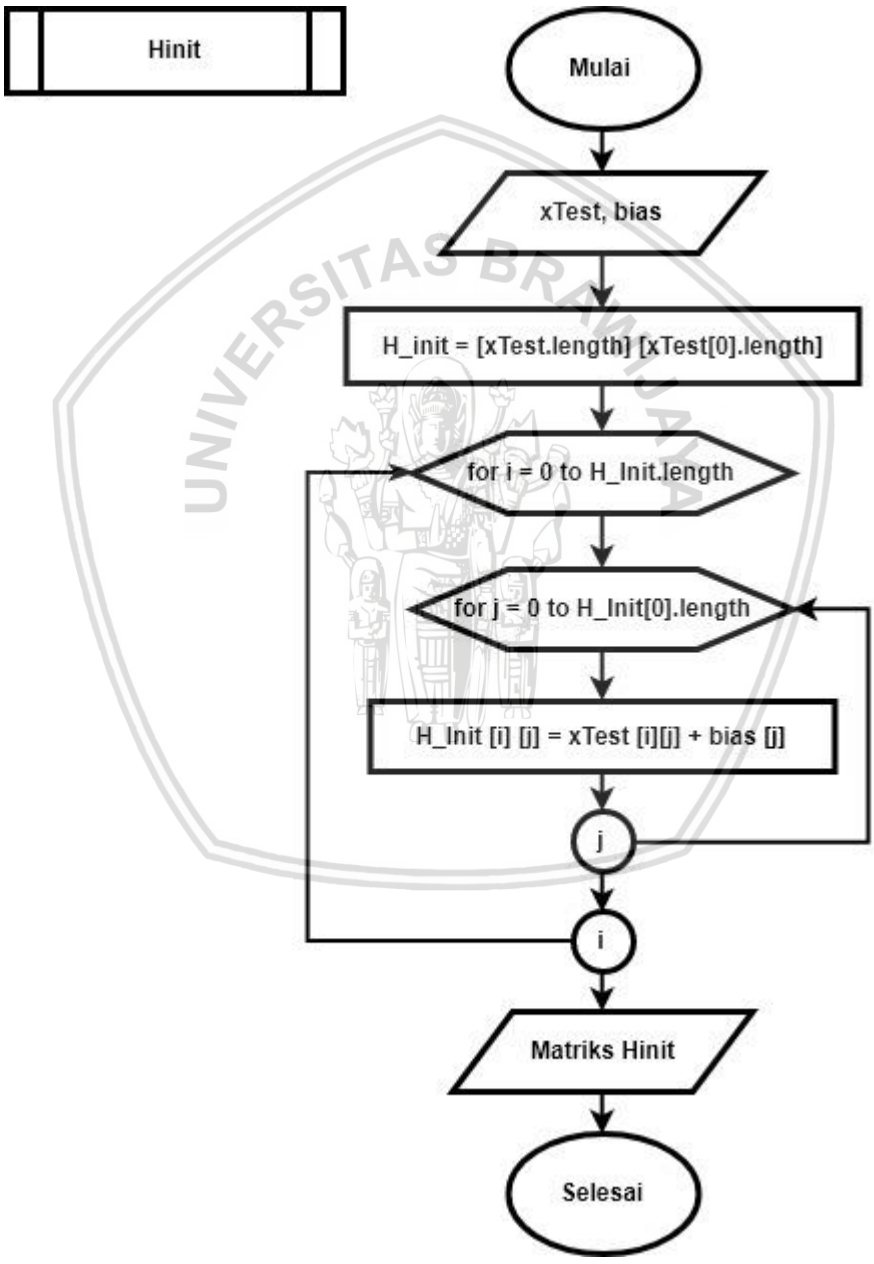


Gambar 4. 7 Diagram Alir Perkalian Matriks

Langkah-langkah untuk melakukan perkalian matriks adalah sebagai berikut:

1. Sistem akan menerima input berupa 2 matriks yang akan diproses.

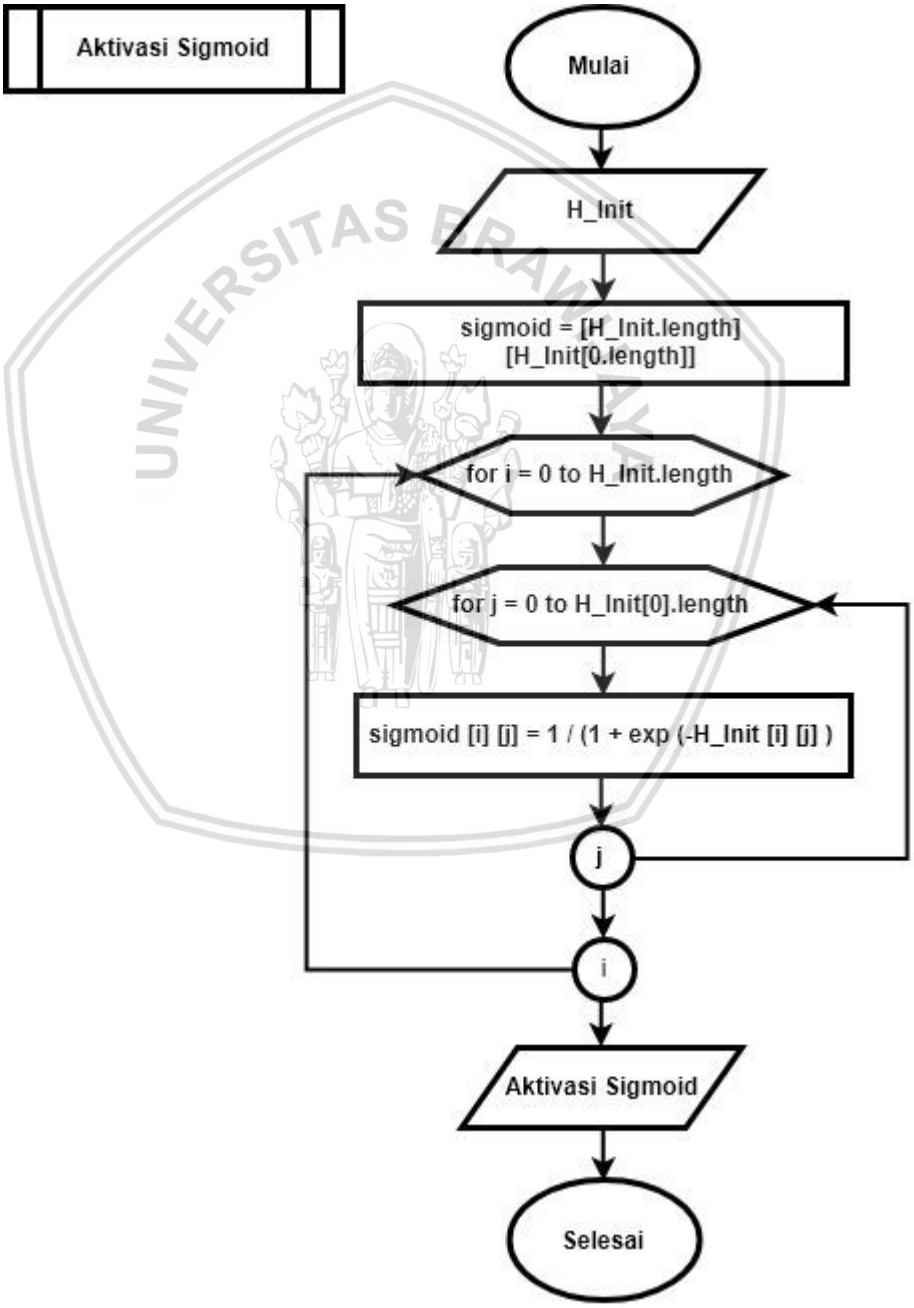
2. Membuat array baru dengan panjang baris sama dengan panjang baris matriks normal dan panjang kolom sama dengan panjang kolom matriks bobot.
3. Melakukan perkalian matriks dengan mengalikan baris matriks normal dengan kolom matriks bobot sebanyak jumlah baris yang terdapat pada matriks normal.



Gambar 4. 8 Diagram Alir Matriks Hinit

Langkah-langkah untuk melakukan perhitungan matriks H_{init} sebagai berikut:

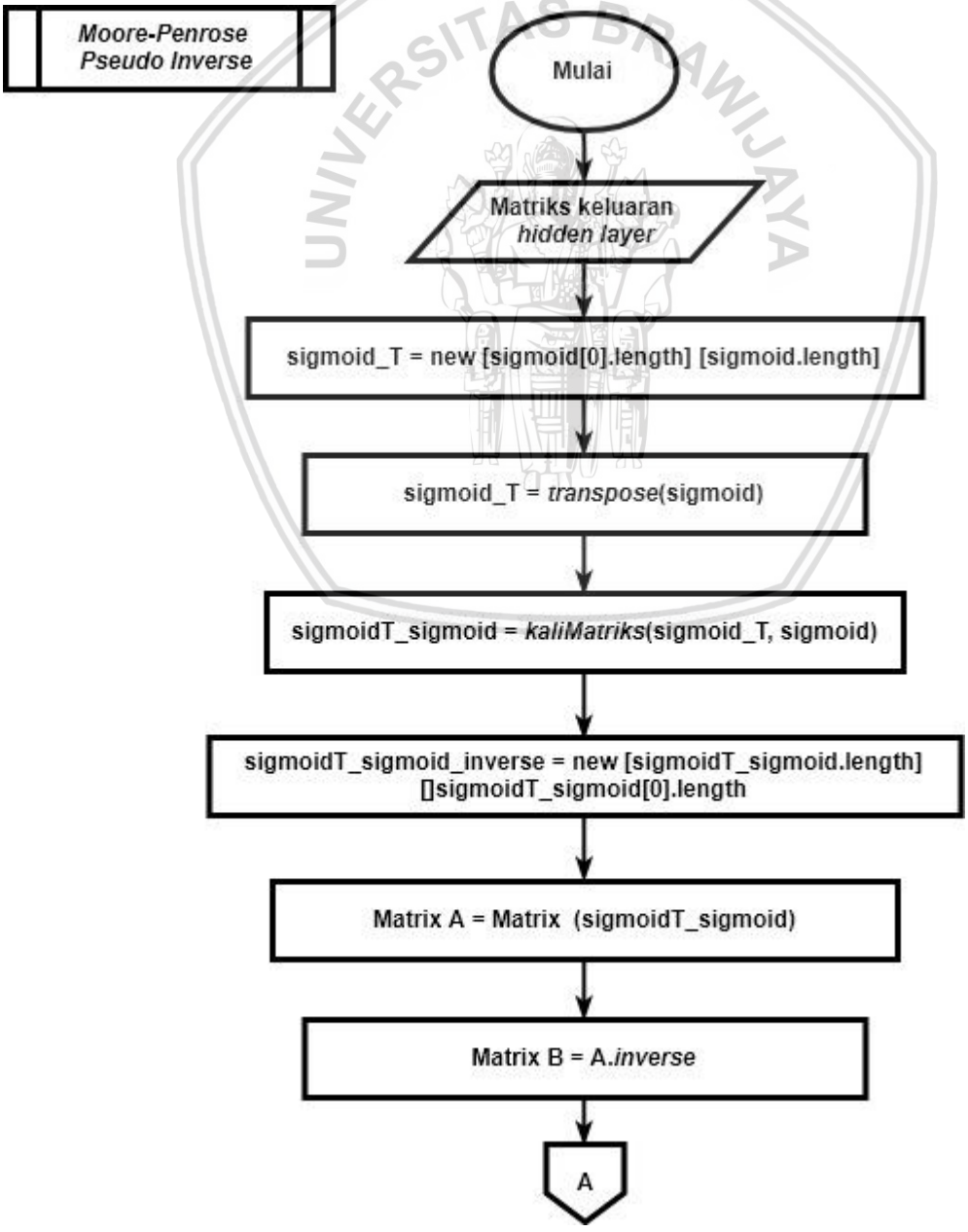
1. Sistem akan menerima input berupa 2 matriks yang akan diproses.
2. Membuat array baru dengan panjang baris dan kolom sama dengan panjang baris matriks xtest.
3. Melakukan perhitungan H_{init} dengan menjumlahkan xTest dengan bias sebanyak jumlah baris dan kolom matriks xTest.

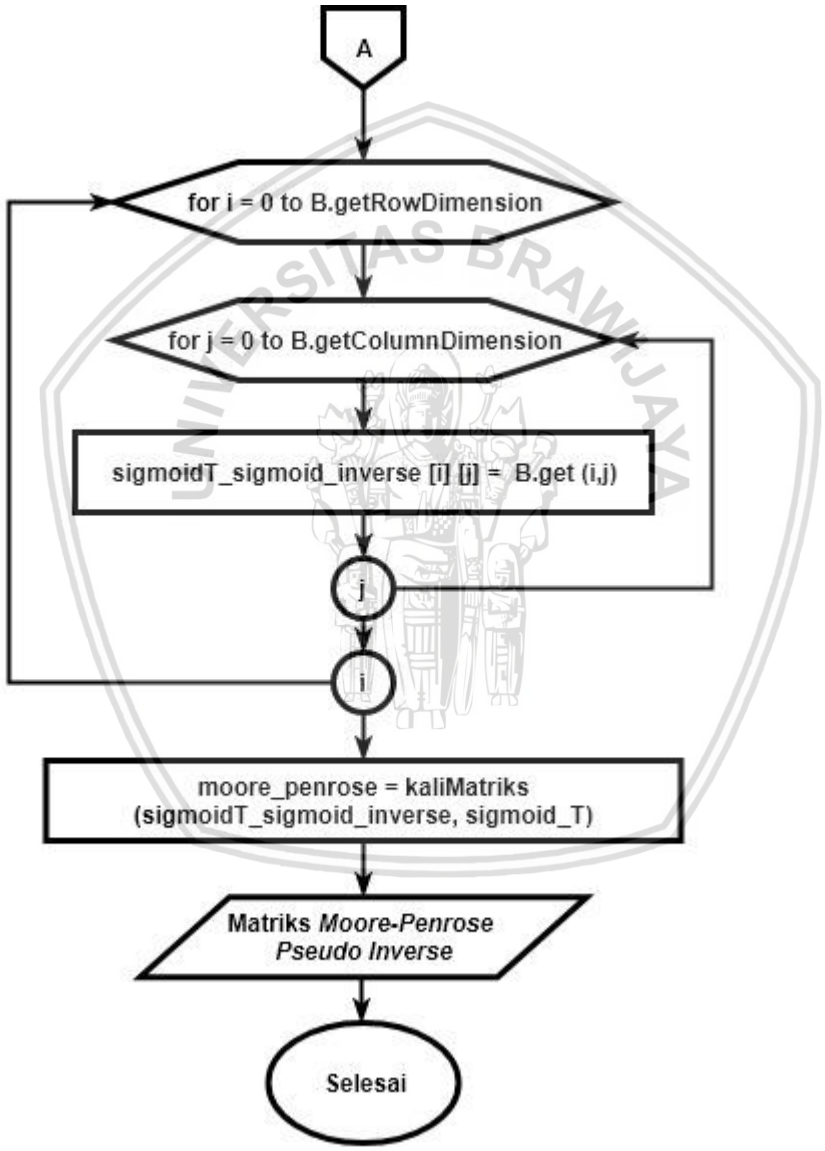


Gambar 4. 9 Diagram Alir Aktivasi Sigmoid

Langkah-langkah untuk melakukan perhitungan aktivasi sigmoid biner sebagai berikut:

1. Sistem akan menerima input berupa hasil dari $H_{init.}$
2. Membuat array baru dengan panjang baris sama dengan panjang baris H_{init}.
3. Melakukan perhitungan sigmoid sesuai dengan persamaan 2.1

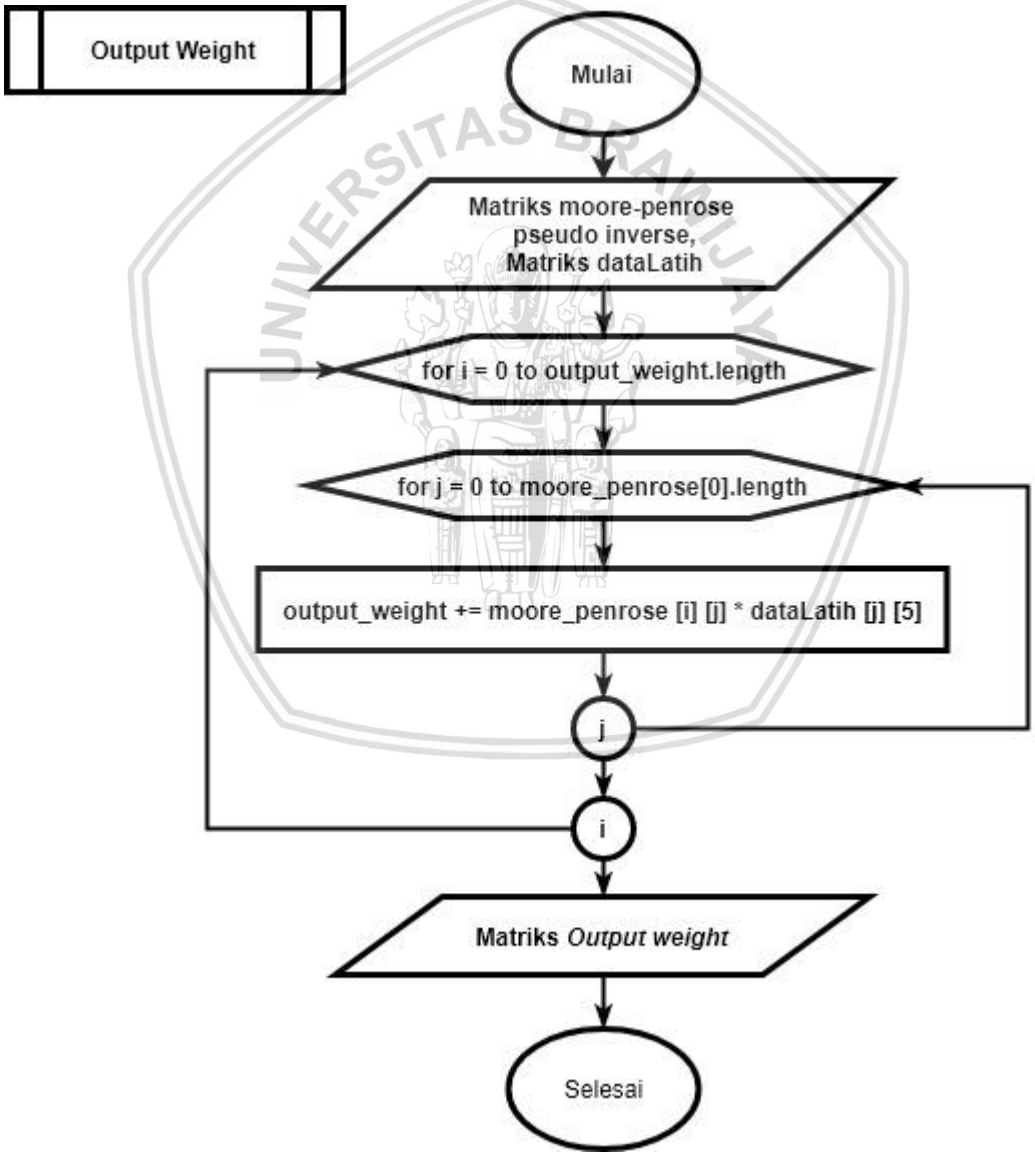




Gambar 4. 10 Diagram Alir *Moore-Penrose Pseudo Inverse*

Langkah-langkah untuk melakukan perhitungan *Moore-Penrose Pseudo Inverse* adalah sebagai berikut:

1. Sistem menerima input berupa hasil dari keluaran *hidden layer.*
2. Melakukan *transpose* matriks keluaran dari *hidden layer.*
3. Melakukan perkalian matriks untuk keluaran dari *hidden layer* dengan hasil *transpose* dari keluaran hidden layer.
4. Melakukan *invers* matriks dari hasil perkalian untuk matriks.
5. Melakukan perkalian matriks untuk matriks hasil *inverse* dengan *transpose* keluaran *hidden layer*.



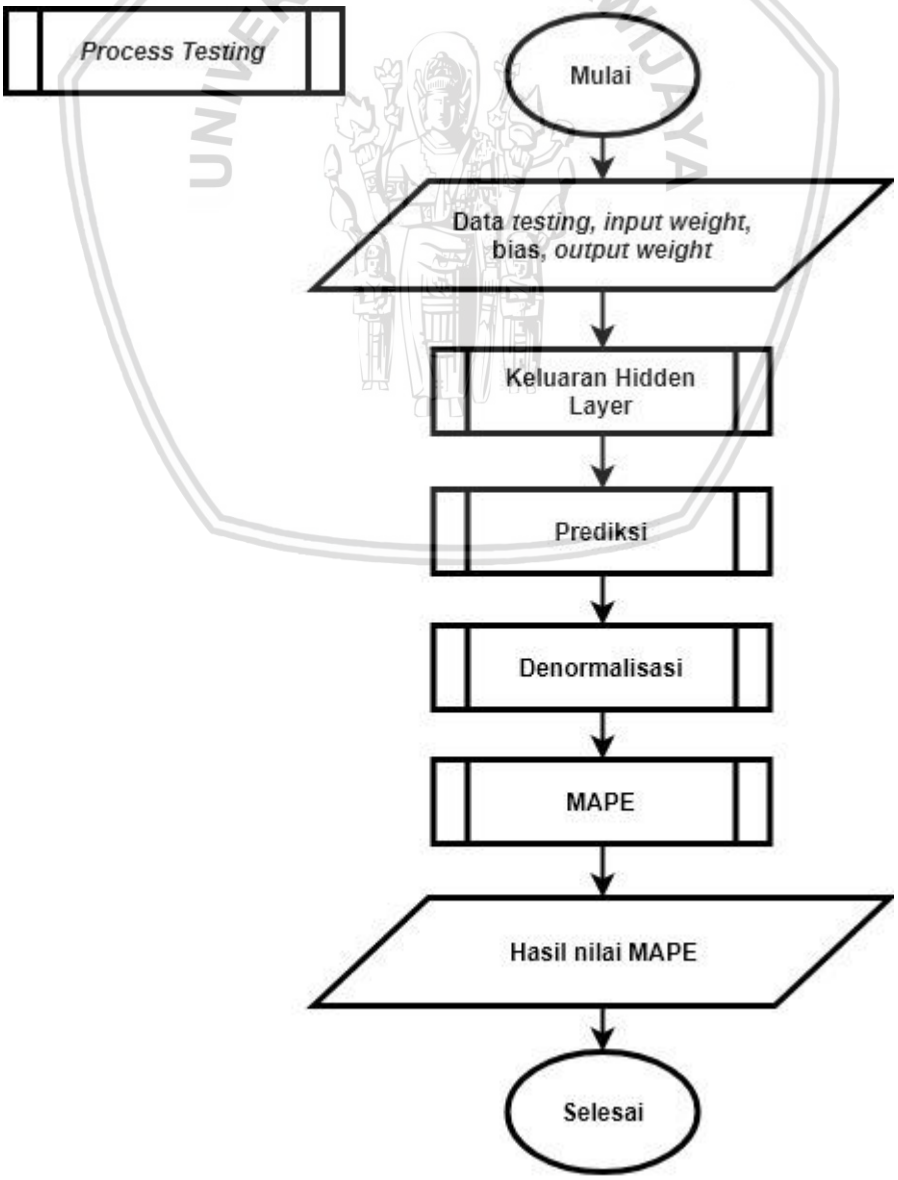
Gambar 4. 11 Diagram Alir *Output Weight*

Langkah-langkah untuk melakukan perhitungan output weight adalah sebagai berikut:

1. Sistem menerima *input* yang didapatkan dari matriks *moore-panrose pseudo invers* dan target data *training*.
2. Melakukan perkalian matriks antara matriks *moore-penrose pseudo invers* dan matriks target data *training* untuk menghasilkan *output weight.*

4.3.2.3 Process Testing

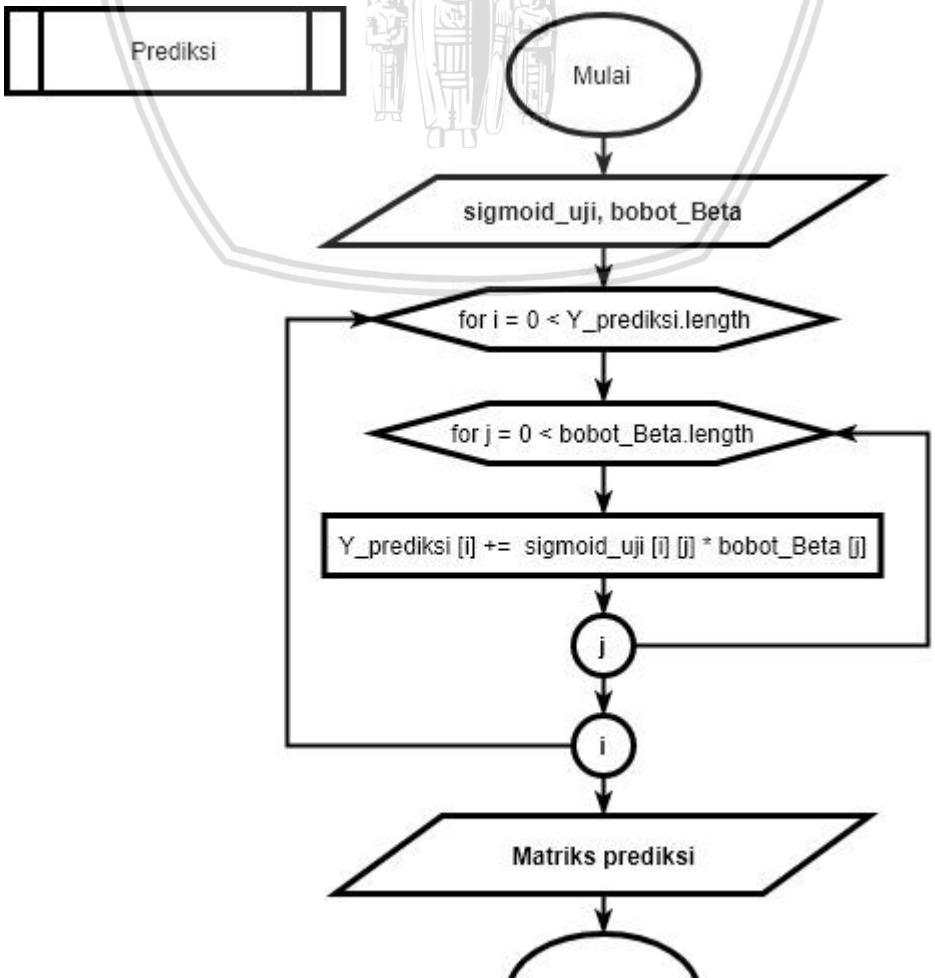
Pada proses *testing* ini, dilakukan berdasarkan *input weight*, *bias* dan *output weight* yang sudah didapatkan dari proses *training*. Setelah itu dilakukanlah proses *testing* dengan langkah-langkah yang sama dan sesuai dengan urutan yang ada pada proses *training*. Diagram alir pada proses testing ditunjukkan pada gambar 4.12



Gambar 4. 12 Diagram Alir Proses *Testing*

Langkah-langkah perhitungan proses testing adalah sebagai berikut:

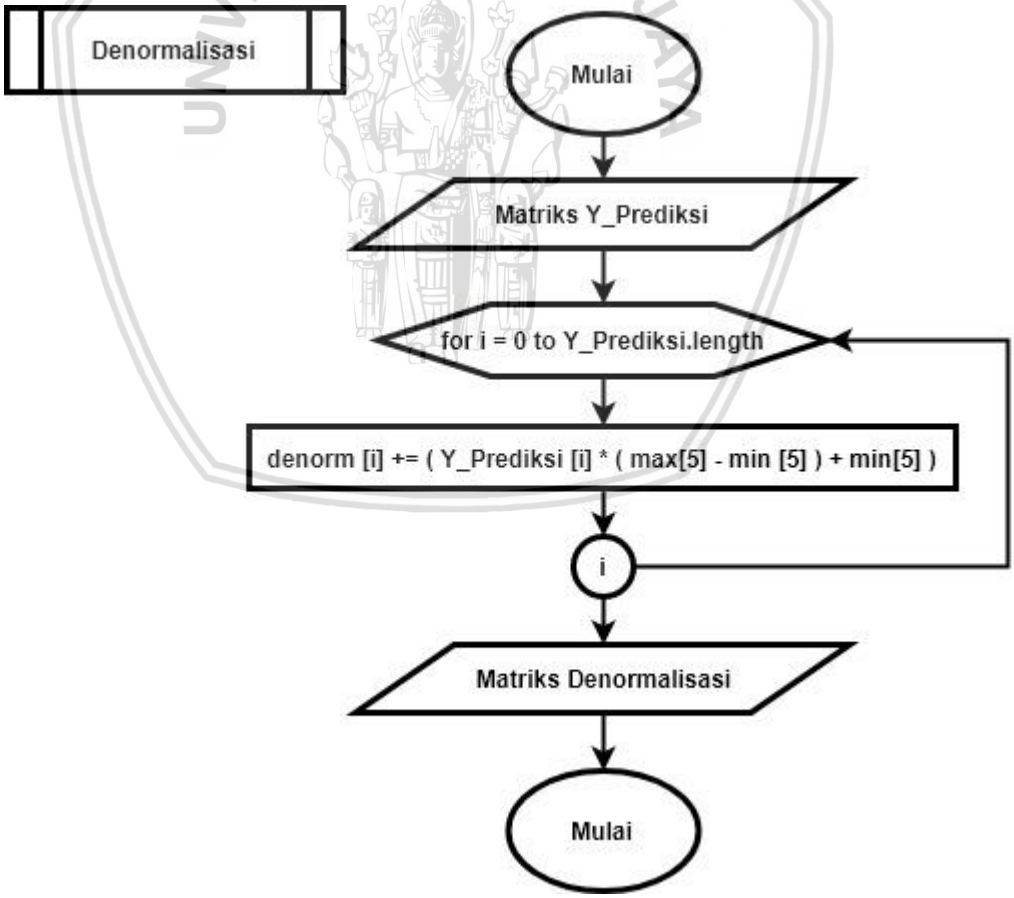
1. Sistem menerima input berupa data testing, input weight, bias dan output weight.
2. Melakukan proses keluaran hidden layer untuk data testing.
3. Melakukan prediksi untuk output sistem. Diagram alir untuk melakukan prediksi sistem ditunjukkan pada gambar diagram 4.13
4. Melakukan perhitungan denormalisasi data untuk mengembalikan data yang telah di normalisasikan.
5. Denormalisasi data untuk mengembalikan data yang sudah di normalisasi. Diagram alir untuk denormalisasi data sendiri akan ditunjukkan pada gambar 4.14.
6. Evaluasi hasil dengan menghitung tingkat kesalahan menggunakan MAPE. Diagram alir untuk MAPE sendiri akan ditunjukkan pada gambar 4.15



Gambar 4. 13 Diagram Alir Prediksi

Langkah-langkah untuk melakukan prediksi sistem adalah sebagai berikut:

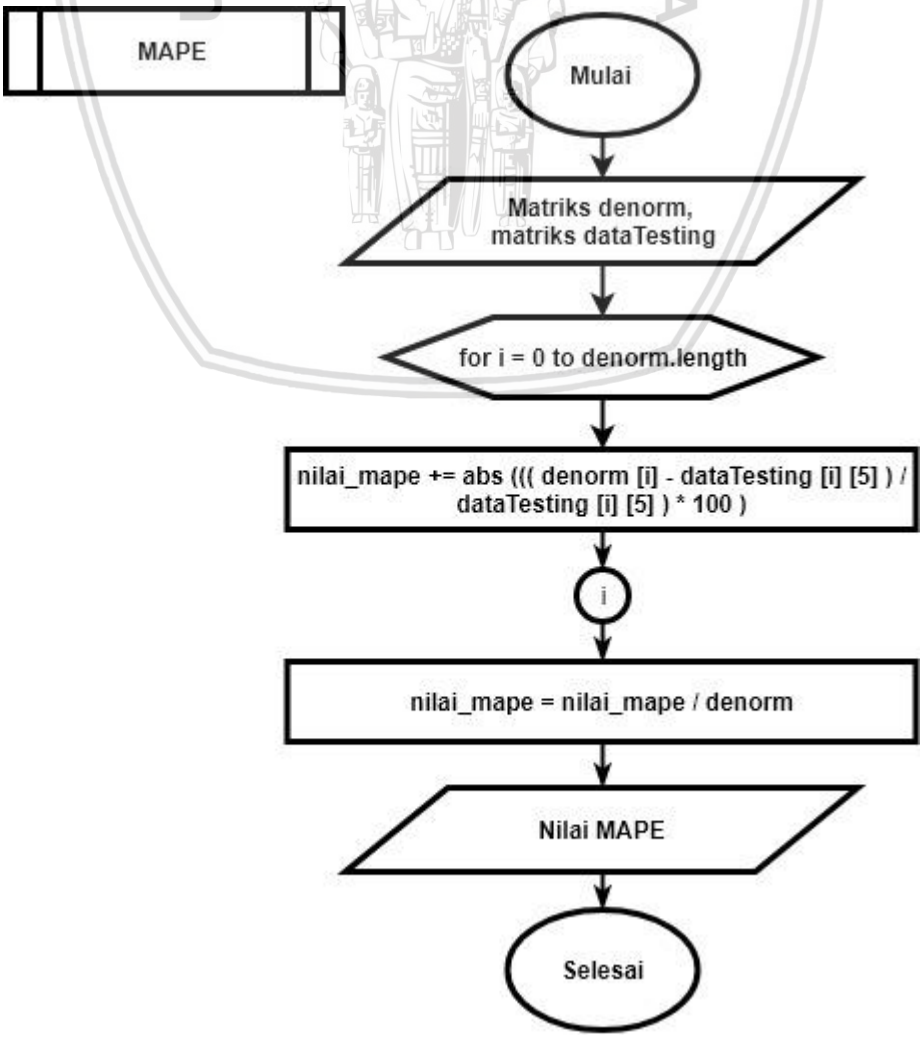
1. Sistem menerima input berupa matriks keluaran sigmoid dan bobot Beta.
2. Melakukan perkalian matriks antara matriks keluaran sigmoid dan matriks bobot beta untuk menghasilkan nilai prediksi sistem.



Gambar 4. 14 Diagram Alir Denormalisasi

Langkah-langkah proses denormalisasi adalah sebagai berikut:

1. Sistem menerima input berupa hasil matriks prediksi.
2. Sistem akan melakukan perhitungan denormalisasi data dengan persamaan 2.9
3. Hasil denormalisasi berupa data hasil prediksi.



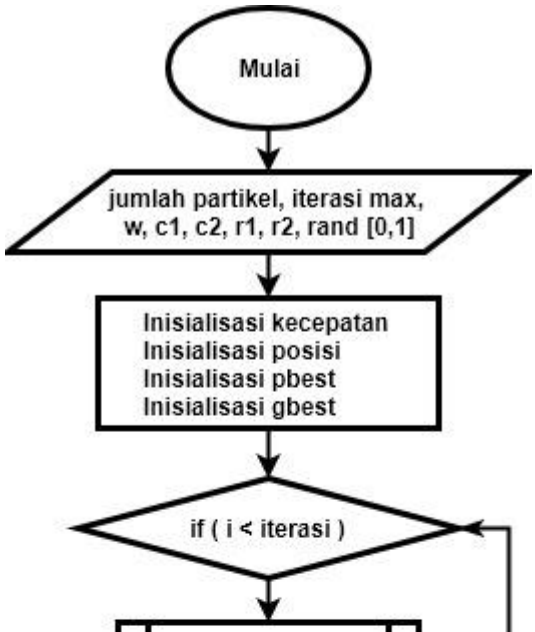
Gambar 4. 15 Diagram Alir PSO

Langah-langkah perhitungan MAPE adalah sebagai berikut:

1. Input data berupa data hasil prediksi yang telah didenormalisasi dan target data testing.
2. Menghitung nilai MAPE dengan persamaan 2.10
3. Hasil keluaran nilai MAPE berupa nilai *error* dalam satuan persen (%)

4.3.3 Siklus Metode *Particle Swarm Optimization* (PSO)

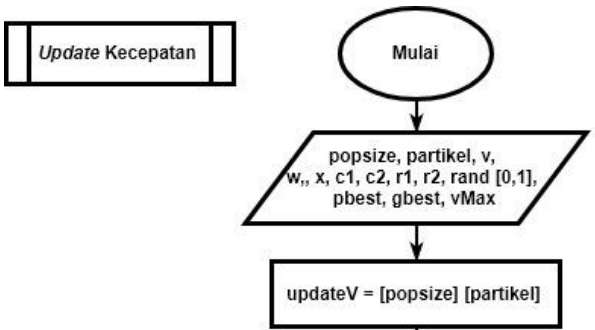
Siklus metode PSO pada penelitian ini merupakan penyelesaian masalah yang dilakukan secara berulang-ulang untuk mendapatkan nilai input weight dan bias yang paling optimal. Langkah-langkah metode PSO ini akan dijelaskan pada sub bab berikut. Diagram alir proses PSO ditunjukkan seperti gambar 4.16



Gambar 4. 16 Diagram Alir PSO

Langkah-langkah seleksi fitur perhitungan PSO adalah sebagai berikut:

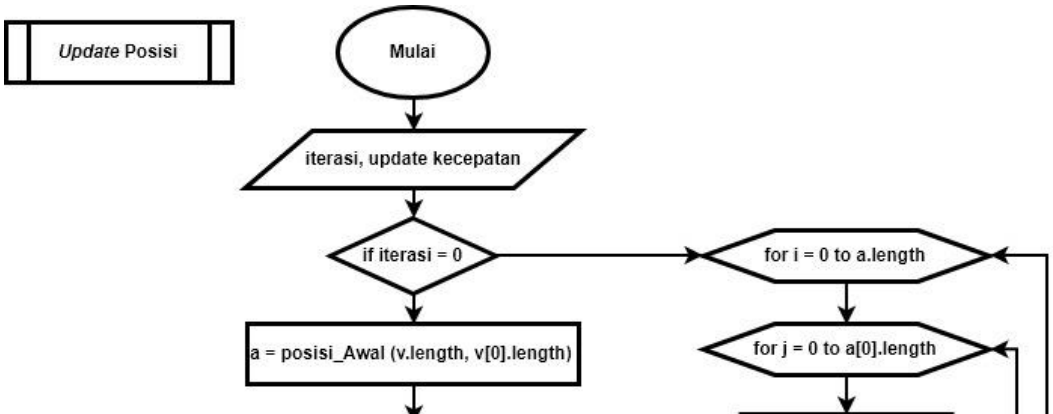
1. Sistem menerima *input* berupa jumlah partikel, iterasi maksimum, bobot inersia, konstanta dan bilangan random yang dibutuhkan pada metode PSO.
2. Melakukan inisialisasi dengan memberi nilai 0 pada kecepatan awal, inisialisasi posisi awal yang mana nilainya ditentukan secara random, inisialisasi *Pbest*, dan *Gbest.*
3. Melakukan update kecepatan, posisi, Pbest dan Gbest selama iterasi masih terpenuhi.
4. Sistem menghasilkan nilai input weight dan bias yang optimal.



Gambar 4. 17 Diagram Alir *Update* Kecepatan

Langkah-langkah update kecepatan adalah sebagai berikut:

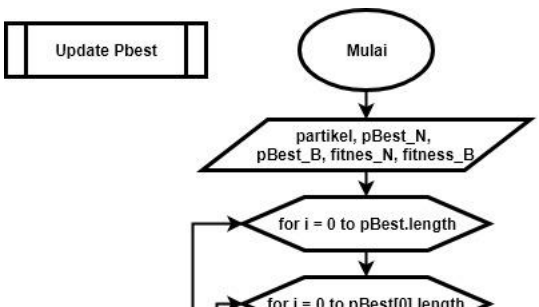
1. Sistem menerima input berupa bobot inersia, konstanta, bilangan random, kecepatan yang ada pada iterasi sebelumnya, posisi, Pbest dan Gbest partikel.
2. Melakukan update kecepatan menggunakan persamaan 2.5



Gambar 4. 18 Diagram Alir *Update* Posisi

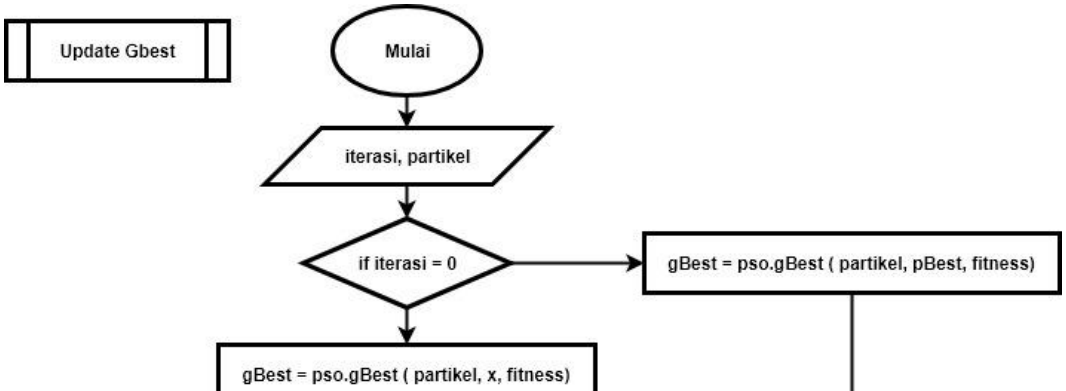
Langkah-langkah *update* posisi adalah sebagai berikut:

1. Sistem menerima input berupa nilai posisi awal dan *update* kecepatan.
2. Melakukan update posisi sesuai dengan persamaan 2.7



Gambar 4. 19 Diagram Alir *Update Pbest*

Pada proses *update Pbest*, dilakukan dengan cara membandingkan nilai *fitness* dari update posisi dengan nilai *Pbest* pada iterasi sebelumnya. Apabila nilai *fitness* lebih kecil maka nilai tersebut yang digunakan sebagai nilai *fitness* pada *update Pbest.*



Gambar 4. 20 Diagram Alir Gbest

Proses perhitungan pada Gbest ini adalah dengan cara memilih nilai fitness pada Pbest sebelumnya dengan nilai fitness yang paling kecil atau rendah.

4.4 Perhitungan Manual

Perhitungan manual adalah contoh perhitungan dari sistem yang akan dibuat untuk mengetahui kebenaran dari perhitungan sistem nanti. Terhadap beberapa langkah dalam perhitungan manual ELM dan PSO seperti yang sudah di jelaskan pada diagram alir 4.20 . Pada perhitungan kali ini, digunakan beberapa contoh data dari keseluruhan data yang ada pada Lampiran A. Data yang digunakan untuk perhitungan manual adalah 15 data dari 45 data keseluruhan, yakni 12 data untuk data training dan 3 data untuk data testing dan *hidden neuron* sebanyak 3. Berikut tabel untuk data *training* dan data *testing.*

Tabel 4. 3 Data Training

No	Bulan	Hari Giling	Rendemen	Luas Areal (ha)	Pekerja	Tebu Digiling (kw)	Produksi Gula (kw)
1	Mei-10	18	5,6	4300	794	411223	22878
2	Jun-10	30	5,6	4300	794	685381	38167
3	Jul-10	31	5,6	4300	794	708226	39414
4	Agu-10	30	5,6	4300	794	685360	38152
No	**Bulan**	**Hari Giling**	**Rendemen**	**Luas Areal**	**Pekerja**	**Tebu Digiling**	**Produksi Gula (kw)**

				(ha)		(ton)	
5	Sep-10	23	5,6	4300	794	525461	29244
6	Okt-10	31	5,6	4300	794	708200	39409
7	Nov-10	29	5,6	4300	794	662530	36875
8	Des-10	9	5,6	4300	794	205587	11444
9	Mei-11	22	7,29	4400	792	500805	36916
10	Jun-11	30	7,29	4400	792	682920	49197
11	Jul-11	31	7,29	4400	792	705694	51979
12	Agu-11	26	7,29	4400	792	591864	41494

Tabel 4. 4 Data testing

No	Bulan	Hari Giling	Rendemen	Luas Areal (ha)	Pekerja	Tebu Digiling (kw)	Produksi Gula (kw)
1	Sep-11	27	7,29	4400	792	614628	43182
2	Okt-11	31	7,29	4400	792	705684	51769
3	Nov-11	20	7,29	4400	792	483672	34636

Berikut langkah-langkah perhitungan manual dengan menggunkan metode ELM dan PSO.

4.4.1 Normalisasi Data

Sebelum memulai perhitungan, data masukan pada sistem berupa angka yang sangat besar. Oleh karena itu, untuk memudahkan proses perhitungan dan agar nilai berada pada range tertentu maka dilakukanlah normalisasi data. Langkah-langkah dalam proses perhitungan normalisasi data adalah sebagai berikut.

Langkah 1: Mencari nilai maximum dan minimum masing-masing dari setiap parameter. Berikut adalah tabel nilai maximum dan minimum masing-masing parameter.

Tabel 4. 5 Nilai Maximum dan Minimum

No	Hari Giling	Rendemen	Luas Areal (ha)	Pekerja	Tebu Digiling (kw)	Produksi Gula (kw)
Maximum	31	7,29	4400	794	708226	51979
Minimum	9	5,6	4300	792	205587	11444

Langkah 2: menghitung normalisasi data dari masing-masing nilai parameter seperti persamaan 2.8. Berikut ini contoh dari perhitungan normalisasi data.

$$x' = \frac{(x - x_{min})}{(x_{max} - x_{min})}$$
$$= \frac{(18 - 9)}{(31-9)} = 0{,}4090909$$

Berikut tabel dari hasil normalisasi data.

Tabel 4. 6 Normalisasi Data

No	X1	X2	X3	X4	X5	T
1	0,409090909	0	0	1	0,409112703	0,282077217
2	0,954545455	0	0	1	0,954549886	0,659257432
3	1	0	0	1	1	0,69002097
4	0,954545455	0	0	1	0,954508106	0,658887381
5	0,636363636	0	0	1	0,636389138	0,439126681
6	1	0	0	1	0,999948273	0,689897619
7	0,909090909	0	0	1	0,909087834	0,627383742
8	0	0	0	1	0	0
9	0,590909091	1	1	0	0,58733604	0,628395214
10	0,954545455	1	1	0	0,949653728	0,931367954
11	1	1	1	0	0,994962587	1
12	0,772727273	1	1	0	0,768497868	0,741334649
13	0,818181818	1	1	0	0,813786833	0,782977674
14	1	1	1	0	0,994942692	0,994819292
15	0,5	1	1	0	0,553249947	0,572147527

4.4.2 Inisialisasi Kecepatan Awal dan Posisi Awal Partikel

Inisialisasi kecepatan awal partikel pada perhitungan PSO dilakukan dengan memberi nilai 0 sedangkan posisi awal partikel dilakukan secara *random* yang berisi nilai dari inisialisasi input weight dan bias yang akan dibahas pada sub bab berikutnya. Jumlah partikel sendiri berjumlah 18 partikel dimana 15 partikel merupakan nilai input weight dan 3 partikel merupakan nilai bias. Berikut tabel inisialisasi kecepatan awal partikel dan posisi partikel.

Tabel 4. 7 Inisialisasi Kecepatan Awal

NO	1	2	3	4	5	6	7	8	9	...	18
V1 (0)	0	0	0	0	0	0	0	0	0	...	0
V2 (0)	0	0	0	0	0	0	0	0	0	...	0
V3 (0)	0	0	0	0	0	0	0	0	0	...	0

Tabel 4. 8 Inisialisasi Posisi Awal Partikel

NO	1	2	3	4	5	6	7	..	18	Fitness
X1 (0)	0,6	0,1	-0,1	0	0,02	0	-0,1	...	0,74	
X2 (0)	0,7	0,5	0,4	0,4	0,8	0,1	0,3	...	0,9	
X3 (0)	0	-0,79	-0,73	0,34	0,61	0,46	0,14	...	0,35	

Untuk mendapatkan nilai fitness, maka diperlukan perhitungan ELM terlebih dahulu. Hasil perhitungan nilai error MAPE pada ELM digunakan sebagai nilai fitness pada perhitungan PSO yang selanjutnya perolehan nilai fitness akan dibahas pada proses perhitungan prediksi metode ELM.

4.4.3 Inisialisasi input weight dan bias

Inisialisasi *input weight* dan bias dilakukan secara *random* dengan *range* untuk *input weight* adalah [-1, 1] dan untuk bias adalah [0,1] dan ukuran matriks *jxk* dimana j merupakan banyaknya *hidden neuron* yang digunakan dan k merupakan banyaknya *input neuron*. Pada perhitungan manual kali ini jumlah *input neuron* sebanyak 5 dan *hidden neuron* yang digunakan sebanyak 3. Sehingga didapatan ukuran matriks *input weight* adalah 3x5 dengan *range* [-1,1] yang ditunjukkan pada tabel berikut.

Tabel 4. 9 Matriks Nilai Input Weight

W	1	2	3	4	5	6	7	8	9	....	15
1	0,6	0,1	-0,1	0	0,02	0	-0,1	0,05	0,1	....	0,09
2	0,7	0,5	0,4	0,4	0,8	0,1	0,3	0,2	0,4	....	0,4
3	0	-0,79	-0,73	0,34	0,61	0,46	0,14	0	-0,67	....	0

Inisialisasi bias diberikan secara random dengan range [0,1] dengan ukuran matriks *1xj*. Berikut tabel inisialisasi bias.

Tabel 4. 10 Nilai Bias

b	1	2	3
1	0,12	0,04	0,74
2	0,4	0,4	0,9
3	0,54	0,71	0,35

4.4.4 Proses Training

Langkah-langkah perhitungan untuk proses training pada metode ELM adalah sebagai berikut:

Langkah 1: Perhitungan Keluaran *Hidden Layer* (H)

Perhitungan keluaran hidden layer didapatkan dari hasil perkalian antara matriks input data (data training) dengan input weight dan bias yang nantinya nilai tersebut akan dipetakan menggunakan fungsi aktivasi sigmoid biner sehingga menghaislkan nilai dengan range 0 sampai 1. Berikut adalah contoh perhitungan keluaran hidden layer yang sesuai dengan persamaan 2.1

Langkah 1.1: Menghitung H_{init}

$$(X * W^T) + b = ((0{,}41 \text{ x } 0{,}6) + (0 \text{ x } 0{,}1) + (0 \text{ x } -0{,}1) + (1 \text{ x } 0) + (0{,}41 \text{ x } 0{,}02)) + 0{,}12$$
$$= 0{,}374$$

Berikut hasil keluaran dari perhitungan H_{init}

Tabel 4. 11 Hasil Nilai Keluaran H_{init}

H_{init}	1	2	3
1	0,373637	0,17682	0,664547
2	0,711818	0,225909	0,697273
3	0,74	0,23	0,7
4	0,711817	0,225906	0,697269
5	0,514546	0,197275	0,678184
6	0,739999	0,229995	0,699995
7	0,683636	0,221818	0,694545
8	0,12	0,14	0,64
9	0,486292	0,04286	0,675133
10	0,71172	0,075469	0,696832
11	0,739899	0,079547	0,699547
12	0,599006	0,059165	0,685983

Langkah1.2: Menghitung Fungsi Aktivasi

$$H = \frac{1}{1 + \exp(-H_{init})} = \frac{1}{1 + \exp(-0{,}374)} = 0{,}592$$

Berikut hasil dari keluaran Fungsi Aktivasi dari nilai keluaran *Hidden Layer*

Tabel 4. 12 Hasil Keluaran Hidden Layer (H)

H	1	2	3
1	0,592337	0,54409	0,660281161

2	0,670803	0,556238	0,667582912
3	0,676996	0,557248	0,668187772
4	0,670803	0,556237	0,667582078
5	0,625872	0,549159	0,66333329
6	0,676996	0,557247	0,66818674
7	0,66455	0,555228	0,66697726
8	0,529964	0,534943	0,654753461
9	0,619233	0,510713	0,662651563
10	0,670781	0,518858	0,667485117
11	0,676974	0,519876	0,668087247
12	0,645429	0,514787	0,665072728

Langkah 2: Menghitung Matriks Moore-Penrose Inverse (H^+)

Matriks *Moore-Penrose Pseudo Inverse* merupakan suatu matriks perkalian antara matriks *inverse* dan *transpose* dari hasil keluaran *Hidden Layer* sesuai dengan persamaan 2.2 . Berikut contoh perhitungan Matriks *Moore-Penrose Inverse*.

Langkah 2.1: Matriks *Transpose* dari Matriks (H)

Hasil matriks dari transpose matriks (H)

Tabel 4. 13 Matriks Transpose H

H^T	1	2	3	4	5	6
1	0,529234	0,670803	0,676996	0,670803	0,625872	0,676996
2	0,54409	0,556238	0,557248	0,556237	0,549159	0,557247
3	0,660281	0,667583	0,668188	0,667582	0,663333	0,668187

H^T	7	8	9	10	11	12
1	0,66455	0,529964	0,619233	0,670781	0,676974	0,645429
2	0,555228	0,534943	0,510713	0,518858	0,519876	0,514787
3	0,666977	0,654753	0,662652	0,667485	0,668087	0,665073

Langkah 2.2: Menghitung Perkalian Matriks Transpose H (H^T) dengan Matriks H

Berikut contoh perhitungan dari perkalian matriks transpose H (H^T) dengan matriks H.

$$H^T * H = ((0{,}529 \text{ x } 0{,}529) + (0{,}670 \text{ x } 0{,}670) + (0{,}677 \text{ x } 0{,}677) + (0{,}670 \text{ x } 0{,}670) + (0{,}625 + 0{,}625) + (0{,}676 + 0{,}676) + (0{,}664 + 0{,}664) + (0{,}529 + 0{,}529) + (0{,}619 \text{ x } 0{,}619) + (0{,}670 \text{ x } 0{,}670) + (\ 0{,}676 \text{ x } 0{,}676) + (0{,}645 + 0{,}645)$$

$= 4{,}952557$

Tabel 4. 14 Hasil Perkalian Matriks Transpose H (H^T) dengan Matriks H

H^T . H	1	2	3
1	4,952557	4,133384	5,0948
2	4,167716	3,497221	4,305916
3	5,136465	4,305917	5,30713

Langkah 2.3: Menghitung Invers Matriks dari Matriks Hasil Perhitungan H^T . H

Perhitungan invers matriks didapatkan dari persamaan OBE (Operasi Baris Elementer). Persaman OBE sendiri merupakan salah satu solusi dalam menyelesaikan invers matriks. Perhitungan inverse matriks dengan persamaan OBE adalah dengan cara menyandingkan matriks yang akan menjadi matriks invers dengan matriks identitas, setelah itu mengubah matriks yang akan menjadi matriks inverse menjadi matriks identitas dan matriks identitas itu menjadi sebuah matriks baru. Matriks baru tersebut yang nantinya menjadi hasil dari invers matriks. Berikut perhitungan invers matriks dengan persamaan OBE.

1. Bentuk matriks identitas (*I*)

$$[H \mid I] = \begin{bmatrix} 4{,}9526 & 4{,}1334 & 5{,}0948 \\ 4{,}1677 & 3{,}4972 & 4{,}3059 \\ 5{,}1364 & 4{,}3059 & 5{,}3071 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

2. B1 ÷ 4,9256 → B1

$$[H \mid I] = \begin{bmatrix} 1 & 0{,}8346 & 1{,}0344 \\ 4{,}1677 & 3{,}4972 & 4{,}3059 \\ 5{,}1364 & 4{,}3059 & 5{,}3071 \end{bmatrix} \begin{bmatrix} 0{,}2030 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

3. B2 – (4,1677 x B1) → B2

$$[H \mid I] = \begin{bmatrix} 1 & 0{,}8346 & 1{,}0344 \\ 0 & 0{,}0189 & 0{,}0185 \\ 5{,}1364 & 4{,}3059 & 5{,}3071 \end{bmatrix} \begin{bmatrix} 0{,}2019 & 0 & 0 \\ -0.8415 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

4. B3 – (5,1364 x B1) → B3

$$[H \mid I] = \begin{bmatrix} 1 & 0{,}8346 & 1{,}0344 \\ 0 & 0{,}0189 & 0{,}0185 \\ 0 & 0{,}0190 & 0{,}0231 \end{bmatrix} \begin{bmatrix} 0{,}2019 & 0 & 0 \\ -0{,}8415 & 1 & 0 \\ -1{,}0371 & 0 & 1 \end{bmatrix}$$

5. B2 ÷ 0,0189 → B2

$$[H \mid I] = \begin{bmatrix} 1 & 0{,}8346 & 1{,}0344 \\ 0 & 1 & 0{,}9808 \\ 0 & 0{,}0190 & 0{,}0231 \end{bmatrix} \begin{bmatrix} 0{,}2019 & 0 & 0 \\ -44{,}615 & 53{,}0167 & 0 \\ -1{,}0371 & 0 & 1 \end{bmatrix}$$

6. B1 – (0,8346 x B2) → B1

$$[H \mid I] = \begin{bmatrix} 1 & 0 & 0{,}2102 \\ 0 & 1 & 0{,}9808 \\ 0 & 0{,}0190 & 0{,}0231 \end{bmatrix} \begin{bmatrix} 37{,}437 & -44{,}247 & 0 \\ -44{,}615 & 53{,}0167 & 0 \\ -1{,}0371 & 0 & 1 \end{bmatrix}$$

7. B3 – (0,0190 x B2) → B3

$$[H \mid I] = \begin{bmatrix} 1 & 0 & 0{,}2102 \\ 0 & 1 & 0{,}9808 \\ 0 & 0 & 0{,}0045 \end{bmatrix} \begin{bmatrix} 37{,}437 & -44{,}247 & 0 \\ -44{,}615 & 53{,}0167 & 0 \\ -0{,}1874 & -1{,}0097 & 1 \end{bmatrix}$$

8. B3 ÷ 0,0045 → B3

$$[H \mid I] = \begin{bmatrix} 1 & 0 & 0{,}2102 \\ 0 & 1 & 0{,}9808 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 37{,}437 & -44{,}247 & 0 \\ -44{,}615 & 53{,}0167 & 0 \\ -42{,}012 & -226{,}3 & 224{,}129 \end{bmatrix}$$

9. B1 – (0,2102 x B3) → B1

$$[H \mid I] = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0{,}9808 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 46{,}267 & 3{,}31657 & -47{,}103 \\ -44{,}615 & 53{,}0167 & 0 \\ -42{,}012 & -226{,}3 & 224{,}129 \end{bmatrix}$$

10. B2 – (0,9808 x R3) → B2

$$[H \mid I] = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 46{,}267 & 3{,}31657 & -47{,}103 \\ -3{,}4111 & 274{,}965 & -219{,}82 \\ -42{,}012 & -226{,}3 & 224{,}129 \end{bmatrix}$$

Hasil dari invers matriks secara keseluruhan adalah sebagai berikut:

Tabel 4. 15 Keluaran Matriks Inverse

$(H^T . H)^{-1}$	1	2	3
1	46,26748	3,316566	-47,1073
2	-3,41113	274,9654	-219,817
3	-42,012	-226,302	224,1287

Langkah 2.4: Menghitung Matriks *Moore-Penrose Pseudo Inverse* (H^+)

Perhitungan matriks ini dilakukan dengan mengalikan matriks inverse dengan matriks *transpose* H menggunakan persamaan 2.2 . Berikut contoh perhitungannya.

$$H^+ = (H^T x\, H)^{-1}\, x\, H^+$$
$$= (46{,}278\, x\, 0{,}52923) + (\, 3{,}31657\, x\, 0{,}54409) + (\, -47{,}107\, x\, 0{,}6608)$$
$$= -4{,}8132$$

Berikut adalah tabel hasil keluaran Mtriks *Moore-Penrose Pseudo Inverse*

Tabel 4. 16 Keluaran Matriks *Moore-Penrose Pseudo Inverse*

H^+	1	2	3	4	5	6
1	-4,81322	1,433149	1,694533	1,433193	-0,46897	1,694577
2	2,659448	3,91171	4,035311	3,911655	3,052719	4,035255
3	2,625118	-4,43502	-4,78817	-4,43502	-1,89794	-4,78817

H^+	7	8	9	10	11	12
1	1,169036	-4,54936	-0,87157	1,312774	1,574327	0,239922
2	3,788534	1,356998	-7,34626	-6,34488	-6,21842	-6,84758
3	-4,07958	3,425437	6,928755	4,003104	3,647473	5,448869

Langkah 3: Perhitungan *Output Weight* ($\widehat{\beta}$)

Perhitungan ini didapatkan dari perkalian matriks *Moore-Penrose Pseudo Inverse* dengan target pada data *training*. Hasil dari *output weight* ini akan digunakan pada proses *testing*, yang nantinya akan menghasilkan output dari sistem. Berikut contoh perhitungannya.

$$\hat{\beta} = H^{+} . Y$$

$$= (-4{,}81322 \text{ x } 0{,}28208) + (\ 1{,}433149 \text{ x } 0{,}65926) + (\ 1{,}694533 \text{ x } 0{,}69002) + (\ 1{,}433193 \text{ x } 0{,}65889\ ) + (\ -0{,}46897 \text{ x } 0{,}43913) + (1{,}694577 \text{ x } 0{,}6899) + (1{,}169036 \text{ x } 0{,}62738) + (\ -4{,}54936 \text{ x } 0) + (-0{,}87157 \text{ x } 0{,}6284) + (1{,}312774 \text{ x } 0{,}93137) + (1{,}574327 \text{ x } 1) + (\ 0{,}239922 \text{ x } 0{,}74133)$$

$$= 5{,}82445$$

Berikut matriks hasil keluaran *output weight*.

Tabel 4. 17 Matriks Keluaran *Output Weight*

$\hat{\beta}$	1
1	5,824451
2	-6,62845
3	0,663577

4.4.5 Proses *Testing*

Setelah dilakukan proses training, maka pada tahapan selanjutnya pada proses perhitungan ELM dilakukanlah proses *testing*. Proses testing dilakukan sama persis seperti proses *training*. Berikut langkah-langkah perhitungan pada proses *testing*.

Langkah 1: Perhitungan Keluaran Hidden Layer (*H*)

Seperti halnya pada proses *training*, keluaran hidden layer didapat dari perkalian input data training dengan input weight dan bias yang sama yang nantinya akan dipetakan juga menggunakan fungsi aktivasi sigmoid. Berikut ini contoh perhitungannya untuk data *testing*.

Langkah 1.1: Menghitung H_{init}

$$(X * W^{T}) + b = \big((0{,}81818\ x\ 0{,}6) + (1\ x\ 0{,}1) + (\ 1\ x\ -0{,}1) + (\ 0\ x\ 0\ ) + (\ 0{,}81379\ x\ 0{,}02)\big) + 0{,}12$$

$$= 0{,}62718$$

Berikut adalah matriks keluaran *Hidden Layer* (H) *r*

Tabel 4. 18 Matriks *Hidden Layer*

H_{init}	1	2	3
1	0,627185	0,063241	0,688695
2	0,739899	0,079545	0,699545

3	0,431065	0,039792	0,674792

Langkah 1.2: Menghitung Fungsi Aktivasi

$$H = \frac{1}{1 + \exp(-H_{init})} = \frac{1}{1 + \exp(-0{,}52718)} = 0{,}65185$$

Berikut hasil dari keluaran Fungsi Aktivasi dari nilai keluaran *Hidden Layer*

Tabel 4. 19 Matriks Keluaran *Hidden Layer*

H	1	2	3
1	0,651851	0,515805	0,665677
2	0,676974	0,519876	0,668087
3	0,606128	0,509947	0,662575

Langkah 2: Menghitung Hasil Prediksi ($\hat{Y}$)

Hasil prediksi pada metode ELM, pada proses *testing* didapat dari perkalian antara matriks keluaran *hidden layer* pada proses testing dengan nilai *output weight* pada proses *training*.

$$\hat{Y} = H * \hat{\beta}$$
$$= (0{,}651851\ x\ 5{,}82445) + (0{,}5158\ x - 6{,}6285) + (0{,}66568\ x\ 0{,}66358)$$
$$= 0{,}81941$$

Berikut hasil keluaran hasil prediksi.

Tabel 4. 20 Matriks Hasil Prediksi

$\hat{Y}$	1
1	0,819413
2	0,940357
3	0,589875

4.4.6 Denormalisasi

Denormalisasi data dilakukan untuk mengembalikan nilai yang sebenarnya pada hasil peramalan. Berikut contoh perhitungannya seperti persamaan 2.9.

$$x = (x' \times (\max - \min)) + min$$

$$x = (0{,}819413 \times (51979 - 11444)) + 11444$$

$$x = 44658{,}915$$

Hasil denormalisasi secara keseluruhan.

Tabel 4. 21 Hasil Denormalisasi Data

Denormalisasi Data	1
1	44658,91504
2	49561,35099
3	35354,57609

4.4.7 Perhitungan nilai evaluasi

Nilai evaluasi dilakukan dengan mencari nilai error Mean Absolute Percentage Error (MAPE). Berikut perhitungan nilai MAPE sesuai dengan persamaan 2.10 .

$$\text{MAPE} = \frac{1}{n}\sum_{i=1}^{n}\left|\left(\frac{y_i - \hat{y}_i}{y_i} \; x\; 100\right)\right|$$

$$= \left(ABS\left(\left(\frac{4318 - 44658{,}915}{4318}\right) \times 100\right)\right.$$

$$+ ABS\left(\left(\frac{5176 - 49561{,}351}{51769}\right) \times 100\right)$$

$$\left. + ABS\left(\left(\frac{34636 - 35354{,}5761}{34636}\right) \times 100\right) \div 3\right)$$

$$= 3{,}2531$$

Berikut nilai MAPE pada proses testing.

Tabel 4. 22 Nilai MAPE

Keterangan	Data Prediksi	Data Aktual
	44658,91504	43182
	49561,35099	51769

	35354,57609	34636
MAPE (%)	3,2531	

4.4.8 Inisialisasi *Pbest* dan *Gbest*

Nilai awal dari *Pbest* sama dengan nilai posisi awal partikel lalu untuk nilai awal *Gbest* dipilih dari *Pbest* yang memiliki nilai *fitness* terkecil. Pemilihan *Pbest* terkecil dikarenakan nilai *fitness* merupakan nilai *error* yang mana semakin kecil nilai *error* maka semakin baik sistem.

Tabel 4. 23 Inisialisasi *Pbest* Awal

NO	1	2	3	4	5	6	7	..	18	Fitness
Pbest1 (0)	0,6	0,1	-0,1	0	0,02	0	-0,1	...	0,74	3,2531
Pbest2 (0)	0,7	0,5	0,4	0,4	0,8	0,1	0,3	...	0,9	15,1347
Pbest3 (0)	0	-0,79	-0,73	0,34	0,61	0,46	0,14	...	0,35	11,6581

Tabel 4. 24 Inisialisasi *Gbest* Awal

NO	1	2	3	4	5	6	7	..	18	Fitness
Gbest1 (0)	0,6	0,1	-0,1	0	0,02	0	-0,1	...	0,74	3,2531

4.4.9 *Update* Kecepatan, Posisi, *Pbest* dan *Gbest* Partikel

Melakukan update kecepatan, posisi, Pbest dan Gbest partikel diperlukan beberapa parameter lain seperti bobot inersia, konstanta dan bilangan random. Berikut perhitungan update kecepatan, posisi, Pbest dan Gbest partikel pada iterasi 1.

- Update Kecepatan

  Pada saat melakukan update kecepatan, agar partikel tidak bergerak terlampau jauh, diperlukan pembatasan kecepatan minimum dan maksimum. Batasan minimum dan maksimum kecepatan dalam proses ini berdasarkan nilai minimum dan maksimum dari representasi solusi partikel. Pada inisialisasi awal sudah ditentukan rentan nilai partikel adalah [-1,1]. Untuk menentukan nilai minimum dan maksimum kecepatan diperlukan perhitungan seperti dibawah dengan beberapa parameter.

  - Nilai min dan max masing-masing partikel [-1,1]
  - Nilai k = 0,6
  - Nilai bobot inersia (w) = 0,5

- Nilai c1 dan c2 = 1
- Nilai r1 = 0,1
- Nilai r2 = 0,4

Untuk menentukan vmax dan vmin masing-maisng partikel diperlukan perhitungan seperti dibawah ini.

$$v\max = k\ \frac{(x\ max - x\ min)}{2}$$

$$v\max = 0{,}6\ \frac{(1-(-1))}{2}$$

$$v\max = 0{,}6 \quad v\min = -0{,}6$$

Untuk menghitung update kecepatan, sesuai dengan persamaan 2.5

$$v_{i,j}^{t+1} = w.v_{i,j}^{t} + c_1.r_1\left(Pbest_{i,j}^{t} - x_{i,j}^{t}\right) + c_2.r_2\left(Gbest_{g,j}^{t} - x_{i,j}^{t}\right)$$

$$= (0{,}5 \times 0) + (1 \times 0{,}1 \times (0{,}6 - 0{,}6) + (1 \times 0{,}4 \times (0{,}6$$

$$- 0{,}6))$$

$$= 0$$

Berikut nilai dari update kecepatan.

Tabel 4. 25 *Update* Kecepatan

NO	1	2	3	4	5	6	7	...	18
V1 (1)	0	0	0	0	0	0	0	...	0
V2 (1)	-0,04	-0,16	-0,2	-0,16	-0,312	-0,04	-0,16	...	-0,064
V3 (1)	0,24	0,356	0,252	-0,136	-0,236	-0,184	-0,096	...	0,156

- Update Posisi Partikel

  Perhitungan update posisi partikel adalah sebagai berikut sesusai dengan persamaan 2.7:

  $$x_{i,j}^{t+1} = x_{i.j}^{t} + v_{i.j}^{t+1}$$

  $$x_{i,j}^{t+1} = 0{,}6 + 0$$

$x_{i,j}^{t+1} = 0{,}6$

Berikut adalah hasil dari update Posisi Partikel

Tabel 4. 26 *Update* Posisi Partikel

NO	1	2	3	4	5	6	...	18	FITNESS
X1 (1)	0,6	0,1	-0,1	0	0,02	0	...	0,74	3,250847
X2 (1)	0,66	0,34	0,2	0,24	0,488	0,06	...	0,836	8,748661
X3 (1)	0,24	-0,434	-0,478	0,204	0,374	0,276	...	0,506	7,301222

- Update Pbest

Untuk update Pbest dilakukan dengan cara membandingkan Pbest sebelumnya dengan update Posisi dan nilai fitness yang paling kecil akan terpilih sebagai nilai Pbest dan fitness yang baru. Setelah dibandingkan nilai dari update posisi yang memiliki nilai fitness terkecil, maka dari itu nilai update posisi digunakan sebagai nilai dari Pbest yang baru.

Tabel 4. 27 *Update Pbest*

NO	1	2	3	4	5	6	...	18	FITNESS
Pbest1 (1)	0,6	0,1	-0,1	0	0,02	0	...	0,74	3,250847
Pbest2 (2)	0,66	0,34	0,2	0,24	0,488	0,06	...	0,836	8,748661
Pbest3 (3)	0,24	-0,434	-0,48	0,204	0,374	0,276	...	0,506	7,301222

- Update Gbest

Nilai dari Pbest terbaru dengan nilai fitness terendah akan menjadi nilai Gbest.

Tabel 4. 28 *Update Gbest*

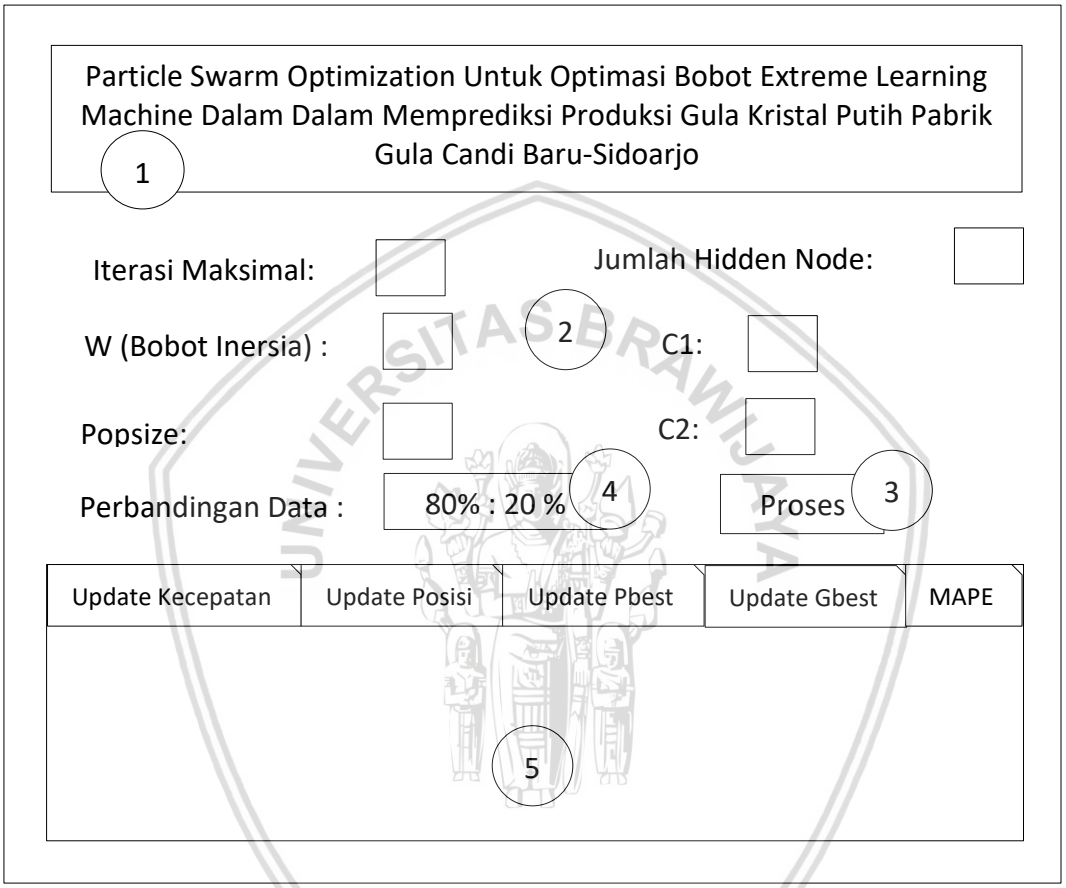
NO	1	2	3	4	5	6	...	18	FITNESS
Gebst(1)	0,6	0,1	-0,1	0	0,02	0	...	0,74	3,250847

4.5 Perancangan Antarmuka

Perancangan antarmuka adalah gambaran dari implementasi sistem dengan menggunakan metode PSO dan ELM untuk mengoptimasi input weight pada proses prediksi produksi gula. Implementasi sistem sendiri terdiri dari 2 halaman antarmuka, halaman antarmuka pertama proses ELM dan PSO lalu halaman antarmuka kedua adalah hasil prediksi dan *input weight* yang optimal.

4.5.1 Perancangan Antarmuka Proses ELM dan PSO

Antarmuka proses ELM dan PSO merupakan antarmuka yang menampilkan proses perhitungan dari metode ELM dan PSO. Antarmuka proses ELM dan PSO berisi parameter-parameter yang akan digunakan dalam perhitungan. Rancangan antarmuka proses perhitungan ELM dan PSO adalah sebagai berikut.



Gambar 4.21 Perancangan Antarmuka ELM dan PSO

Keterangan dari perancangan antarmuka ELM dan PSO:

1. Header sistem
2. Text field untuk mengisi parameter
3. Button proses untuk melakukan proses perhitungan
4. Pemilihan jumlah perbandingan data

5. Menampilkan Update Kecepatan, Update Posisi, Update Pbest, Update Gbest, MAPE.

4.6 Perancangan Skenario Pengujian

Pengujian yang dilakukan pada penelitian ini merupakan pengujian terkait metode ELM dan PSO yang digunakan untuk memprediksi produksi gula. Tujuan dari pengujian ini adalah untuk mengetahui kinerja dari metode yang sedang digunakan. Masing-masing pengujian akan dilakukan dengan 5 kali percobaan. Berikut skenario pengujian yang dilakukan pada penelitian:

1. Pengujian Jumlah Ukuran Populasi
2. Pengujian Jumlah Data Training
3. Pengujian Jumlah Hidden Neuron
4. Pengujian Jumlah Iterasi Maksimal
5. Pengujian Bobot Inersia

4.6.1 Pengujian Jumlah Ukuran Populasi

Pengujian jumlah ukuran populasi dilakukan untuk mengetahui jumlah ukuran populasi yang terbaik untuk menghasilkan nilai error yang optimal. Banyaknya ukuran populasi yang akan diuji dimulai dari 5 dan kelipatannya. Berikut rancangan pengujian untuk jumlah ukuran populasi.

Tabel 4. 29 Pengujian Jumlah Ukuran Populasi

Percobaan ke-i	5	10	15	20	...	100
1						
2						
3						
4						
5						

4.6.2 Pengujian Jumlah Data Training

Pengujian jumlah data training dilakukan untuk mengetahui pengaruh banyaknya data training terhadap algoritma ELM. Pengujian kali ini terdiri dari 5 jenis perbandingan data, yaitu 80%:20%, 70%:20%, 60%:20%, 50%:20%, dan 40%:20% untuk data training dan data testing. Berikut rancangan pengujian untuk jumlah data training.

Tabel 4. 30 Pengujian Jumlah Data *Training*

Percobaan ke-i	80%:20%	70%:20%	60%:20%	50%:20%	40%:20%
1					
2					
3					
4					
5					

4.6.3 Pengujian Jumlah Hidden Neuron

Pengujian jumlah hidden neuron bertujuan untuk mengetahui jumlah hidden neuron terbaik untuk mendapatkan hasil prediksi yang optimal. Pengujian jumlah hidden neuron dimulai dari angka 3 sampai 10. Berikut rancangan pengujian untuk jumlah hidden neuron.

Tabel 4. 31 Pengujian Jumlah *Hidden Neuron*

Percobaan ke-i	3	4	5	6	7	8	9	10
1								
2								
3								
4								
5								

4.6.4 Pengujian Bobot Inersia

Pengujian bobot inersia dilakukan untuk mengetahui bobot inersia terbaik untuk mendapatkan hasil prediksi yang optimal. Bobot inersia dimulai dari 0,5 sampai 1. Berikut rancangan pengujian bobot inersia.

Tabel 4. 32 Pengujian Bobot Inersia

Percobaan ke-i	0,5	0,6	0,7	0,8	0.9	1
1						
2						
3						
4						
5						

4.6.5 Pengujian Jumlah Iterasi Maksimal

Pengujian jumlah iterasi maksimum bertujuan untuk mengetahui jumlah iterasi maksimum terbaik untuk mendapatkan hasil preduksi yang optimal. Pengujian jumlah iterasi dimulai dari angka 50 berkelipatan hingga 250. Berikut rancangan pengujian untuk jumlah iterasi maksimum.

Tabel 4. 33 Pengujian Jumlah Iterasi Maksimum

Percobaan ke-i	50	100	150	200	250
1					
2					
3					
4					
5					

BAB 5 IMPLEMENTASI

Pada bab ini menjelaskan tentang implementasi dan antarmuka dari perancangan system yang telah dibuat untuk melakukan optimasi bobot pada metode *Extreme Learning Machine* dengan menggunakan *Particle Swarm Optimization*.

1.1 Implementasi Sistem

Berdasarkan perancangan yang sudah dibuat dan dijelaskan pada Bab sebelumnya, selanjutnya akan dijelaskan tentang implementasi dari sistem berdasarkan perancangan yang sudah dibuat. Implementasi sistem dilakukan dengan menggunakan Bahasa pemrograman java dengan editor Netbeans.

5.1.1 Implementasi Normalisasai Data

Pada proses ini untuk mengubah jangkauan data berubah menjadi nilai 0 sampai 1. Implementasi dari normalisasi data ditunjukkan dengan kode program seperti pada table 5.1 berikut.

Tabel 5. 1 Kode Program Implementasi Normalisasi Data

No	Kode Program
1	`public static void bacaData() {`
2	`for (int i = 0; i < min.length; i++) {`
3	`    min[i] = data_base[0][i];`
4	`    for (int j = 0; j < data_base.length; j++) {`
5	`         if (Double.valueOf(data_base[j][i]) < min[i]) {`
6	`             min[i] = Double.valueOf(data_base[j][i]);`
7	`         }`
8	`    }`
9	`}`
10	`for (int i = 0; i < max.length; i++) {`
11	`    for (int j = 0; j < data_base.length; j++) {`
12	`        if (Double.valueOf(data_base[j][i]) > max[i]) {`
13	`            max[i] = Double.valueOf(data_base[j][i]);`
14	`        }`
15	`    }`
16	`}`

```
17  }
18
19  static double[][] normalisasi(double[][] normal) {
20     double[][]normalisasi_data = new
21                     double[normal.length][normal[0].length];
22     for (int i = 0; i < normalisasi_data.length; i++) {
23          for (int j = 0; j < normalisasi_data[0].length;
24  j++) {
25              normalisasi_data[i][j] = (normal[i][j] -
26  min[j]) / (max[j] - min[j]);
27              }
28          }
29          return normalisasi_data;
30  }
```

Penjelasan untuk kode program implementasi normalisasi data adalah sebagai berikut:

1. Baris 2-6 : Proses untuk pencarian nilai minimum
2. Baris 10-13 : Proses untuk pencarian nilai maximum
3. Baris 19-29 : Proses untuk perhitungan nilai normalisasi data

5.1.2 Implementasi *Input Weight*

Pada proses ini untuk menentukan nilai dari *input weight* sendiri dilakukan secara *random* dengan *range* [-1,1]. *Input weight* berbentuk matriks dengan ordo yang sesuai dengan jumlah *hidden neuron* yang dimasukkan. Input weight pada proses ini didapat dari posisi awal pada metode PSO. Implementasi dari *input weight* ditunjukkan dengan kode program seperti pada tabel 5.2 berikut.

Tabel 5. 2 Kode Program Implementasi *Input Weight*

No	Kode program
1	`double[][] posisi_Awal(int popSize, int partikel`
2	`    ) {`
3	`        double x[][] = new double[popSize][partikel];`
4	`        double y[][] = new double[popSize][partikel];`
5	
6	`        for (int i = 0; i < x.length; i++) {`

```
7               for (int j = 0; j < x[0].length; j++) {
8                   x[i][j] = (double) ((Math.random() * (1 -
(-1))) + (-1));
9                   y[i][j] =
Double.valueOf(df.format(x[i][j]));
10                  System.out.print(y[i][j]+" ");
11              }
12              System.out.println("");
13          }
14          return y;
15      }
16
```

Penjelasan untuk kode program implementasi *input weight* adalah sebagai berikut:

1. Baris 1-4 : Inisialisasi variabel
2. Baris 6-16 : Proses pemberian nilai untuk posisi awal partikel secara random dengan range -1 sampai 1

5.1.3 Implementasi *Transpose* Matriks dan Perkalian Matriks

Pada proses ini matriks yang awalnya berupa baris dirubah menjadi kolom dan sebaliknya apabila awalnya berupa kolom maka dirubah menjadi baris. Sedangkan untuk proses perkalian matriks sendiri, merupakan sebuah proses perkalian yang membutuhkan 2 buah matriks. Implementasi dari *transpose* matriks dan perkalian matriks ditunjukkan dengan kode program seperti pada tabel 5.3 berikut.

Tabel 5. 3 Kode Program *Transpose* Matriks dan Perkalian Matriks

No	Kode Program
1	`static double[][] transpose(double[][] matriks) {`
2	`        double[][] matriksT = new`
3	`double[matriks[0].length][matriks.length];`
4	`        for (int i = 0; i < matriks.length; i++) {`
5	`            for (int j = 0; j < matriks[0].length; j++) {`
6	`                matriksT[j][i] = matriks[i][j];`
7	`            }`
8	`        }`
9	`        return matriksT;`
10	`    }`

```
11
12  static double[][] xTest(double[][] normal, double[][]
    bobot) {
13          double[][] xTest = new
14  double[normal.length][bobot[0].length];
15          for (int i = 0; i < xTest.length; i++) {
16              for (int j = 0; j < xTest[0].length; j++) {
17                  for (int k = 0; k < normal[0].length - 1;
18  k++) {
19                      xTest[i][j] += (normal[i][k] *
    bobot[k][j]);
20                  }
21              }
22          }
23          return xTest;
24      }
```

Penjelasan untuk kode program implementasi transpose matriks dan perkalian matriks adalah sebagai berikut:

1. Baris 1-2 : Inisialisasi variabel matriksT
2. Baris 4-10 : Proses untuk *transpose* matriks
3. Baris 12-14 : Inisialisasi variabel kali
4. Baris 15-24 : Proses untuk perkalian matriks

5.1.4 Implementasi Perhitungan *Output Hidden Layer*

Proses perhitungan ini dilakukan pada saat proses pelatihan dan pengujian pada metode ELM. Implementasi dari perhitungan *output hidden layer* ditunjukkan dengan kode program seperti pada tabel 5.4 berikut.

Tabel 5. 4 Kode Program *Output Hidden Layer*

No	Kode program
1	`static double[][] H_Init(double[][] xTest, double[] bias) {`
2	`        double[][] H_Init = new`
3	`double[xTest.length][xTest[0].length];`
4	`        for (int i = 0; i < H_Init.length; i++) {`
5	`            for (int j = 0; j < H_Init[0].length; j++) {`
6	`                H_Init[i][j] = xTest[i][j] + bias[j];`

```
7               }
8           }
9           return H_Init;
10      }
11
12  static double[][] sigmoid(double[][] H_Init) {
13          double[][] sigmoid = new
14  double[H_Init.length][H_Init[0].length];
15          for (int i = 0; i < H_Init.length; i++) {
16              for (int j = 0; j < H_Init[0].length; j++) {
17                  sigmoid[i][j] = 1 / (1 + Math.exp(-
18  H_Init[i][j]));
19              }
20          }
21          return sigmoid;
22      }
```

Penjelasan untuk kode program *ouput hidden layer* adalah sebagai berikut:

5. Baris 1-2 : Inisialisasi variabel H_Init
6. Baris 4-10 : Proses perhitungan *output hidden layer,* dengan cara mengalikan *input* dan bobot *transpose*
7. Baris 12-14 : Inisialisasi variabel sigmoid
8. Baris 15-21 : Proses perhitungan nilai aktivasi dari hasil keluaran H_init

5.1.5 Implementasi Perhitungan Moore-Penrose Pseudo Invers

Pada proses perhitungan ini memiliki beberapa tahapan dalam penyelesaiannya, diantaranya *transpose* dari output sigmoid (H^T), perkalian matriks antara H^T dengan H, invers dari perkalian matriks, dan yang terakhir perkalian hasil invers dengan H^T. Implementasi dari perhitungan *moore-penrose pseudo invers* ditunjukkan dengan kode program seperti pada tabel 5.5 berikut.

Tabel 5. 5 Kode Program *Moonre-Penrose Pseudo Invers*

No	Kode Program
1	`static double[][] moonre_penrose(double[][] sigmoid) {`
2	`        double[][] sigmoid_T = new`
3	`double[sigmoid[0].length][sigmoid.length];`

```
4           sigmoid_T = transpose(sigmoid);
5           double[][] sigmoidT_sigmoid = kaliMatriks(sigmoid_T,
6   sigmoid);
7           double[][] sigmoidT_sigmoid_invers = new
8   double[sigmoidT_sigmoid.length][sigmoidT_sigmoid[0].length];
9           Matrix A = new Matrix(sigmoidT_sigmoid);
10          Matrix B = A.inverse();
11          for (int i = 0; i < B.getRowDimension(); i++) {
12              for (int j = 0; j < B.getColumnDimension(); j++)
13  {
14                  sigmoidT_sigmoid_invers[i][j] = B.get(i, j);
15              }
16          }
17          double[][] moonre_penrose =
18  kaliMatriks(sigmoidT_sigmoid_invers, sigmoid_T);
19          return moonre_penrose;
20      }
```

Penjelasan untuk kode program *ouput hidden layer* adalah sebagai berikut:

9. Baris 1-9 : Inisialisasi variabel sigmoid_T, sigmoidT_sigmoid, sigmoidT_sigmoid_invers
10. Baris 10-15 : Proses perhitungan *invers* dari hasil perkalian H^T dengan H
11. Baris 16-18 : Proses perhitungan *moore-penrose pseudo invers* dengan cara mengalikan antara hasil *invers* dengan H^T

5.1.6 Implementasi Perhitungan *Output Weight*

Pada proses perhitungannya didapat dari hasil perkalian antara *moore-penrose pseudo invers* dengan target data. Implementasi dari perhitungan *output weight* ditunjukkan dengan kode program seperti pada tabel 5.6 berikut.

Tabel 5. 6 Kode Program *Output Weight*

No	Kode Program
1 2	`static double[] output_weight(double[][] moonre_penrose, double[][] dataLatih) {`

```
3           double[] output_weight = new
4   double[moonre_penrose.length];
5           for (int i = 0; i < output_weight.length; i++) {
6               for (int j = 0; j < moonre_penrose[0].length;
7   j++) {
8                   output_weight[i] += (moonre_penrose[i][j] *
9   dataLatih[j][5]);
10              }
11          }
12          return output_weight;
13      }
```

Penjelasan untuk kode program *ouput weight* adalah sebagai berikut:

12. Baris 1-4 : Inisialisasi variabel output_weight
13. Baris 5-12 : Proses perhitungan *output weight*

5.1.7 Implementasi Perhitungan Nilai Prediksi

Pada proses ini untuk memperoleh hasil dari prediksi. Implementasi dari perhitungan nilai prediksi ditunjukkan dengan kode program seperti pada tabel 5.7 berikut.

Tabel 5. 7 Kode Program Nilai Prediksi

No	Kode Program
1	`static double[] Y_Prediksi(double[][] sigmoid_Uji, double[]`
2	`bobot_Beta) {`
3	`        double[] Y_Prediksi = new`
4	`double[sigmoid_Uji.length];`
5	`        for (int i = 0; i < Y_Prediksi.length; i++) {`
6	`            for (int j = 0; j < bobot_Beta.length; j++) {`
7	`                Y_Prediksi[i] += (sigmoid_Uji[i][j] *`
8	`bobot_Beta[j]);`
9	`            }`
10	`        }`
11	`        return Y_Prediksi;`
12	`    }`

Penjelasan untuk kode program perhitungan nilai prediksi adalah sebagai berikut:

14. Baris 1-4 : Inisialisasi variabel Y_Prediksi
15. Baris 5-11 : Proses perhitungan nilai prediksi dengan melakukan perkalian matriks sigmoid dengan *output weight*

5.1.8 Implementasi Denormalisasi

Pada proses ini mengembalikan nilai yang sudah dinormalisasi sebelumnya. Implementasi dari proses denormalisasi ditunjukkan dengan kode program seperti pada tabel 5.8 berikut.

Tabel 5. 8 Kode Program Denormalisasi

No	Kode Program
1	`static double[] denorm(double[] Y_Prediksi) {`
2	`denorm = new double[Y_Prediksi.length];`
3	`for (int i = 0; i < Y_Prediksi.length; i++) {`
4	`denorm[i] += ((Y_Prediksi[i] * (max[5] -`
5	`min[5])) + min[5]);`
6	`}`
7	`return denorm;`
8	`}`

Penjelasan untuk kode program proses denormalisasi adalah sebagai berikut:

16. Baris 1-2 : Inisialisasi variabel denorm
17. Baris 3-8 : Proses perhitungan denormalisasi data

5.1.9 Implementasi Perhitungan *Mean Absolute Precentage Error*

Proses perhitungan MAPE yang merupakan salah satu cara untuk menghitung nilai evaluasi. Implementasi dari proses MAPE ditunjukkan dengan kode program seperti pada tabel 5.9 berikut.

Tabel 5. 9 Kode Program MAPE

No	Kode Program
1	`static double MAPE(double[] denorm, double[][]`
2	`dataTesting) {`
3	`double nilai_mape = 0;`

```
4           for (int i = 0; i < denorm.length; i++) {
5               nilai_mape += Math.abs(((denorm[i] -
6   dataTesting[i][5]) / dataTesting[i][5]) * 100);
7           }
8           nilai_mape = nilai_mape / denorm.length;
9           return nilai_mape;
10      }
```

Penjelasan untuk kode program proses perhitungan MAPE adalah sebagai berikut:

18. Baris 1-3 : Inisialisasi variabel nilai_mape
19. Baris 4-10 : Proses perhitungan *mean absolute percentage error*

5.1.10 Implementasi Inisialisasi Kecepatan Awal Partikel

Proses inisialisasi awal kecepatan awal partikel dilakukan dengan memberi nilai 0. Implementasi dari proses inisialisasi kecepatan awal partikel ditunjukkan dengan kode program seperti pada tabel 5.10 berikut.

Tabel 5. 10 Kode Program Kecepatan Awal Partikel

No	Kode Program
1	`public double[][] v_Awal(int popSize, int partikel) {`
2	`        double v[][] = new double[popSize][partikel];`
3	`        for (int i = 0; i < popSize; i++) {`
4	`            for (int j = 0; j < partikel; j++) {`
5	`                v[i][j] = 0;`
6	`            }`
7	`        }`
8	`        return v;`
9	`    }`

Penjelasan untuk kode program proses inisialisasi kecepatan awal partikel adalah sebagai berikut:

20. Baris 1-2 : Inisialisasi variabel matriks v
21. Baris 3-9 : Proses pemberian nilai 0 pada kecepatan awal partikel

5.1.11 Implementasi Inisialisai Posisi Awal Partikel

Proses inisialisasi awal posisi awal partikel dilakukan secara random dengan panjang yang sesuai banyaknya *input* data dengan *range* nilai [-1,1]. Implementasi dari proses inisialisasi posisi awal partikel ditunjukkan dengan kode program seperti pada tabel 5.11 berikut.

Tabel 5. 11 Kode Program Inisialisasi Posisi Awal Partikel

No	Kode Program
1	`double[][] posisi_Awal(int popSize, int partikel`
2	`    ) {`
3	`        double x[][] = new double[popSize][partikel];`
4	`        double y[][] = new double[popSize][partikel];`
5	
6	`        for (int i = 0; i < x.length; i++) {`
7	`            for (int j = 0; j < x[0].length; j++) {`
8	`  x[i][j] = (double) ((Math.random() * (1 - (-1))) + (-1));`
9	`        y[i][j] = Double.valueOf(df.format(x[i][j]));`
10	`                System.out.print(y[i][j]+" ");`
11	`            }`
12	`            System.out.println("");`
13	`        }`
14	`        return y;`
15	`    }`

Penjelasan untuk kode program proses inisialisasi posisi awal partikel adalah sebagai berikut:

22. Baris 1-4 : Inisialisasi variabel
23. Baris 6-15 : Proses pemberian nilai untuk posisi awal partikel secara random dengan range -1 sampai 1

5.1.12 Implementasi Perhitungan Nilai *Fitness*

Pada proses ini nilai fitness didapat dari nilai MAPE dari perhitungan ELM. Implementasi dari proses perhitungan nilai *fitness* ditunjukkan dengan kode program seperti pada tabel 5.12 berikut.

Tabel 5. 12 Kode Program Perhitungan ELM

No	Kode Program
1	`static double ELM(double[] p, double[][] dataUji,`
2	`double[][] dataLatih) {`
3	
4	`norm_Data_Latih = normalisasi(dataLatih);`
5	`norm_Data_Uji = normalisasi(dataUji);`
6	
7	`for (int i = 0; i < bobot.length; i++) {`
8	`for (int j = 0; j < bobot[0].length; j++) {`
9	`bobot[i][j] = p[j + ((5 % (i + 1)) * 5)];`
10	`}`
11	`bias[i] = p[i + (bobot.length * 5)];`
12	`}`
13	`bobotT = transpose(bobot);`
14	`xTest = xTest(normalisasi(dataLatih), bobotT);`
15	`H_Init = H_Init(xTest, bias);`
16	`sigmoid = sigmoid(H_Init);`
17	`moonre_penrose = moonre_penrose(sigmoid);`
18	`output_weight = output_weight(moonre_penrose,`
19	`norm_Data_Latih);`
20	`xTest_Uji = xTest(normalisasi(dataUji), bobotT);`
21	`H_Init_Uji = H_Init(xTest_Uji, bias);`
22	`sigmoid_Uji = sigmoid(H_Init_Uji);`
23	`Y_Prediksi = Y_Prediksi(sigmoid_Uji,`
24	`output_weight);`
25	`denorm = denorm(Y_Prediksi);`
26	`MAPE = MAPE(denorm, dataUji);`
27	
28	`return MAPE;`
29	`}`

Penjelasan untuk kode program proses perhitungan ELM adalah sebagai berikut:

24. Baris 1-5 : inisialisasi variabel
25. Baris 7-12 : proses inisialisasi nilai bias

26. Baris 13-29 : pemanggilan fungsi-fungsi ELM

Tabel 5. 13 Kode Program Nilai *Fitness*

No	Kode Program
1	`double Fitness(double[] posisi_Awal) {`
2	`        double Fitness = 0;`
3	`        Fitness = ELM.ELM(posisi_Awal, ElmManual.data_Uji,`
4	`ElmManual.data_Latih);`
5	`        return Fitness;`
6	`    }`

Penjelasan untuk kode program proses perhitungan nilai *fitness* adalah sebagai berikut:

27. Baris 1-2 : Inisialisasi variabel Fitness
28. Baris 3-6 : Pemanggilan nilai ELM pada *class* ElmManual

5.1.13 Implementasi Inisialisasi Pbest Awal Partikel

Proses inisialisasi *pBest* awal partikel , dimana nilai *pBest* awal partikel sama dengan nilai dari posisi awal partikel. Implementasi dari proses inisialisasi *pBest* awal partikel ditunjukkan dengan kode program seperti pada tabel 5.14 berikut.

Tabel 5. 14 Kode Program Inisialisai *Pbest* Awal

No	Kode Program
1	`double[][] pBest_Awal(double[][] a) {`
2	`        double pBest_Awal[][] = new`
3	`double[a.length][a[0].length];`
4	`        for (int i = 0; i < pBest_Awal.length; i++) {`
5	`           for (int j = 0; j < pBest_Awal[0].length; j++) {`
6	`                pBest_Awal[i][j] = a[i][j];`
7	`            }`
8	`        }`
9	`        return pBest_Awal;`
10	`    }`

Penjelasan untuk kode program proses inisialisasi *Pbest* awal adalah sebagai berikut:

29. Baris 1-3 : Inisialisasi variabel pBest_Awal

30. Baris 4-10 : Proses inisialisasi Pbest awal

5.1.14 Implementasi Inisialisasi Gbest Awal Partikel

Proses inisialisasi *gBest* awal partikel,dimana nilai *gBest* awal partikel didapatkan dari nilai *fitness* terkecil pada *pBest* partikel. Implementasi dari proses inisialisasi *gBest* awal partikel ditunjukkan dengan kode program seperti pada tabel 5.15 berikut.

Tabel 5. 15 Kode Program Inisialisasi *Gbest* Awal

No	Kode Program
1	`double[] gBest(int nPartikel, double[][] pBest, double[]`
2	`fitness) {`
3	`double[] gBest = new double[nPartikel + 1];`
4	`int indexMinimal = 0;`
5	`min = fitness[0];`
6	`for (int i = 1; i < fitness.length; i++) {`
7	`if (fitness[i] < min) {`
8	`min = fitness[i];`
9	`indexMinimal = i;`
10	`}`
11	`}`
12	`for (int i = 0; i < gBest.length; i++) {`
13	`if (i == gBest.length - 1) {`
14	`gBest[gBest.length - 1] =`
15	`fitness[indexMinimal];`
16	`} else {`
17	`gBest[i] = pBest[indexMinimal][i];`
18	`}`
19	`}`
20	`return gBest;`
21	`}`

Penjelasan untuk kode program proses inisialisasi *Gbest* awal adalah sebagai berikut:

31. Baris 1-5 : Inisialisasi variabel gBest, indexMinimal dan min

32. Baris 6-21 : Proses pencarian nilai *fitness* terkecil dari *pbest*

5.1.15 Implementasi Perhitungan Update Kecepatan Partikel

Perhitungan *update* kecepatan partikel yang mana perhitungan ini sesuai dengan persamaan 2.5, dimana nilai dar r dibangkitkan secara acak dengan range [0,1]. Implementasi dari proses perhitungan update kecepatan partikel ditunjukkan dengan kode program seperti pada tabel 5.16 berikut.

Tabel 5. 16 Kode Program *Update* Kecepaatan Partikel

No	Kode Program
1	`double[][] updateKecepatan(int popSize, int partikel,`
2	`double[][] v, double w, double c1, double r1, double`
3	`pBest[][], double[] gBest, double[][] x, double c2, double`
4	`r2, double vMax) {`
5	`double updateV[][] = new double[popSize][partikel];`
6	`for (int i = 0; i < popSize; i++) {`
7	`for (int j = 0; j < partikel; j++) {`
8	`updateV[i][j] = (w * v[i][j]) + (c1 * r1 *`
9	`(pBest[i][j] - x[i][j])) + (c2 * r2 * (gBest[j] –`
10	`x[i][j]));`
11	`if (updateV[i][j] > vMax) {`
12	`updateV[i][j] = vMax;`
13	`} else if (updateV[i][j] < -vMax) {`
14	`updateV[i][j] = -vMax;`
15	`}`
16	`}`
17	`}`
18	`return updateV;`
19	`}`

Penjelasan untuk kode program proses *update* kecepatan partikel adalah sebagai berikut:

33. Baris 1-5 : Pembuatan method updateKecepatan dengan parameter

34. Baris 6-16 : Proses perhitungan *update* kecepatan

5.1.16 Implementasi Update Posisi

Perhitungan *update* posisi partikel yang mana perhitungan ini sesuai dengan persamaan 2.5. Implementasi dari proses perhitungan *update* posisi partikel ditunjukkan dengan kode program seperti pada tabel 5.17 berikut.

Tabel 5. 17 Kode Program Update Posisi

No	Kode Program
1	`double[][] posisi(int iterasi, double[][] v) {`
2	`if (iterasi == 0) {`
3	`a = posisi_Awal(v.length, v[0].length);`
4	`for (int i = 0; i < a.length; i++) {`
5	`partikel_ = konversi_Array(a, i);`
6	`fitness[i] = Fitness(partikel_);`
7	`temp_fitness_N[i] = fitness[i];`
8	`x[i][x[0].length - 1] = fitness[i];`
9	`}`
10	`for (int i = 0; i < a.length; i++) {`
11	`for (int j = 0; j < a[0].length; j++) {`
12	`x[i][j] = a[i][j];`
13	`}`
14	`}`
15	`} else {`
16	`for (int i = 0; i < a.length; i++) {//`
17	`for (int j = 0; j < a[0].length; j++) {`
18	`temp_x[i][j] = x[i][j];`
19	`x[i][j] = x[i][j] + v[i][j];`
20	`}`
21	`partikel_ = tes.konversi_Array(x, i);//gak`
22	`fitness[i] = tes.Fitness(partikel_);`
23	`}`
24	`for (int i = 0; i < x.length; i++) {//`
25	`x[i][x[0].length - 1] = fitness[i];`
26	`}`
27	`}`
28	`return x;`
29	`}`

Penjelasan untuk kode program *update* posisi partikel adalah sebagai berikut:

35. Baris 2-13 : Deklaras posisi awal partikel
36. Baris 14-29 : Proses update posisi awal partikel

5.1.17 Implementasi *Update Pbest*

Proses *update pBest* partikel yang mana proses *update* pBest membandingkan nilai *pBest* sebelumnya dengan nilai hasil dari *update* posisi . Implementasi dari proses perhitungan *update pBest* partikel ditunjukkan dengan kode program seperti pada tabel 5.18 berikut.

Tabel 5. 18 Kode Program *Update Pbest*

No	Kode Program
1	`double[][] updatePBest(int partikel, double[][] pBest_N,`
2	`double[][] pBest_B, double[] fitness_N, double[] fitness_B)`
3	`{`
4	`double[][] pBest = new`
5	`double[pBest_N.length][pBest_N[0].length];`
6	`for (int i = 0; i < pBest.length; i++) {`
7	`for (int j = 0; j < pBest[0].length; j++) {`
8	`if (fitness_B[i] < fitness_N[i]) {`
9	`pBest[i][j] = pBest_N[i][j];`
10	`} else if (fitness_B[i] > fitness_N[i]) {`
11	`pBest[i][j] = pBest_B[i][j];`
12	`} else if (fitness_B[i] == fitness_N[i]) {`
13	`pBest[i][j] = pBest_B[i][j];`
14	`}`
15	`}`
16	`}`
17	`return pBest;`
18	`}`

Penjelasan untuk kode program *update pBest* partikel adalah sebagai berikut:

1. Baris 1-4 : Inisialisasi variabel
2. Baris 5-18 : Proses *update pBest* partikel

5.1.18 Implementasi *Update Gbest*

Proses *update gBest* sama seperti inisialisasi *gBest* awal yakni memilih nilai *fitness* dari *pBest* yang terkecil.

Tabel 5. 19 Kode Program Update Gbest

No	Kode Program
1	`double[] gBest(int iterasi, int partikel) {`
2	`if (iterasi == 0) {`
3	`gBest = pso.gBest(partikel, x, fitness);`
4	`} else {`
5	`gBest = pso.gBest(partikel, pBest, fitness);`
6	`}`
7	`return gBest;`
8	`}`

Penjelasan untuk kode program *update pBest* partikel adalah sebagai berikut:

1. Baris 1-3 : Pemanggilan method *gBest*
2. Baris 4-8 : Proses pencarian nilai fitness

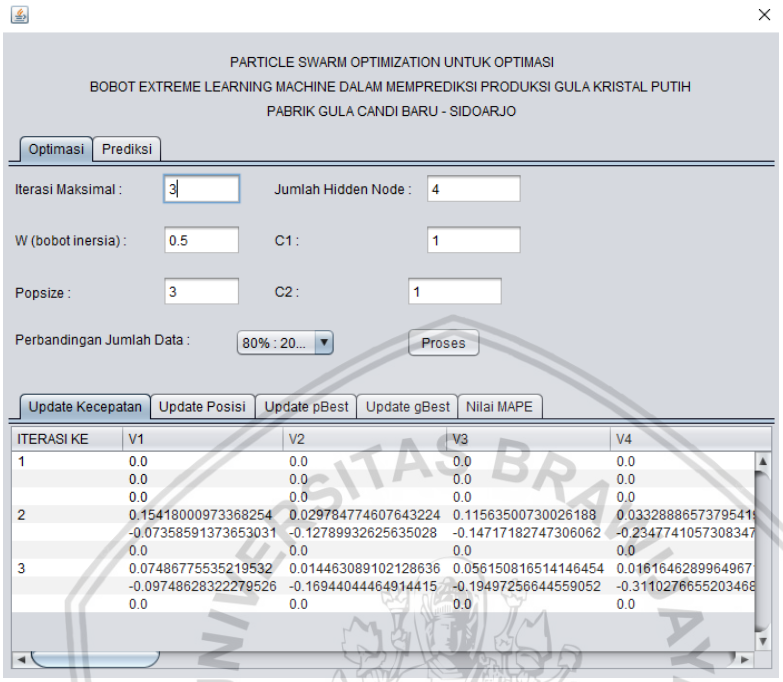
1.2 Implementasi Antarmuka

Implementasi antarmuka merupakan implementasi yang dibuat berdasarkan perancangan antarmuka yang sudah dibuat pada bab sebelumnya. Antarmuka yang digunakan pada penelitian ini terdiri dari 2 halaman, 1 halaman antarmuka utama dan 1 halaman antarmuka pendukung.

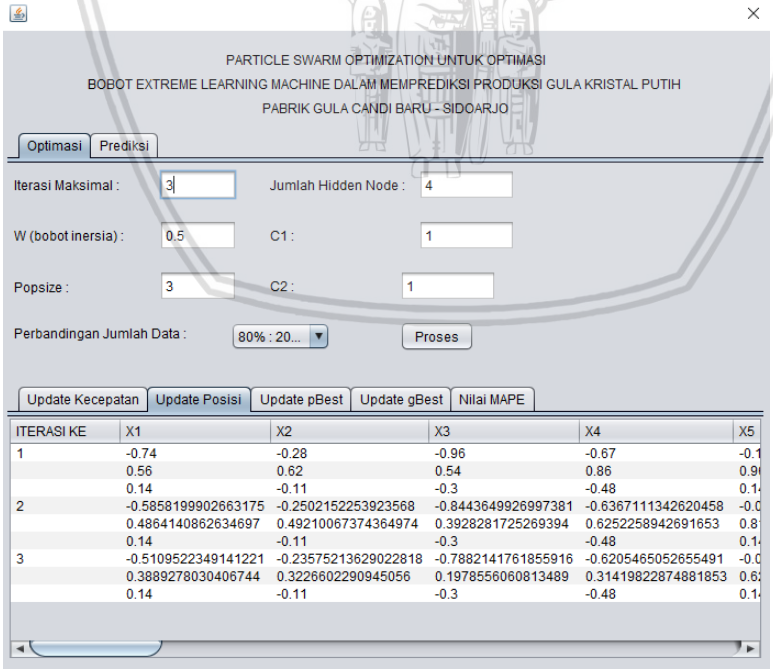
5.2.1 Implementasi Antarmuka Proses PSO dan ELM

Implementasi antarmuka proses ELM dan PSO merupakan halaman antarmuka utama yang digunakan untuk melakukan proses optimasi bobot. Halaman ini berisi parameter-parameter yang dibutuhkan PSO dan ELM dalam melakukan optimasi bobot. Parameter yang ada adalah iterasi maksimal, jumlah hidden neuron, bobot inersia (w), konstanta (c1 dan c2) dan jumlah popsize. Selain itu terdapat tombol button proses yang dapat ditekan oleh *user* untuk melakukan proses perhitungan bobot. Hasil yang ditampilkan pada halaman utama antarmuka ini terdapat 5 tampilan, yakni tampilan yang pertama update kecepatan, yang kedua update posisi,

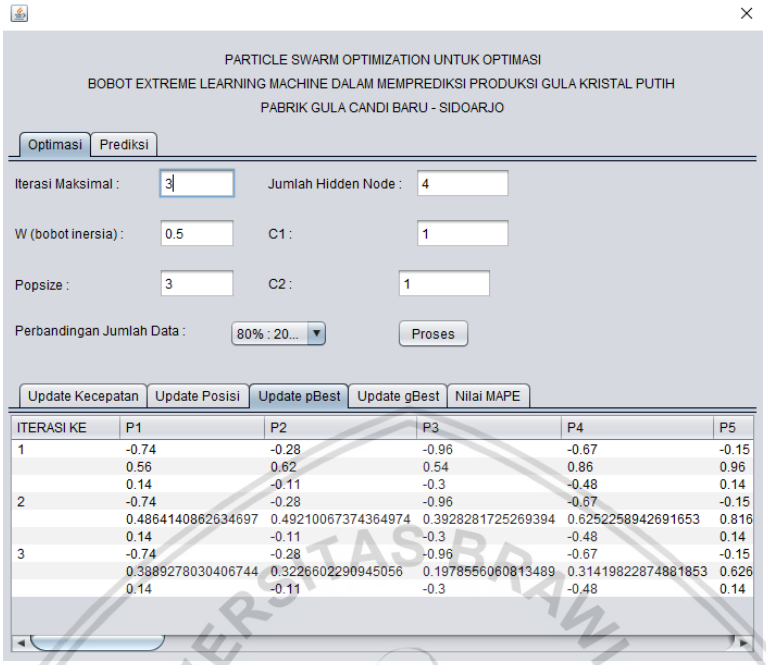
yang ketiga update pBest, keempat update gBest, dan terakhir nilai MAPE. Implementasi antarmuka proses PSO dan ELM ditunjukkan pada gambar 5.1 berikut.



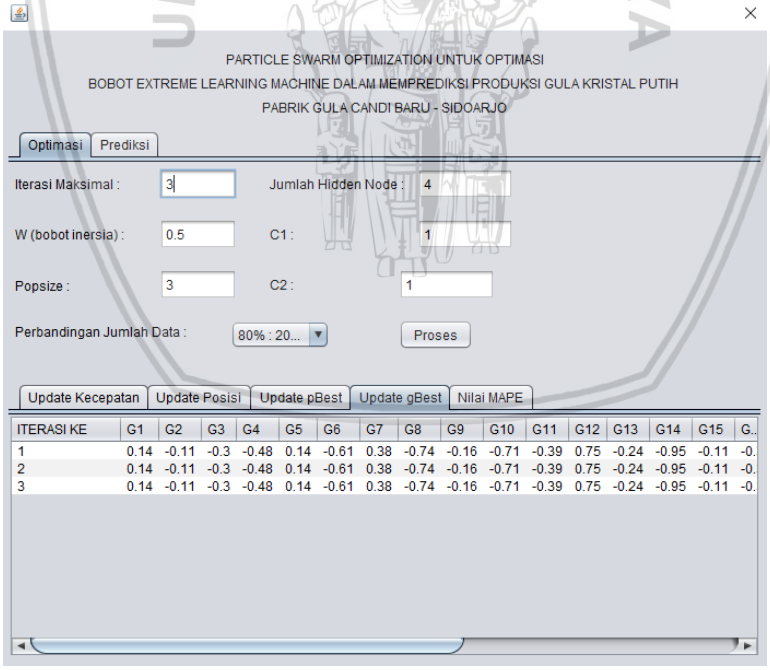
Gambar 5. 1 Implementasi Antarmuka *Update* Kecepatan



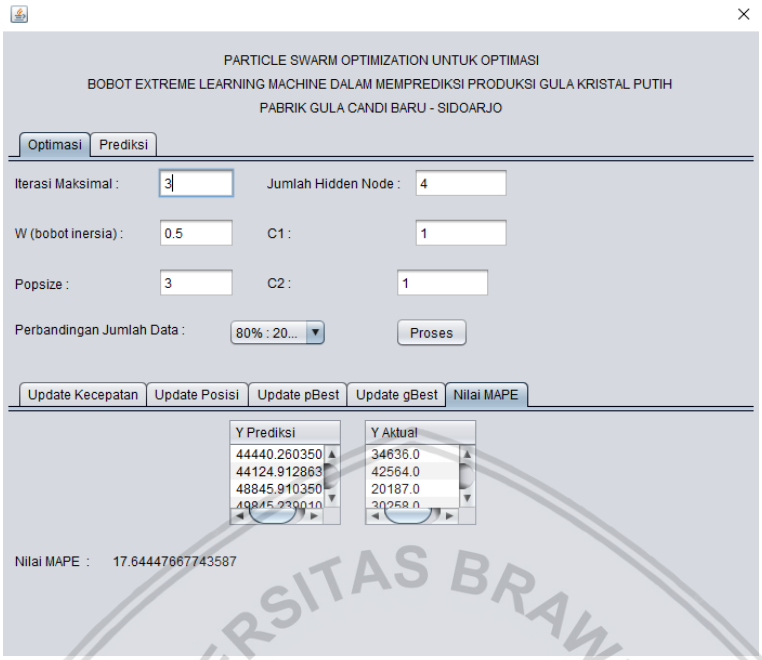
Gambar 5. 2 Implementasi Antarmuka *Update* Posisi



Gambar 5. 3 Implementasi Antarmuka *Update Pbest*



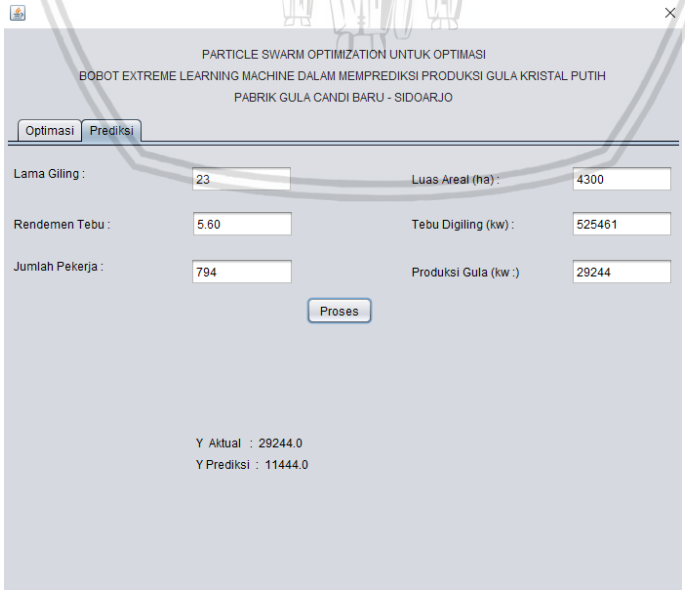
Gambar 5. 4 Implementasi Antarmuka *Update Gbest*



Gambar 5. 5 Implentasi Antarmuka Nilai MAPE

5.2.2 Implementasi Antarmuka Prediksi

Pada tampilan antarmuka prediksi menampilkan nilai Y aktual dan Y prediksi pada proses ELM, dimana nilai *input weight* nya didapat dari nilai *gbest* yang memiliki nilai *fitness* terkecil, dengan masukan variabel yang dibutuhkan dari *user*. Implementasi antarmuka prediksi ditunjukkan pada gambar 5.6 seperti berikut.



Gambar 5. 6 Implementasi Antarmuka Prediksi

BAB 6 PENGUJIAN DAN ANALISIS

Pada bab ini menjelaskan tentang pengujian dan analisis sistem yang telah diimplementasikan. Pengujian sistem dilakukan berdasarkan skenario pengujian yang sudah dibahas pada bab sebelumnya.

1.1 Pengujian

Pengujian yang dilakukan berdasarkan scenario pengujian adalah sebagai berikut:

1. Pengujian jumlah ukuran populasi
2. Pengujian jumlah data *training*
3. Pengujian jumlah *hidden neuron*
4. Pengujian jumlah iterasi maksimal
5. Pengujian bobot inersia

Setiap pengujian akan dilakukan sebanyak 10 kali percobaan dan kemudian dievaluasi berdasarkan rata-rata *fitness* yang dihasilkan.

1.2 Hasil dan Analisis Pengujian

1.2.1 Pengujian Jumlah Ukuran Populasi

Pengujian berdasarkan banyaknya jumlah ukuran populasi yang bertujuan untuk mengetahui pengaruh jumlah ukuran populasi yang terbaik untuk menghasilkan nilai *error* yang optimal. Banyaknya ukuran populasi yang akan diuji mulai dari 10 dan kelipatannya sampai 50. Pada masing-masing variasi ukuran populasi akan dilakukan percobaan sebanyak 10 kali percobaan. Pada perhitungan PSO dan ELM banyak melibatkan variabel yang dibangkitkan secara acak, sehingga mengakibatkan nilai yang dihasilkan berbeda pada setiap kali *running* program.

Nilai parameter lain yang digunakan pada pengujian ini adalah sebagai berikut:

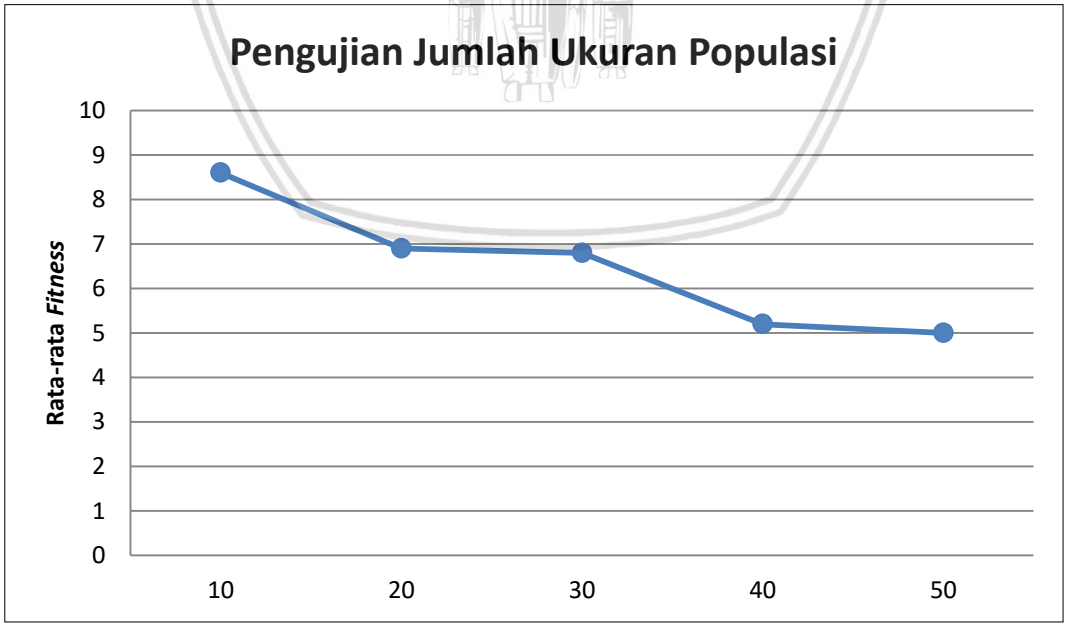
a. Jumlah Iterasi : 100
b. Bobot Inersia : 0.5
c. Nilai Konstanta 1 (c1) : 1
d. Nilai Konstanta 2 (c2) : 1
e. Jumlah *Hidden Neuron* : 3
f. Banyak data *training* dan data *testing* : 80% : 20%

Hasil pengujian untuk jumlah ukuran populasi dapat dilihat pada Tabel 6.1 berikut:

Tabel 6. 1 Pengujian Jumlah Ukuran Populasi

Percobaan Ke – i	Nilai *fitness* pada pengujian jumlah ukuran populasi				
	10	20	30	40	50
1	5.58	10.83	12.36	6.17	3.47
2	16.51	2.31	7.17	14.02	2.77
3	14.06	4.23	3.03	4.48	3.31
4	5.73	4.18	4.14	1.1	4.14
5	17.34	9.97	3.67	4.38	3.38
6	8.41	3.3	4.7	2.27	13.1
7	3.47	4.31	2.81	2.11	8.3
8	4.57	19.96	15.85	8.32	3.08
9	5.2	8.8	2.92	4.57	3.24
10	5.14	2.01	11.9	5.25	5.32
Rata-rata *fitness*	8.6	6.9	6.8	5.2	5.0

Hasil pengujian jumlah ukuran populasi secara keseluruhan akan ditunjukkan pada Lampiran C. Berdasarkan pengujian jumlah ukuran populasi didapatkan nilai rata-rata MAPE terbaik dari tiap variasi ukuran populasi. Grafik pengujian hasil ukuran populasi ditunjukkan pada Gambar 6.1.



Gambar 6. 1 Pengujian Jumlah Ukuran Populasi

Berdasarkan grafik hasil pengujian jumlah ukuran populasi di atas, ukuran populasi 50 mengasilkan rata-rata nilai *fitness* terkecil, yaitu 5.0. Dari gambar diatas dapat disimpulkan apabila jumlah ukuran populasi semakin besar maka nilai *fitness* semakin kecil.

1.2.2 Pengujian Jumlah Iterasi Maksimal

Pengujian jumlah iterasi maksimum bertujuan untuk mengetahui jumlah iterasi terbaik untuk mendapatkan hasil prediksi yang optimal. Pengujian jumlah iterasi dimulai dari angka kelipatan 50 hingga 250. Pada masing-masing variasi jumlah iterasi dilakukan percobaan sebanyak 10 kali percobaan. Pada perhitungan PSO dan ELM banyak melibatkan variabel yang dibangkitkan secara acak, sehingga mengakibatkan nilai yang dihasilkan berbeda pada setiap kali *running* program.

Nilai parameter lain yang digunakan pada pengujian ini adalah sebagai berikut:

a. Bobot Inersia : 0.5
b. Nilai Konstanta 1 (c1) : 1
c. Nilai Konstanta 2 (c2) : 1
d. Banyak data training dan testing : 80% : 20%
e. Jumlah ukuran populasi : 50
f. Jumlah Hidden Neuron : 3

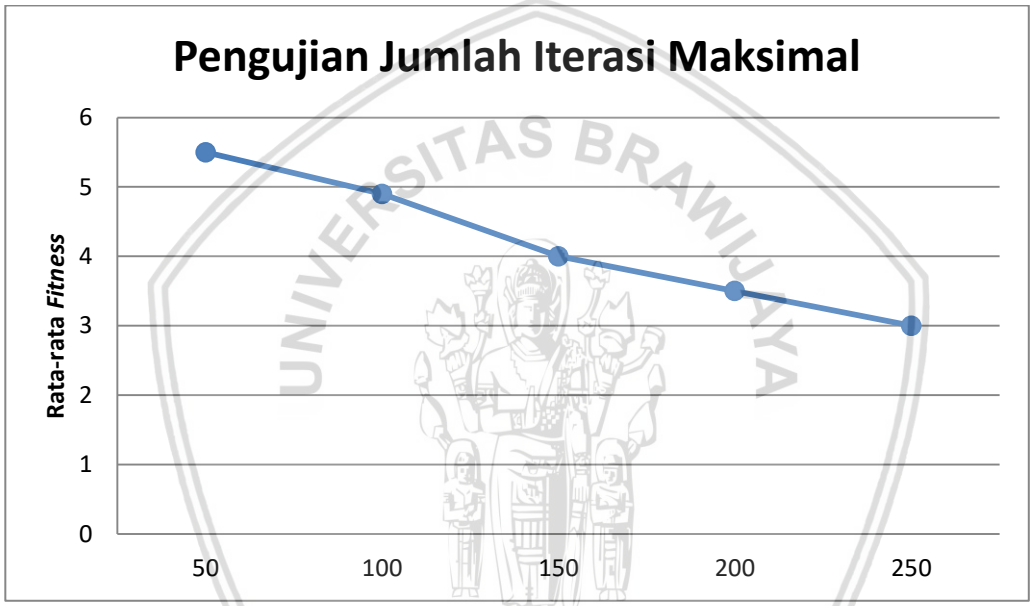
Hasil pengujian untuk jumlah iterasi maksimal dapat dilihat pada Tabel 6.2:

Tabel 6. 2 Tabel Pengujian Jumlah Iterasi Maksimal

Percobaan Ke – i	Nilai Fitness pada Pengujian Jumlah Iterasi Maksimal				
	50	100	150	200	250
1	3.31	3.96	8.49	4.07	4.33
2	5.7	4.98	3.54	4.43	1.51
3	1.21	11	4.1	4.02	4.41
4	10.8	11.75	4.42	2.38	2.68
5	9.7	8.39	3.64	1.43	3.59
6	7.74	3.19	3.06	6.54	2.67
7	3.7	1.71	3.31	4.06	3.23
8	3.66	0.41	5.17	2.2	2.18

9	6.92	1.91	1.43	1.12	3.09
10	2.26	2.07	3.21	5.65	2.94
Rata-rata *Fitness*	5.5	4.9	4.0	3.5	3.0

Berdasarkan pengujian jumlah iterasi maksimal didapatkan nilai rata-rata *fitness* terbaik dari tiap variasi pengujian. Grafik hasil pengujian jumlah maksimal iterasi ditunjukkan pada Gambar 6.2.



Gambar 6. 2 Pengujian Jumlah Iterasi Maksimal

Berdasarkan pada gambar 6.2 diatas, penurunan rata-rata nilai *fitness* mengalami konsistensi. Dapat disimpulkan dari gambar bahwa nilai rata-rata *fitness* juga dipengaruhi oleh banyaknya jumlah iterasi. Semakin banyak jumlah iterasi maka nilai *fitness* akan semakin menurun, hal ini dikarenakan banyaknya iterasi membuat pencarian solusi yang optimal lebih banyak. Nilai *fitness* yang terbaik yakni pada *iterasi* 250 dengan nilai *fitness* 3.0.

1.2.3 Pengujian Bobot Inersia (w)

Pengujian bobot inersia dilakukan untuk mengetahui bobot terbaik guna mendapatkan hasil prediksi yang optimal. Pengujian bobot inersia dimulai dari nilai 0.5 sampai 1. Pada masing-masing variasi bobot inersia dilakukan percobaan sebanyak 10 kali percobaan. Pada perhitungan PSO dan ELM banyak melibatkan

variabel yang dibangkitkan secara acak, sehingga mengakibatkan nilai yang dihasilkan berbeda pada setiap kali *running* program.

Nilai parameter lain yang digunakan pada pengujian ini adalah sebagai berikut:

a. Nilai Konstanta 1 (c1) : 1
b. Nilai Konstanta 2 (c2) : 1
c. Banyak data training dan testing : 80% : 20%
d. Jumlah ukuran populasi : 50
e. Jumlah Hidden Neuron : 3
f. Jumlah Iterasi : 250

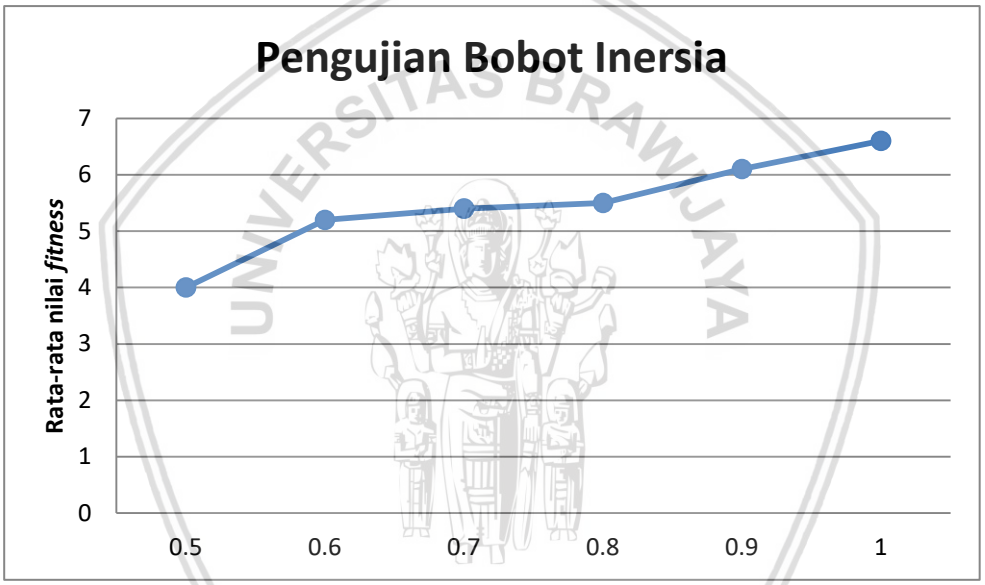
Hasil pengujian untuk bobot inersia dapat dilihat pada Tabel 6.3:

Tabel 6. 3 Tabel Pengujian Bobot Inersia

Percobaan Ke – i	Nilai *Fitness* pada Pengujian Bobot Inersia					
	0.5	0.6	0.7	0.8	0.9	1
1	1.05	9.47	3.08	3.86	12.4	3.5
2	2.08	2.37	11.2	6.29	4.03	2.78
3	6.01	6.24	3.77	6.09	2.21	8.03
4	7.11	1.24	8.1	4.22	0.1	4.87
5	2.87	8.6	11.16	4.69	1.85	2.74
6	14.29	13.9	4.59	2.81	13.23	6.58
7	1.41	5.87	2.48	2.93	2.9	13.63
8	2.04	2.92	3.83	9.16	2.01	5.33
9	3.12	0.1	4.23	12.72	18.75	15.35
10	0.91	2.03	2.25	2.6	3.8	3.61
Rata-rata *Fitness*	4.0	5.2	5.4	5.5	6.1	6.6

Berdasarkan pengujian bobot inersia didapatkan nilai rata-rata *fitness* terbaik dari tiap variasi pengujian. Grafik hasil pengujian bobot inersia pada bobot inersia ditunjukkan pada Gambar 6.3.

Berdasarkan Gambar 6.3 hasil pengujian bobot inersia menunjukkan bahwa, semakin besar nilai bobot inersia makan rata-rata *fitness* yang dihasilkan akan semakin besar pula. Peningkatan nilai rata-rata *fitness* dikarenakan apabila bobot inersia semakin besar maka nilai kecepatan partikel yang dihasilkanpun juga besar sehingga kecepatan pada partikel menjadi lebih lambat pada awal pencarian solusi. Lambatnya kecepatan partikel pada awal pencarian solusi akan memberikan kesempatan eksploitasi lokal yang lebih besar. Eksploitasi ini memiliki tujuan untuk mencari solusi yang optimal pada wilayah tertentu sebelum melakukan pencarian solusi pada wilayah lain yakni eksplorasi. Oleh sebab itu, apabila bobot inersia yang digunakan semakin kecil maka kecepatan partikel akan semakin cepat sehingga kesempatan eksploitasi partikel semakin kecil dan partikel akan lebih cepat melakukan eksplorasi (Sugianto, et al., 2017).



Gambar 6. 3 Pengujian Bobot Inersia

1.2.4 Pengujian Jumlah *Hidden Neuron*

Pengujian jumlah hidden neuron bertujuan untuk mengetahui jumlah node terbaik pada hidden neuron untuk mendapatkan hasil prediksi yang optimal. Pengujian jumlah hidden neuron dimulai dari angka 5 sampai 10. Pada masing-masing variasi jumlah hidden neuron dilakukan percobaan sebanyak 10 kali percobaan. Pada perhitungan PSO dan ELM banyak melibatkan variabel yang dibangkitkan secara acak, sehingga mengakibatkan nilai yang dihasilkan berbeda pada setiap kali *running* program.

Nilai parameter lain yang digunakan pada pengujian ini adalah sebagai berikut:

a. Jumlah Iterasi : 250
b. Bobot Inersia : 0.5
c. Nilai Konstanta 1 (c1) : 1
d. Nilai Konstanta 2 (c2) : 1
e. Banyak data training dan testing : 80% : 20%
f. Jumlah ukuran populasi : 50

Hasil pengujian untuk jumlah posize dapat dilihat pada Tabel 6.4 berikut:

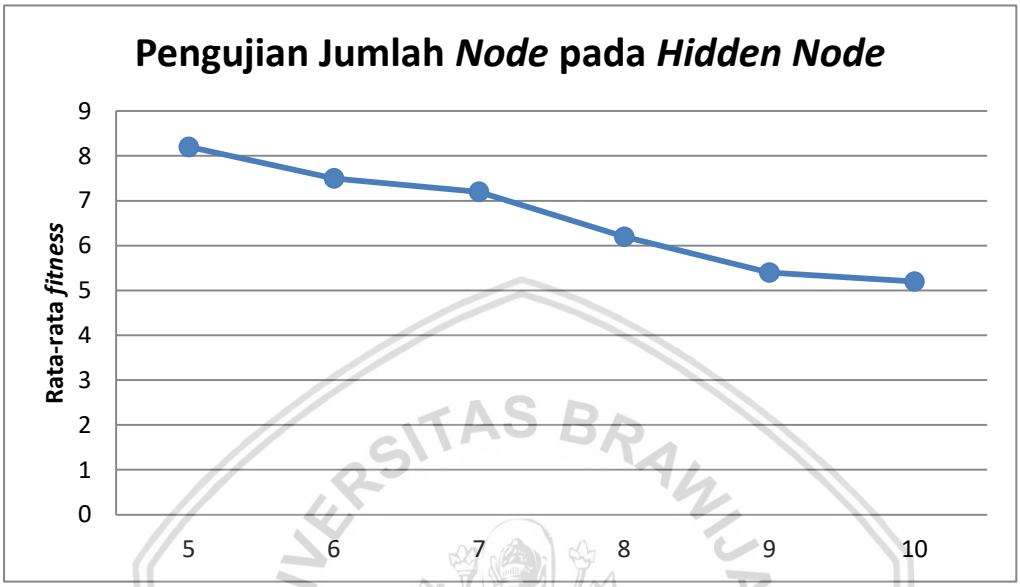
Tabel 6. 4 Tabel Pengujian Jumlah Hidden Neuron

Percobaan Ke – i	Nilai *Fitness* pada pengujian jumlah *Hidden Neuron*					
	5	6	7	8	9	10
1	12.9	12.8	8.72	4.39	1.45	5.49
2	6.11	18.1	2.28	2.67	4.31	2.76
3	3.87	1.53	12.9	5.73	4.2	3.27
4	16.32	2.85	10.11	7.34	2.28	7.02
5	1.99	10.5	3.17	5.21	4.44	8.09
6	14.5	16.7	6.55	2.35	2.95	3.59
7	6.13	3.8	1.63	13.13	1.86	1.81
8	8.06	3.52	5.48	6.1	13.5	4.12
9	4.65	2.87	16.46	6.01	13.9	13.7
10	8.24	2.66	4.82	9.11	5.84	2.33
Rata-rata *Fitness*	8.2	7.5	7.2	6.2	5.4	5.2

Berdasarkan pengujian jumlah *node* pada *hidden neuron* didapatkan nilai rata-rata *fitness* terbaik dari tiap variasi jumlah *node* pada *hidden neuron*. Grafik pengujian hasil jumlah *node* pada *hidden neuron* ditunjukkan pada Gambar 6.4.

Berdasarkan Gambar 6.4 hasil pengujian jumlah *node* pada *hidden neuron* menunjukkan bahwa semakin banyak jumlah *node* pada *hidden neuron* maka nilai dari rata-rata fitness akan semakin besar. Dari pengujian yang dilakukan, didapatkan nilai rata-rata fitness terkecil terdapat pada jumlah node 10 yakni 5.2. Fungsi dari

node sendiri dalam metode ELM adalah sebagai penghubung yang terbentuk antara *input layer* dan *output layer*. Sehingga dapat disimpulkan bahwa *node* pada *hidden neuron* dengan jumlah 10 menghasilkan nilai yang optimal.



Gambar 6. 4 Pengujian Jumlah Node pada Hidden Node

1.2.5 Pengujian Jumlah Data

Pengujian jumlah data training dilakukan untuk mengetahui pengaruh banyaknya data training terhadap proses perhitungan algoritma ELM. Pengujian ini terdiri dari 5 jenis perbandingan data, yaitu 80%:20%, 70%:20%, 60%:20%, 50%:20%, dan 40%:20% untuk data training dan data testing dengan tujuan untuk mengetahui pengaruh daru jumlah data training terhdap jumlah data testing yang mana data testing sebesar 20% dari keseluruhan data (database). Pada masing-masing variasi perbandingan data dilakukan percobaan sebanyak 10 kali percobaan. Pada perhitungan PSO dan ELM banyak melibatkan variabel yang dibangkitkan secara acak, sehingga mengakibatkan nilai yang dihasilkan berbeda pada setiap kali *running* program.

Nilai parameter lain yang digunakan pada pengujian ini adalah sebagai berikut:

a. Jumlah Iterasi : 250
b. Bobot Inersia : 0.5
c. Nilai Konstanta 1 (c1) : 1
d. Nilai Konstanta 2 (c2) : 1

e. Jumlah Hidden Neuron : 10

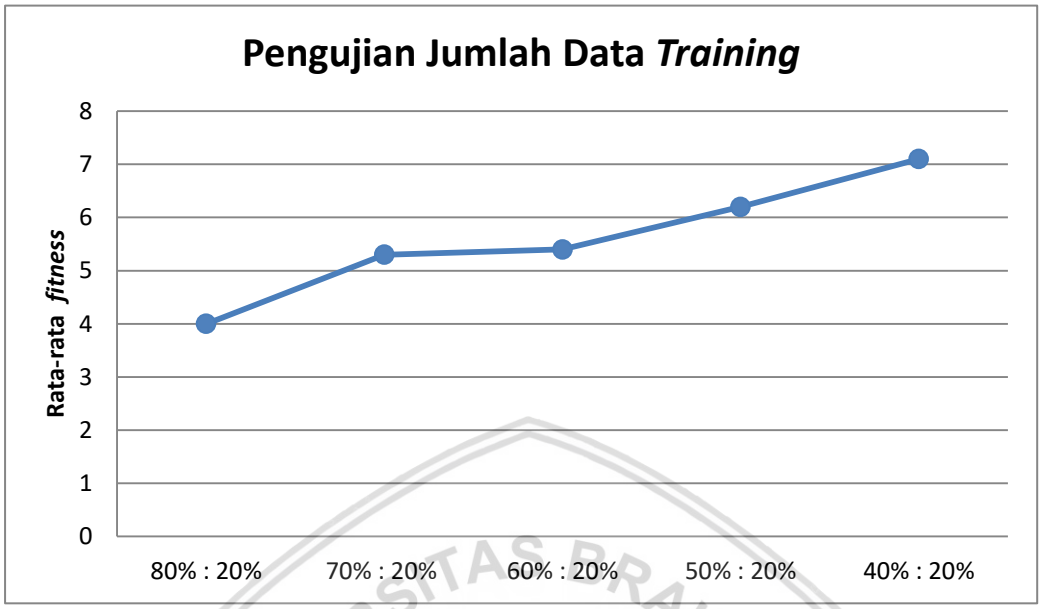
f. Jumlah ukuran populasi : 50

Hasil pengujian untuk jumlah data *training* dapat dilihat pada Tabel 6.5 berikut:

Tabel 6. 5 Tabel Pengujian Jumlah Data Training

Percobaan Ke – i	Nilai *Fitness* pada Pengujian Jumlah Data Training				
	80% 20%	70%:20%	60%:20%	50%:20%	40%:20%
1	0.95	9.18	11.09	12.62	4.9
2	2.12	2.66	8.09	10.32	2.8
3	1.99	1.83	2.77	10.16	12.87
4	5.44	2.14	5.7	5.46	8.02
5	5.79	6.54	9	0.52	0.72
6	4.67	8.44	2.81	2.5	3.67
7	0.92	3.06	4.89	5.41	15.67
8	8.39	12.4	3.93	3.63	13.01
9	7.08	1.84	3.99	3.21	3.7
10	2.96	5.79	2.09	8.57	5.64
Rata-rata *Fitness*	4.0	5.3	5.4	6.2	7.1

Berdasarkan pengujian data *training* didapatkan nilai rata-rata *fitness* terbaik dari tiap variasi jumlah data *training*. Grafik pengujian hasil jumlah data *training* ditunjukkan pada Gambar 6.5.



Gambar 6. 5 Pengujian Jumlah Data Training

Berdasarkan pada Gambar 6.5 rata-rata nilai *fitness* akan bertambah besar apabila jumlah data *training* semakin kecil. Hal tersebut dibuktikan pada perbandingan data 80% : 20% dengan rata-rata nilai *fitness* terkecil yang memiliki nilai 4.0. Hal ini dikarenakan metode ELM merupakan metode pelatihan dimana semakin banyak nilai data training yang digunakan maka semaikin banyak data yang dilatih pada proses pelatihan sehingga memiliki banyak pertimbangan untuk menghasilkan optimasi yang baik.

1.2.6 Pengujian Hidden Neuron pada *Extreme Learning Machine* (ELM) tanpa Particle Swarm Optimization (PSO)

Pada pengujian ini bertujuan untuk mencari nilai node pada *hidden neuron* yang optimal tanpa menggunakan metode PSO. Pengujian jumlah node dimulai dari angka 5 sampai 10. Pada masing-masing variasi jumlah node dilakukakan percobaan sebanyak 10 kali.

Hasil pengujian untuk Hidden Neuron pad ELM tanpa PSO dapat dilihat pada Tabel 6.6 :

Tabel 6. 6 Tabel Pengujian Hidden Neuron pada ELM tanpa PSO

Percobaan Ke – i	Nilai fitness pada pengujian jumlah hidden neuron pada ELM tanpa PSO					
	5	6	7	8	9	10
1	12.06	10	23.35	13.41	9.5	18.26
2	45.7	13.95	11.49	7.41	16.83	9.5
3	19.37	19.95	12.39	18.94	18.84	7.87

4	13.58	17.41	17.5	39.8	7.51	15.58
5	33.01	12.08	26.61	8.23	32.8	19.69
6	13.66	41.85	11.06	10.76	16.64	24.73
7	17.94	21.13	9.27	20.9	21.2	9.8
8	38.77	7.05	12.29	7.21	11.49	17.82
9	30.41	7.6	27.79	19.69	8.78	13.31
10	15.74	9.27	7.78	10.79	13.11	14.89
Rata-rata *Fitness*	24.0	16.0	15.9	15.7	15.6	15.1

1.2.7 Pengujian Prediksi *Extreme Learning Machine* (ELM)

Berdasarkan parameter-parameter optimal dari hasil pengujian yang didapatkan, dilakukan pengujian algoritme ELM dan ELM-PSO untuk membandingkan hasil nilai error algoritme ELM dan nilai error ELM-PSO. Perbandingan hasil nilai error dari kedua algoritme ditunjukkan pada Tabel 6.7.

Tabel 6. 7 Tabel Perbandingan nilai *error*

Percobaan Ke – i	Nilai error ELM	Nilai error ELM-PSO
1	37.45%	3.26%
2	47.56%	4.25%
3	30.45%	8.16%
4	22.79%	6.49%
5	16.82%	5.7%
6	28.58%	5.7%
7	13.43%	11.59%
8	21.34%	10.77%
9	12.42%	4.14%
10	23.02%	1.24%

Dari 10 percobaan yang dilakukan, didapatkan nilai *error* paling kecil dari masing-masing algoritme adalah 13.43 untuk algoritme ELM biasa dan 1.24 untuk algoritme ELM-PSO. Hal ini menunjukkan bahwa metode optimasi PSO mampu meningkatkan nilai *error* untuk permasalahan prediksi produksi. Nilai *input weight* yang optimal didapat pada nilai *fitness* dengan nilai 1.24 adalah sebagai berikut [0.99, -0.4, -0.04, 0.73, -0.73, -0.42, -0.33, -0.18, 0.11, 0.23, -0.85, -0.94, -0.62, -0.28, 0.18, -0.9, 0.68, -0.06, 0.55, -0.5, 0.15, 0.5, -0.37, -0.35, -0.56, 0.17, -0.31, -0.14, -0.27, -0.19, -0.41, -

0.72, -0.19, -0.73, -0.02, 0.27, 0.13, -0.58, 0.89, 0.56, -0.01, 0.37, -0.28, 0.51, 0.43, -0.24, -0.38, 0.84, -0.9, -0.1, 0.68, -0.57, -0.41, -0.56, 0.43, -0.71, 0.08, -0.36, -0.69, -0.87]. Selanjutnya nilai tersebut digunakan untuk melakukan pengujian prediksi ELM yang bertujuan untuk mencari nilai MAPE dari semua data dengan menggunakan *input weight* yang optimal. Data yang digunakan untuk pengujian ini diambil secara random dari data *testing*, hasil menunjukkan nilai MAPE sebesar 0.59%.

1.2.8 Hasil Pengujian dan Analisis Global

Pada penelitian ini telah dilakukan 7 kali percobaan terkait dengan metode yang digunakan, yakni pengujian jumlah hidden neuron, pengujian perbandingan data, pengujian hidden neuron ELM tanpa PSO dan pengujian prediksi untuk mengetahui kinerja dari metode *Extreme Learning Machine*, sedangkan pengujian jumlah ukuran populasi, iterasi maksimal dan bobot inersia digunakan untuk mengetahui kinerja dari metode *Particle Swarm Optimization.* Pengujian pertama terkait dengan jumlah ukuran populasi menunjukkan apabila jumlah ukuran populasi semakin besar maka nilai rata-rata *fitness* semakin kecil. Dari pengujian tersebut didapat jumlah ukuran populasi sebanyak 50 dengan rata-rata nilai *fitness* 5.01%. Pengujian kedua terkait dengan jumlah iterasi maksimal, dikarenakan banyaknya iterasi membuat pencarian solusi optimal lebih banyak. Dari pengujian ini didapatkan iterasi maksimal adalah 250 dengan rata-rata nilai *fitness* 3.06%. Pengujian ketiga terkait terkait dengan pengaruh bobot inersia terhadap nilai *fitness*, bahwa semakin besar bobot inersia yang digunakan maka perubahan kecepatan yang dihasilkan juga besar yang artinya kecepatan partikel akan menjadi lambat. Sedangkan bobot inersia sendiri merupakan pengontrol dari perubahan kecepatan partikel dalam mencari solusi yang optimal sehingga menyebabkan partikel mempunyai kesempatan untuk melakukan eksploitasi lebih dalam dan menghasilkan solusi yang paling optimal. Dalam pengujian ini dihasilkan nilai bobot inersia yang optimal adalah 0.5 dengan nilai rata-rata *fitness* 4.08%.

Pengujian keempat terkait dengan banyaknya jumlah hidden neuron yang menunjukkan semakin banyak jumlah hidden neuron maka nilai fitness akan semakin kecil. Dari pengujian ini didapatkan jumlah hidden neuron yang paling baik berjumlah 10 dengan nilai rata-rata fitness 5.21%. Pengujian kelima terkait dengan pengaruh banyaknya perbandingan data training dan data testing yang digunakan. Dari hasil pengujian menunjukkan bahwa penggunaan data training yang banyak bisa memberikan hasil yang baik. Perbandingan data training dan data testing dari hasil pengujian adalah 36 data training dan 9 data testing dengan nilai fitness 4.03 %. Pengujian keenam terkait dengan pengujian *hidden neuron* pada ELM tanpa menggunakan PSO sebagai optimasi untuk nilai *input weight* dan bias. Pada pengujian ini nilai hidden neuron 10 juga memiliki nilai yang paling kecil dengan nilai *error* 15.41%. Pengujian terakhir atau pengujian ketujuh terkait dengan pengujian

prediksi dimana dalam pengujian ini menguji nilai input weight yang optimal. Dalam pengujian ini data diambil secara random yang menghasilkan nilai MAPE sebesar 0.59%.

BAB 7 PENUTUP

Pada bab ini akan dibahas mengenai kesimpulan dan saran dari penelitian yang telah dilakukan.

1.1 Kesimpulan

Berdasarkan perancangan, implementasi dan hasil pengujian *Particle Swarm Optimization* dalam Memprediksi Produksi Gula Kristal Putih dengan *Extreme Learning Machine*, didapatkan kesimpulan sebagai berikut:

1. Sistem optimasi bobot ELM untuk prediksi gula dengan PSO telah diimplementasikan dan dilakukan pengujian untuk mendapatkan parameter-parameter terbaik. Parameter-parameter terbaik ini digunakan untuk mendapatkan hasil optimasi bobot yang optimal. Parameter-parameter terbaik yang didapat dari hasil pengujian adalah jumlah ukuran populasi 50, perbandingan data *training* 80% dengan jumlah data *training* sebanyak 36 data, jumlah *hidden neuron* 10, bobot inersia 0.5, dan iterasi maksimal yakni 250. Dari hasil perhitungan parameter tersebut didapatkan nilai *Mean Absolute Percentage Error*(MAPE)-nya 1.24% dengan jumlah partikel dengan nilai [0.99, -0.4, -0.04, 0.73, -0.73, -0.42, -0.33, -0.18, 0.11, 0.23, -0.85, -0.94, -0.62, -0.28, 0.18, -0.9, 0.68, -0.06, 0.55, -0.5, 0.15, 0.5, -0.37, -0.35, -0.56, 0.17, -0.31, -0.14, -0.27, -0.19, -0.41, -0.72, -0.19, -0.73, -0.02, 0.27, 0.13, -0.58, 0.89, 0.56, -0.01, 0.37, -0.28, 0.51, 0.43, -0.24, -0.38, 0.84, -0.9, -0.1, 0.68, -0.57, -0.41, -0.56, 0.43, -0.71, 0.08, -0.36, -0.69, -0.87]
2. Tingkat kesalahan prediksi pada ELM yang dioptimasi dengan PSO dengan nilai evaluasi menggunakan MAPE menghasilkan nilai sebesar 0.59% dengan tingkat prediksi yang baik.
3. Tingkat kesalahan antara ELM biasa dengan ELM-PSO lebih kecil tingkat kesalah ELM-PSO daripada ELM biasa. Hal ini menunjukkan bahwa metode optimasi PSO mampu meningkatkan nilai *error* untuk permasalahan prediksi produksi.

1.2 Saran

Saran yang dapat diberikan dari hasil penelitian yang digunakan sebagai bahan pengembangan dalam penelitian ini, adalah sebagai berikut:

1. Menggunakan lebih banyak jumlah data sehingga hasil yang didapat dapat lebih optimal karena perbandingan jumlah data juga mempengaruhi hasil dari prediksi.
2. Pada penelitian mendatang peneliti dapat menambahkan parameter lain yang merupakan faktor-faktor dalam produksi gula. Sehingga hasil dari

prediksi lebih baik karena semakin banyak parameter yang digunakan hasil prediksi yang didapat juga semakin baik.

3. Pada penelitian mendatang peneliti juga dapat menggabungkan metode ELM dengan metode lain untuk dibandingkan dengan PSO-ELM sehingga dapat digunakan pada penelitian selanjutnya untuk meningkatkan hasil yang didapat pada penelitian sebelumnya.

DAFTAR PUSTAKA

Agustina, Irwin Dwi. 2010. *Penerapan Metode Extreme Learning Machine untuk Peramalan Permintaan*. Institut Teknologi Sepuluh Nopember.

Cholissodin, I., 2016. Analisis Big Data-Semester Ganjil 2017-2018.

Cholissodin, I., 2017. Buku Ajar Swarm Intteligence.

Cholissodin, Imam. 2016. *Optimasi Kandungan Gizi Susu Kambing Peranakan Etawa (PE) Menggunakan ELM – PSO Di UPT Pembibitan Ternak dan Hijauan Makanan Ternak Singosari – Malang*. Universitas Brawijaya.

Handika, I Putu S. 2016. *Perbandingan Metode Extreme Learning Machine dan Particle Swarm Optimization Extreme Learning Machine untuk Peramalan Jumlah Penjualan Barang*. Teknologi Elektro, Vol.15, No.1, Januari-Juni.

Muhamad, H., Cahyo, A.P., Sugianto, N.A., Surtiningsih, L., 2016. *Optimasi Naïve Bayes Classifier Dengan Menggunakan Pasrticle Swarm Optimization Pada Data Iris*. Jurnal Teknologi Informasi dan Ilmu Komputer. Universitas Brawijaya.

Napitulu, Dewi Agustina. 2013. *Faktor-Faktor Yang Mempengaruhi Produksi Gula Dalam Negeri dan Proyeksi Produksi dan Konsumsi Gula Di Indonesia. Universitas* Atma Jaya Yogyakarta.

Subagyo, Pangestu. 1986. *Forcesting Konsep dan Aplikasi*. BPEE UGM: Yogyakarta.

Sugiyanto, Catur. 2007. Permintaan Gula Di Indonesia. Universitas Gajah Mada.

Sun, Z.L., Choi, T.M., Au, K.F., dan Yu, Y. 2008. *Sales Forecasting using Extreme Learning Machine with Application in Fashion Retailing. Elsevier Decision Support Systems* 46 (2008) 411-419.

Siwi, Iga P. 2016. *Peramalan Produksi Gula Pasir Menggunakan Extreme Learning Machine (ELM) Pada PG Candi Baru Sidoarjo.* Universitas Brawijaya.

S, Karpachlevi. 2014. *Classification of ECG Signals Using Hybrid Particle Swarm Optimization in Extreme Learning Machine*. *Journal of Applied Sciences and Engineering Research*, Vol. 3, Issue 5.

Sugianto, Nur Afifah. 2017. *Klasifikasi Keminatan Menggunakan Algoritme Extreme Learning Machine dan Particle Swarm Optimization Untuk Seleksi Fitur (Studi Kasus: Program Studi Teknik Informatika FILKOM UB)*. Universitas Brawijaya.

Vincent Gaspersz. 2002. *Production Planning and Inventory Control*. PT. Gramedia Pustaka Utama: Jakarta

Zhao, Min-Ru, Zhang, Jian-Ming, Han, Feri. 2014. *An Improved Extreme Learning Machine with Adaptive Growth of Hidden Nodes based on Particle Swarm*

Optimization. International Joint Conference on Neural Netwoks (IJCNN). Beijing, China. July 6-11 2014.