PERAMALAN PEMAKAIAN AIR PADA PLTGU DI PEMBANGKITAN LISTRIK JAWA BALI UNIT GRESIK MENGGUNAKAN *EXTREME LEARNING MACHINE* DENGAN OPTIMASI ALGORITME GENETIKA

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:
Heny Dwi Jayanti
NIM: 145150207111047

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

PENGESAHAN

PERAMALAN PEMAKAIAN AIR PADA PLTGU DI PEMBANGKITAN LISTRIK JAWA BALI UNIT GRESIK MENGGUNAKAN *EXTREME LEARNING MACHINE* DENGAN OPTIMASI ALGORITME GENETIKA

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh :
Heny Dwi Jayanti
NIM: 145150207111047

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

Dosen Pembimbing I	*Dosen Pembimbing II*
Imam Cholissodin, S.Si, M.Kom	Edy Santoso, S.Si, M.Kom
NIK: 201201 850719 1 001	NIP: 19740414 200312 1 004

Mengetahui
Ketua Jurusan Teknik Informatika

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

IDENTITAS TIM PENGUJI

- Penguji I (Majelis penguji ujian skripsi)
  - Mochammad Ali Fauzi, S.Kom, M.Kom
  - NIK. 201502 890101 1 001
- Penguji II
  - Lailil Muflikhah, S.Kom, M.Sc
  - NIP. 19741113 200501 2 001

PERNYATAAN ORISINALITAS

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

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

Malang, 25 Mei 2018

Heny Dwi Jayanti

NIM: 145150207111047

ABSTRAK

Air merupakan kebutuhan mutlak sehari-hari yang memiliki peran penting. Banyak orang yang memanfaatkan air tidak hanya untuk kebutuhan rumah tangga namun juga untuk kebutuhan perindustrian. Salah satu pemanfaatan air yang digunakan untuk sektor industri adalah PLTGU pada PT Pembangkitan Jawa Bali. Dalam penggunaan air terdapat 2 jenis air yang harus diperhatikan yaitu air laut dan air tawar. Untuk air laut memilki kapasitas lebih banyak sehingga mampu memenuhi kebutuhan industri dibandingkan dengan air tawar. Hal ini membuat industri listrik tersebut telah mengolah air laut menjadi air tawar yang disebut dengan proses desalinasi. Namun pada proses PLTGU sering mengalami masalah dalam pengolahan air seperti terjadinya kebocoran pipa karena terjadi korosi, perbedaan perlakuan pengisian air, serta proses desalinasi yang memakan waktu lama mengakibatkan kinerja turbin tidak stabil. Dengan beberapa permasalahan yang muncul, maka dibutuhkan sebuah solusi. Pada penelitian ini peneliti telah mengusulkan sebuah sistem peramalan pemakaian air dengan menggunakan metode *extreme learning machine* (ELM) dengan optimasi Algoritme genetika. Algoritme genetika digunakan untuk mengoptimasi nilai *input weight* yang didapatkan secara *random* pada metode ELM. Sedangkan untuk merepresentasikan kromosom menggunakan *real code*. Pada tahap reproduksi menggunakan *crossover* dengan metode *extended intermediate crossover* dan mutasi dengan metode *random mutation*. Hasil pengujian metode ELM&Algoritme genetika menghasilkan rata-rata nilai MAPE sebesar 0.428 dengan parameter perbandingan nilai *crossover rate* (Cr) senilai 0.4 dan *mutation rate* (Mr) senilai 0.6, jumlah *popsize* sebesar 200, jumlah generasi sebesar 1000, dan jumlah data *training* sebesar 80% dari keseluruhan dataset. Sedangkan dengan parameter yang sama dilakukan pengujian menggunakan metode ELM memiliki rata-rata MAPE sebesar 4.517. Dari hasil MAPE menunjukkan bahwa gabungan metode ELM dengan algoritme genetika mampu memperkecil nilai *error* pada peramalan dibandingkan dengan metode ELM.

Kata kunci: Peramalan, optimasi, pemakaian air, *Extreme Learning Machine*, Algoritme Genetika.

ABSTRACT

Water is an absolute necessity every day that has an important role. Many people use the water not only for household needs but also for industrial needs. One of the utilization of water used for the industrial sector is PLTGU in PT Pembangkitan Jawa Bali. In the user of water, there are two types of water that must consider, that is the sea water and fresh water. For seawater has more capacity so that it can meet needs of industry compared with fresh water. This makes the electricity industry has treated the sea water into fresh water called desalination process. However, in the PLTGU process often experience problems in water treatment such as the occurrence of leaking pipe due to corrosion, the difference of water filling treatment, and the long-time desalination process resulted in unstable turbine performance. With some problems that arise, then needed a solution. In this study, researchers have proposed a water forecasting system using the method of extreme learning machine (ELM) with the optimization of Genetic Algorithm. The genetic algorithm is used to optimize the input weight values obtained randomly on the ELM method. Meanwhile, to represent chromosomes using real code. At the reproduction stage using the crossover with extended intermediate crossover method and mutation by random mutation method. The result of ELM test method and genetic algorithm resulted in average MAPE value of 0.428 with a parameter value of crossover rate (Cr) value 0.4 and mutation rate (Mr) equal to 0.6, popsize amount 200, number of generation 1000, and training data amount 80% of the entire dataset. While the same parameters were tested using ELM method has MAPE average of 4.517. From the results obtained MAPE, shows that the combined ELM method with genetic algorithm able to minimize the error value in forecasting compared with the ELM method.

Keywords: *Forecasting, Optimation, water used, Extreme Learning Machine*, *Genetic Algorithm*.

KATA PENGANTAR

Dengan mengucapkan kata syukur penulis panjatkan kehadirat Tuhan Yang Maha Esa telah melimpahkan rahmat, hidayah serta karunia-Nya sehingga penulis dapat menyelesaikan skripsi yang berjudul "Peramalan Pemakaian Air Pada PLTGU Di Pembangkitan Listrik Jawa Bali Unit Gresik Menggunakan *Extreme Learning Machine* Dengan Optimasi Algoritme Genetika" dengan lancar dan baik.

Pada kesempatan kali ini, penulis menyampaikan banyak terimakasih kepada semua pihak yang telah memberikan bimbingan, dukungan, saran, doa dan motivasi dalam pengerjaan penelitian. Oleh karena itu, penulis ingin menyampaikan rasa terimakasih khususnya kepada:

1. Imam Cholissodin, S.Si, M.Kom selaku Dosen Pembimbing pertama atas bimbingan serta saran yang diberikan kepada penulis selama penelitian
2. Edy Santoso, S.Si, M.Kom selaku Dosen Pembimbing kedua atas bimbingan serta saran yang telah diberikan kepada penulis selama penelitian
3. Agus Wahyu Widodo, S.T, M.Cs., selaku Ketua Program Studi Teknik Informatika Fakultas Ilmu Komputer Universitas Brawijaya Malang
4. Tri Astoto Kurniawan, S.T, M.T., selaku Ketua Jurusan Teknik Informatika Fakultas Ilmu Komputer Universitas Brawijaya Malang
5. Kedua orang tua serta keluarga penulis telah memberikan dukungan, motivasi serta doa selama penelitian
6. Bapak I Wayan Santiyasa selaku staff Sistem Informasi pada PLTGU PT Pembangkitan Jawa Bali UP Gresik yang telah memberikan data peneliti serta arahan
7. Sahabat-sahabat yang telah memberikan dukungan sepenuhnya kepada penulis yaitu Anim Rofi'ah, Audia Refanda, Novi Nur P, Dhea Azahria, Alysha Ghea, Miracle, serta GL Dev
8. Semua pihak yang telah membantu kelancaran peneliti dalam melaksanakan penelitian ini

Penulis menyadari bahwa dalam penulisan pada penelitian ini masih banyak kekurangan dan kesalahan yang dikarenakan oleh keterbatasan ilmu. Oleh sebab itu, penulis sangat mengharapkan kritik dan saran yang bersifat membangun untuk penelitian selanjutnya. Penulis berharap penelitian ini dapat memberikan manfaat bagi penulis sendiri maupun bagi seluruh pihak.

Malang, 25 Mei 2018

Penulis

27heny@gmail.com

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR PERSAMAAN

DAFTAR KODE PROGRAM

DAFTAR LAMPIRAN

BAB 1 PENDAHULUAN

1.1 Latar belakang

Air merupakan sumber kehidupan mahkluk hidup yang sangat penting terutama bagi manusia dengan berbagai macam kebutuhan dasar manusia (*basic human need*) (Suryadmaja, et al., 2015). Seiring dengan bertambahnya jumlah penduduk Indonesia mendorong untuk meningkatnya kebutuhan air. Selain itu fungsi air tidak hanya mendukung kebutuhan dasar manusia, akan tetapi juga mendukung sektor lainnya seperti kebutuhan air untuk mendukung kegiatan pertanian, untuk pelayaran, untuk kebutuhan rumah tangga, serta berfungsi sebagai pendukung industri di pembangkitan listrik.

PT Pembangkitan Jawa Bali Unit Gresik merupakan industri pembangkitan listrik yang berlokasi di Gresik Jawa Timur. PT Pembangkitan Jawa Bali merupakan sebuah perusahaan yang menghasilkan listrik yang mempunyai tiga pembangkit listrik diantaranya Pembangkitan Listrik Tenaga Gas (PLG), Pembangkitan Listrik Tenaga Gas dan Uap (PLTGU), serta Pembangkitan Listrik Tenaga Uap (PLTU). Pada penelitian ini akan difokuskan pada Pembangkitan Listrik Tenaga Gas dan Uap (PLTGU). Pada PLTGU, air yang digunakan berasal dari air laut kemudian akan dilakukan proses desalinasi untuk mengubah menjadi air tawar. Proses desalinasi adalah proses pengolahan air laut menjadi air tawar. Air hasil desalinasi tersebut berfungsi untuk pemasok utama air kondensat yang terdapat pada wadah penampung. Kegunaan air salah satunya digunakan untuk menggerakkan turbin yang telah dipanaskan oleh limbah gas bumi menjadi uap kemudian diproses oleh generator untuk menjadi listrik.

Kebutuhan air yang sangat banyak untuk proses pengolahan pembangkitan listrik, dimana setiap kali proses penguapan air selalu berkurang kapasitasnya. Sehingga di dalam tangki air terdapat kapasitas daya tampung yang selalu dipantau supaya tidak terjadi kekurangan air dalam proses produksi listrik. Pada proses pembangkitan listrik tenaga gas dan uap sering terdapat kendala-kendala dalam peyaluran air tangki seperti kebocoran pipa, waktu yang dibutuhkan untuk proses desalinasi lama, dan perbedaan perlakuan terhadap pengisian air. Kendala-kendala tersebut sangat mengganggu jumlah air yang seharusnya di proses untuk menggerakkan turbin kemudian gagal karena jumlah air di dalam tangki tidak mencukupi. Oleh sebab itu, kapasitas air di dalam tangki harus selalu penuh atau air di dalam tangki tidak kurang dari batas minimal pemakaian. Akan tetapi, dari pihak karyawan PT. PJB dalam menangani permasalahan-permasahan tersebut masih manual. Penanganan dalam kekurangan persediaan air dari masing-masing pihak karyawan desalinasi berbeda-beda. Ada karyawan yang selalu beracuan pada ukuran tangki, seperti setiap tinggi air mencapai 7 meter maka akan dilakukan pemompaan air laut untuk memenuhi tangki lagi. Selain itu jika sudah kehabisan persediaan air maka akan mengambil persediaan air dari Pembangkitan Listrik Tenaga Uap (PLTU). Oleh karena itu sangat dibutuhkan peramalan

pemakaian air supaya tidak terjadi kekurangan persediaan meskipun terjadi kebocoran pipa, atau faktor lainnya.

Banyak penelitian-penelitian mengenai peramalan yang akurat, salah satunya menggunakan metode *Extreme Learning Machine* (ELM). Pada metode ELM pertama kali diperkenalkan oleh Huang, dimana ELM adalah sebuah metode pada jaringan syaraf tiruan *feedforward* dengan *single hidden* layer (Khotimah, et al., 2010). Metode ELM dibuat untuk mengatasi kekurangan yang terdapat pada jaringan syarat tiruan terutama dalam hal *learning speed.* Penelitian-penelitian peramalan sudah banyak dilakukan serta dikembangkan. Salah satu penelitian sebelumnya menggunakan *Extreme Learning Machine* (ELM) untuk sistem pendukung keputusan dalam peramalan jumlah kunjungan pasien menghasilkan nilai *error* MSE (*Mean Squared Error*) yang optimal (Fardani, et al., 2015). Pengujian yang dilakukan pada penelitian ini dengan mencocokkan hasil yang terdapat pada data asli dengan hasil yang dikeluarkan oleh sistem. Dengan menggunakan 116 data *testing* untuk melakukan evaluasi berdasarkan fungsi aktivasi dan jumlah *hidden layer.*

Pada metode ELM terdapat nilai *input weight* yang diperoleh secara *random* sehingga nilai evaluasi yang didapatkan bisa lebih besar. Hal tersebut perlu dilakukan optimasi pada *input weight* dengan tujuan untuk mendapatkan nilai yang optimal yang digunakan pada peramalan. Salah satu solusi yang digunakan untuk mengoptimalkanya dengan menggunakan Algoritme Genetika yang bertujuan untuk mendapatkan nilai *error* yang lebih kecil dan hasil peramalan yang lebih baik. Hal tersebut telah dibuktikan oleh penelitian sebelumnya yang berjudul "Genetic Algorithm Optimization for Extreme Learning Machine based Microalga Growth Forcasting of *Chlamydomonas sp*" (Purnomo, et al., 2015). Hasil dari penelitian tersebut menunjukkan bahwa *Hybrid* Algoritme Genetika mampu memberikan nilai *error* yang sangat kecil yaitu 0.004 menggunakan RMSE dan RAE sebesar 26.142% dengan 110 neuron dalam mengoptimalkan *input weight*.

Beberapa penelitian beserta solusi yang diberikan sebelumnya bisa menjadi acuan untuk menyelesaikan permasalahan ini. Maka algoritme *Extreme Learning Machine* (ELM) dengan optimasi Algoritme Genetika dirasa sudah cukup untuk digunakan dalam mendapatkan solusi dari peramalan pemakaian air. Sehingga dengan perpaduan dua metode tersebut diharapkan mampu memberikan solusi dari permasalahan diatas. Untuk pengujiannya menggunakan nilai *error* MAPE (*Means Absolute Percentage Error*). Pada penelitian perbandingan empiris perhitungan nilai *error* dalam kasus peramalan, nilai MAPE (*Means Absolute Percentage Error*) memberikan hasil yang lebih baik dibandingkan menggunakan RMSE (*Root Mean Squared Error*) (Armstrong & Collopy, 1992). Sehingga MAPE diharapkan mampu menghitung tingkat keakuratan dalam permasalahan ini.

1.2 Rumusan masalah

Berdasarkan latar belakang permasalahan yang telah dijelaskan diatas, maka dapat dirumuskan permasalahannya sebagai berikut:

1. Bagaimana mengimplementasikan metode *Extreme learning Machine* (ELM) untuk peramalan pemakaian air pada PLTGU dengan optimasi algoritme genetika?
2. Berapa nilai tingkat keakuratan dalam MAPE (*Means Absolute Percentage Error*) dengan dalam hasil peramalan pemakaian air menggunakan metode *Extreme learning Machine* (ELM) dengan optimasi algoritme genetika?

1.3 Tujuan

Tujuan dari penelitian ini sebagai berikut:

1. Menerapkan metode *Extreme learning Machine* (ELM) dengan optimasi algoritme genetika untuk peramalan pemakaian air.
2. Mengetahui tingkat keakuratan peramalan pemakaian air yang diperoleh menggunakan metode *Extreme learning Machine* (ELM) dengan optimasi algotima genetika menggunakan MAPE (*Mean Absolute Precentage Error*).

1.4 Manfaat

Manfaat pada penelitian ini adalah untuk mengetahui hasil peramalan pemakaian air pada pembangkitan listrik di PT. PJB Unit Gresik agar dapat membantu karyawan dalam mengambil keputusan untuk memperkirakan pemakaian air yang harus dipenuhi saat proses desalinasi.

1.5 Batasan masalah

Agar penelitian ini fokus pada permasalahan yang telah dirumuskan, maka diperlukan batasan masalah dalam penelitian ini sebagai berikut:

1. Data pemakaian air pada PLTGU yang digunakan berasal dari PT. PJB Unit Gresik mulai dari tanggal 1 januari 2017 sampai dengan 31 juli 2017.
2. Parameter yang digunakan adalah jumlah produksi pemakaian air setiap hari.
3. Perhitungan tingkat kesalahannya menggunakan *Mean Absolute Precentage Error* (MAPE).

1.6 Sistematika pembahasan

Sistematika penulisan pada penelitian ini terdapat tujuh bab sebagai berikut:

BAB 1 PENDAHULUAN

Pada bab ini terdapat latar belakang permasalahan, rumusan masalah, tujuan penelitian, manfaat penelitian, batasan masalah, serta sistematika pembahasan dalam penelitian.

BAB 2 LANDASAN KEPUSTAKAAN

Bab ini menjelaskan tentang teori yang berhubungan dengan objek penelitian yang diperoleh dari berbagai sumber pustaka. Teori yang terdapat pada bab

ini adalah *Extreme Learning Machine* (ELM), Konsep Algoritme Genetika, serta Kebutuhan Air.

BAB 3 METODOLOGI

Pada bab metodologi menjelaskan tentang metode yang digunakan dalam penelitian yang terdiri dari studi literature, pengumpulan data, analisa kebutuhan dan perancangan sistem, implementasi pada sistem, pengujian dan analisis sistem, serta penarikan kesimpulan.

BAB 4 PERANCANGAN

Bab ini membahas tentang perancangan algoritme, perhitungan manualisasi, perancangan antarmuka, perancangan skenario pengujian.

BAB 5 IMPLEMENTASI

Pada bab ini berisi penjelasan tentang implementasi dari sistem, serta algoritme operasi yang akan diimplementasikan pada penelitian ini.

BAB 6 PENGUJIAN DAN ANALISIS

Bab ini meliputi teknik pengujian pada sistem yang sudah jadi, serta analisis dari hasil pengujian sistem.

BAB 7 PENUTUP

Bab ini berisi tentang kesimpulan yang diperoleh dari hasil penelitian, pengujian sistem, serta saran untuk pengembangan selanjutnya.

BAB 2 LANDASAN KEPUSTAKAAN

Landasan kepustakaan berisi tentang kajian pustaka dan teori yang mendukung penelitian. Pada kajian pustaka membahas tentang penelitian-penelitian yang sudah dilakukan sebelumnya sebagai acuan dalam pelaksanaan penelitian berikutnya. Untuk dasar teori pada penelitian ini difokuskan pada kebutuhan air.

1.1 Kajian Pustaka

Kajian pustaka pada penelitian ini akan membahas mengenai peramalan dengan metode *Extreme Learning Machine* (ELM) dengan optimasi algotima genetika yang telah dilakukan sebelumnya. Beberapa referensi mengenai peramalan akan digunakan peneliti untuk mendukung proses penelitian ini. penelitian – penelitian sebelumnya akan dijelaskan pada Tabel 2.1.

Pada penelitian pertama tentang "Genetic Algorithm Optimization for Extreme Learning Machine based Microalgal Growth Forecasting of Chlamydomonas sp". Pada penelitian ini bertujuan untuk peramalan pertumbuhan microalga menggunakan metode *Extreme Learning Machine* (ELM). Sedangkan Algoritme Genetika digunakan untuk mengoptimalkan nilai *input weight* pada proses ELM. Setiap generasi pada Algorime genetika menghasilkan *fitness* setiap inidividu pada populasi yang diperoleh dari nilai *Root Mean Squared Error* (RMSE) pada metode ELM. Nilai RMSE pada ELM diperoleh dari data peramalan yang dibandingkan dengan data nyata. Setelah mencapai pada generasi tertentu, maka nilai yang terakhir dengan *fitness* tertinggi akan diambil dan dijadikan sebagai *input weight* untuk proses peramalan. Hasil dari penelitian menunjukkan bahwa metode ELM dengan optimasi Algoritme Genetika mampu diimplementasikan degan hasil terbaik pada peramalan pertumbuhan microalga dengan nilai RMSE sebesar 0.004 dan nilai RAE sebesar 26.142% dengan 300 neuron. Sedangkan peramalan pertumbuhan microalga menggunakan ELM tanpa dioptimasi *input weight*-nya menghasilkan nilai RMSE sebesar 0.005 dan RAE sebesar 32.876% dengan 110 neuron (Purnomo, et al., 2015).

Penelitian kedua dilakukan oleh Sun, et al., (2008) tentang "Sales Forecasting Using Extreme Learning Machine With Applications In Fashion Retailing". Pada penelitian ini, mengenai peramalan penjualan fashion yang sangat dibutuhkan karena digunakan untuk mengetahui hubungan antara penjualan dan beberapa faktor seperti faktor desain. Atribut yang digunakan dalam peramalan penjualan fashion adalah tanggal, kode fashion, warna, ukuran, harga, jumlah penjualan. Metode yang digunakan peneliti adalah metode ELM. Untuk langkah awal yang dilakukan dalam penelitian ini adalah mengekstrak data penjualan satu jenis fashion dari data mentah. Faktor-faktor yang paling signifikan dipilih menjadi *input* di ELM. Kemudian untuk outputnya berupa jumlah penjualan fashion. Selanjutnya data dipilih untuk dijadikan sebagai data *training* dan diambil 20% dari data untuk data *testing*. Setelah itu data *training* di normalisasi masing-masing *input*nya dalam kisaran tertentu. Selanjutnya langkah paling akhir adalah di denormalisasi

untuk dikembalikan ke data awal. Hasil dari penelitian ini menunjukkan bahwa metode yang digunakan mengungguli beberapa metode peramalan penjualan yang didasarkan pada jaringan saraf backpropagation (BPNN). Peramalan penjualan yang efektif dapat membantu mengambil keputusan dalam menghitung biaya produksi, material, dan dapat menentukan harga jual. Selain itu, hasil peramalan dari ELM lebih stabil dari pada BPNN.

Untuk penelitian yang ketiga tentang "Optimasi Penjadwalan Proyek Dengan Penyeimbangan Biaya Menggunakan Kombinasi CPM Dan Algoritme Genetika". Penelitian ini mengenai manajemen penjadwalan proyek dengan memperhatikan pemakaian waktu dan sumber daya yang tersedia, tetapi kesesuaian presedensi diantara kegiatan tetap terpenuhi. Pada penelitian ini bertujuan untuk mencari nilai yang optimal dari penjadwalan proyek dengan mempertimbangkan faktor-faktornya seperti durasi kerja, predesesor, dan biaya yang diberikan kepada karyawan. Hasil dari penelitian ini didapatkan bahwa perpaduan antara metode CPM dengan Algoritme Genetika yaitu menghasilkan suatu penjadwalan proyek dengan orientasi waktu dan biaya proyek. Jadi, algoritme genetika dan CPM memberikan hasil yang akurat dalam penentuan jadwal yang optimal sehingga cocok untuk para kontraktor dalam melakukan penjadwalan proyek (Arifudin, 2012).

Penelitian yang keempat mengenai "Analisis Pengendalian Persediaan Produk Dengan Metode EOQ Menggunakan Algoritma Genetika Untuk Mengefisiensikan Biaya Persediaan". Permasalahan dari penelitian ini adalah tidak tersedianya sistem pengendalian persedian barang sehingga menimbulnya dana yang dikeluarkan menjadi terlalu besar. Selain itu, kerusakan kualitas barang juga menjadi tinggi peluangnya. Pada penelitian ini, metode *economic order quantity* (EOQ) dengan metode algoritme genetika di padukan untuk menyelesaikan permasalah yang terjadi di PT XYZ. Pada algoritme genetika mempunyai keunggulan lebih fleksibilitas terhadap permasalahan dan bersifat kemungkinan-kemungkinan untuk mendapatkan suatu solusi yang optimal dari kandidat-kandidat yang diberikan. Sedangkan untuk metode EOQ adalah metode yang digunakan untuk pembelian bersama dengan beberapa jenis item. Faktor-faktor yang dipertimbangkan dalam metode EOQ adalah biaya pemesanan, biaya penyimpanan, dan biaya pembelian. Langkah pertama akan mencari nilai EOQ yang optimal kemudian nilai EOQ dibandingkan dengan EOQ yang dioptimalkan dengan algoritme genetika. Kesimpulan dari penelitian ini adalah memberikan nilai validasi menggunakan persamaan barlas lebih kecil dari 30%. Meskipun dari beberapa pergantian variabel populasi hasil awal dengan pergantian populasi hasil akhir menunjukkan hasil yang tidak jauh berbeda (Indroprasto & Suryani, 2012).

Dengan beberapa referensi diatas, maka dapat digunakan untuk penelitian peramalan di hari berikutnya menggunakan metode *Extreme Learning Machine* (ELM) dengan mengoptimalkan parameter-parameter pemakaian air menggunakan Algoritme Genetika. Pada Tabel 2.1 akan ditunjukkan metode beserta objek sebagai acuan perbandingan untuk penelitian peneliti.

Tabel 2.1 Tinjauan pustaka

No	Penulis	Objek	Metode	Hasil
1	(Purnomo, et al., 2015)	Objek: Pertumbuhan Microalga	Metode *Extreme Learning Machine* (ELM) dan Algoritme Genetika	Hasil yang diberikan menunjukkan bahwa *Hybred* Algorime Genetika dapat memberikan nilai error lebih kecil dibandingan hanya menggunakan metode ELM saja.
2	(Sun, et al., 2008)	Objek: Penjualan fashion Atribut: Tanggal, kode fashion, warna, ukuran, harga, jumlah penjualan.	*Extreme Learning Machine* (ELM)	Hasil penelitan menggunakan *Extreme Learning Machine* dapat membantu mangambil keputusan dalam biaya produksi material, dan menentukan harga jual. Metode *Extreme Learning Machine* lebih efisien dibandingkan metode backpropagation.
3	(Arifudin, 2012)	Objek: Penjadwalan Proyek	CPM dan Algoritme Genetika	Metode CPM dan Algoritme Genetika menunjukkan bahwa metode tersebut mampu memberikan hasil akurasi dalam penentuan jadwal yang optimal. Sehingga pada penelitian ini memberikan solusi kepada para kontraktor

Tabel 2.1 Tinjauan pustaka (Lanjutan)

No	Penulis	Objek	Metode	Hasil
4	(Indroprasto & Suryani, 2012)	Objek: Persediaan Produk	Algoritme Genetika	Hasil menunjukkan bahwa nilai validasi menggunakan persamaan barlas lebih kecil dari 30% yang telah di uji dengan mengganti beberapa nilai variable pada populasi awal
5	(Penulis, 2018)	Objek: Pemakaian Air Pada Pembangkitan Listrik Tenaga Gas Dan Uap (PLTGU)	*Extreme Learning Machine* Dengan Optimasi Algoritme Genetika	Metode *Extreme Learning Machine* Dengan Optimasi Algoritme Genetika dapat menyelesaikan peramalan dari pemakaian air di Pembangkitan Listrik Unit Gresik

1.2 Air

Air adalah sumber daya yang sangat bermanfaat bagi kehidupan mahkluk hidup, sebagian besar kegiatan manusia membutuhkan air sehingga kebutuhan air dipandang sangat mutlak untuk dipenuhi (Saparuddin, 2010). Air merupakan sumber daya yang tidak hanya mendukung dari sektor rumah tangga saja, melainkan dari sektor pertanian, peternakan, transportasi, bahkan pembangkit listrik. Biasanya kegunaan air untuk pendukung pertanian digunakan untuk pemeliharaan dan penanaman. Sedangkan untuk peternakan digunakan untuk membersihkan kotoran, minum ternak, serta untuk memandikan ternah. Untuk pembangkitan listrik, air diolah menjadi uap yang dipanaskan oleh gas yang digunakan untuk memutarkan turbin kemudian diolah di generator untuk diijadikan listrik.

Menurut peraturan Pemerintahan Republik Indonesia Nomer 82 Tahun 2001 Tentang "Pengelolaan Kualias Air serta Pengendalian Pencemaran Air", pengertian air adalah komponen lingkungan hidup yang penting bagi kelangsungan kehidupan manusia dan makluk hidup lainnya, serta dapat memajukan kesejahteraan manusia. Salah satu dari kesejahteraan yang diberikan dari fungsi air adalah dapat mendukung aktifitas manusia dalam kehidupan sehari-hari seperti pemanfaatan air untuk pembangikitan listrik tenaga uap dan gas (PLTGU).

Salah satu jenis air yang memiliki kapasitas banyak adalah air laut. Air laut merupakan air yang mempunyai volume terbesar dari sumber air yang lainnya.

Kandungan salinitas dalam air laut sangat tinggi sehingga mempengaruhi kelayakan air untuk digunakan dalam kebutuhan sehari-hari (Mishra & Ramarabhu, 2011). Kegunaan air laut tidak hanya untuk tempat berlangsungnya kehidupan hewan laut, melainkan fungsi dari air laut yang sangat membantu manusia yaitu transportasi. Banyak pelabuhan yang digunakan pusat transportasi antar pulau, selain itu air laut juga bisa diolah menjadi pembangkit listrik (Yuningsih & Masduki, 2011).

1.3 Peramalan

Peramalan atau prediksi adalah proses untuk memperkirakan beberapa kebutuhan dimasa mendatang yang meliputi kebutuhan kuantitatif, kebutuhan kualitatif, dan waktu (Hidayat, et al., 2013). Peramalan atau yang biasa disebut dengan *forecasting* merupakan teknik yang banyak tuntutan untuk mendapatkan informasi cepat, akurat, dan lengkap (Rozi & Sukmana, 2016). Pentingnya peramalan dalam permasalahan mengakibatkan metode peramalan semakin bertambah dan berkembang. Salah satu metode yang memiliki keunggulan dari metode peramalan yang lainnya yaitu metode *Extreme Learning Machine* (ELM). Metode tersebut mampu mengungguli metode jaringan syaraf tiruan backpropagation (Sun, et al., 2008). Sehingga banyak peneliti yang menggunakan metode ELM sebagai metode perhitungan dalam peramalan.

Pada teknik peramalan terdapat beberapa jenis yang digunakan untuk menentukan jenis penelitian peramalan (Soman & Zareipour, 2010). Berikut ini jenis-jenis dari peramalan sebagai berikut:

a. Peramalan jangka waktu sangat pendek

   Adalah peramalan yang dilakukan dalam 5 menit sampai 30 menit.

b. Peramalan jangka pendek

   Peramalan jenis ini adalah peramalan dalam waktu kurang dari 30 bulan.

c. Peramalan jangka menengah

   Adalah peramalan yang dilakukan dalam waktu 1 sampai 5 tahun.

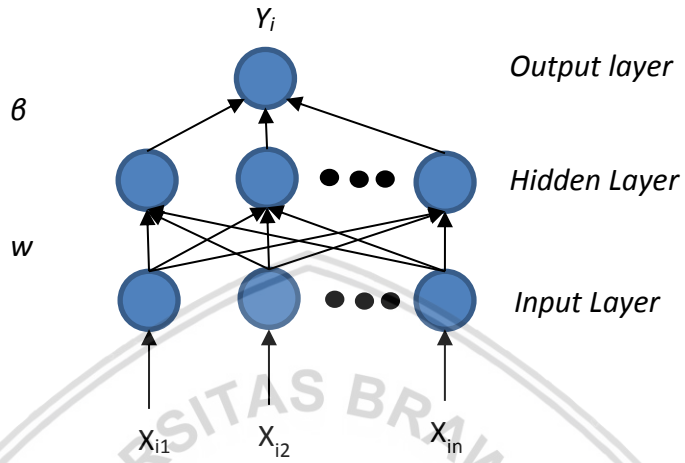
d. Peramalan jangka panjang

   Peramalan ini adalah peramalan dalam waktu lebih dari 5 tahun.

1.3.1 Metode *Extreme Learning Machine* (ELM)

Metode *Extreme Learning Machine* (ELM) adalah metode pada jaringan syaraf tiruan atau *single hidden layer feedforward neural networks* (SLFNs). Metode ELM memiliki keunggulan pada *learning speed* sehingga mampu digunakan untuk mengatasi permasalahan dalam peramalan (Huang, et al., 2006). Nilai parameter pada ELM seperti *input wight* dan *hidden bias* dipilih secara acak sehingga mempengaruhi performa pada kinerja metode ELM. ELM memiliki model matematis yang berbeda dari metode lainnya karena model tersebut lebih sederhana dan efektif. Selain itu, ELM juga memanfaatkan teori *invers* matrik

dalam proses *learning*. Teori yang digunakan yaitu *moore penrose pseudoinverse.* Secara umum, struktur model jaringan syaraf tiruan menggunakan metode *Extreme Learning Machine* (ELM) sebagai metode pembelajaran dapat dilihat pada Gambar 2.1.



Gambar 2.1 Struktur jaringan syaraf tiruan dengan ELM sebagai metode pembelajarannya

Pada proses peramalan menggunakan metode ELM terdapat pembagian data *training* dan data *testing*. Untuk pembagian data *training* dan data *testing* akan dibagi sebagai berikut:

1. Untuk data *training* mendapatkan 80% dari keseluruhan data yang digunakan untuk tahap pengembangan.
2. Untuk data *testing* mendapatkan 20% dari keseluruhan data yang digunakan untuk tahap evaluasi.

1.3.2 Tahapan dalam Proses *Training* Metode *Extreme Learning Machine* (ELM)

Tahapan dalam proses *training* menggunakan *Extreme Learning Machine* (ELM) sebagai berikut (Khotimah, et al., 2010):

1. Melakukan normalisasi pada data *training* menggunakan metode *Min Max Normalization*

   Pada tahap pertama, data akan dilakukan normalisasi dengan rentang tertentu. Fungsi aktivasi yang digunakan akan menghasilkan *output* dengan rentang data [0;1]. Berikut ini merupakan rumus yang digunakan dalam proses normalisasi ditunjukkan pada Persamaan 2.1:

$$X = \frac{(x_p - \min x_p)}{(\max x_p - \min x_p)} \quad (2.1)$$

   Keterangan:

   x = Nilai hasil normalisasi dengan rentang antara 0 sampai 1

xp = Nilai data asli sebelum dilakukan normalisasi

min xp = Nilai minimum dari dataset

max xp = Nilai maksimum dari dataset

2. Menentukan nilai input weight (W_{jk}) dan bias (b)secara *random*

   Setelah dilakukan proses normalisasi pada dataset, selanjutnya dilakukan inisialisasi *input weight* dengan ukuran matrik *j* × *k* dan nilai bias dengan ukuran 1 × *j*. nilai *j* adalah banyaknya *hidden node* sedangkan nilai *k* adalah banyaknya *input node*.

3. Menentukan fungsi aktivasi dan jumlah *hidden layer*

   Pada tahap ini, fungsi aktivasi dan jumlah *hidden layer* dihitung menggunakan rumus yang ditunjukkan pada Persamaan 2.2 sebagai berikut:

$$H = 1/(1 + EXP((-X_{train} * W^T) + b)) \quad (2.2)$$

   Keterangan:

   H = Matrik *hidden layer*

   X_{train} = Data *training*

   W^T = *Input weight transpose*

   b = Bias

4. Menentukan nilai dari matrik *Moore-Penrose Pseudo Generalized Invers*

   Untuk menghitung matrik *moore-penrose generalized invers* didapatkan dari hasil perkalian matrik *invers* dengan *transpose hidden layer.* Pada Persamaan 2.3 berikut ini merupakan cara perhitungan menghitung matrik *moore-penrose generalized invers.*

$$H^+ = (H^T . H)^{-1} * H^T \quad (2.3)$$

   Keterangan:

   H^+ = Symbol dari matrik *Moore-Penrose Pseudo Generalized Invers*

   H = Matrik dari *hidden layer*

   H^T = *Transpose* matrik dari *hidden layer*

5. Menghitung nilai dari hasil *output weight*

   Output weight didapatkan dari perhitungan *Moore-Penrose Pseudo Generalized Invers* yang dikalikan dengan matrik *transpose* matrik dari *hidden layer*. Pada Persamaan 2.4 berikut ini merupakan perhitungan *output weight.*

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

   Keterangan:

   $\hat{\beta}$ = *Output weight*

   H^+ = Matrik *moore-penrose generalized invers* dari hasil matrik *H.*

Y = Matrik target

1.3.3 Tahapan dalam Proses *Testing* Metode *Extreme Learning Machine* (ELM)

Pada tahap ini digunakan untuk mengevaluasi dari sistem yang telah dikembangkan dengan metode ELM. Pada proses *testing* menggunakan nilai dari *input weight* dan nilai dari *output weight* yang diperoleh pada proses *training.* Berikut ini tahapan pada proses testing ELM:

1. Memberikan inisialisasi pada nilai *input weight* dan nilai *bias* yang diperoleh dari proses *training.*
2. Menghitung nilai keluaran dari *hidden layer* dengan fungsi aktivasi dengan Persamaan 2.2 dimana nilai X_{train} diganti dengan X_{test}.
3. Menghitung nilai *output weight* dari hasil perhitungan *training.* Kemudian menghitung *output layer* yang dijadikan *output* peramalan ditunjukkan pada Persamaan 2.5.

$$\hat{Y} = H * \hat{\beta} \quad (2.5)$$

Keterangan:

$\hat{Y}$ = Nilai *output* dari hasil peramalan

$\hat{\beta}$ = Nilai *Output Weight* dari hasil *hidden layer* ke *output layer*

H = Nilai dari *output hidden layer* dengan fungsi aktivasi.

4. Menghitung nilai denormalisasi untuk mengembalikan nilai yang telah dinormalisasi pada tahap *training* sampai *testing* untuk menjadi data yang sebenarnya. Pada Persamaan 2.6 berikut ini merupakan cara menghitung denormalisasi data.

$$x = \big(x'(max - min)\big) + min \quad (2.6)$$

Keterangan:

x = Hasil denormalisasi data

x' = Nilai hasil peramalan sebelum didenormalisasi

max = Nilai maksimum dari dataset

min = Nilai minimum dari dataset

Setelah dilakukan *testing,* selanjutnya akan dihitung nilai kesalahan (*error*) menggunakan MAPE (*Mean Absolute Precentage Error*). Tujuan dari perhitungan MAPE adalah untuk mengetahui tingkat kesalahan absolut pada peramalan yang dibandingkan dengan nilai sebenarnya, dan dinyatakan pada Persamaan 2.7 (Hansun, 2012).

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{(Y_i - \check{Y}_i)}{Y_i}\right| \times 100 \quad (2.7)$$

Keterangan:

$\check{Y}_i$ = Nilai hasil peramalan

Y_i = Nilai aktual

n = Banyak data

1.4 Algoritme Genetika

Algoritme genetika adalah algoritme yang terinspirasi dari teori evolusi oleh Darwin, dengan mengatakan bahwa "siapa yang kuat, dia yang menang". Selain itu, suatu kelangsungan hidup dapat dipertahankan melalui proses reproduksi, proses *crossover*, dan proses mutasi (Hermawanto, 2013). Dari konsep tersebut kemudian dikembangkan menjadi sebuah metode komputasi yang lebih alamiah yaitu Algoritme Genetika. Algoritme genetika merupakan suatu teknik algoritme yang digunakan untuk mencari solusi optimal. Dalam pencarian titik optimal menggunakan fungsi propabilistik (Widodo & Mahmudy, 2010).

Ada beberapa istilah yang digunakan dalam Algoritme genetika diantaranya kromosom yang digunakan untuk memberikan solusi hasil dari komputasi algoritme genetika, yang kedua terdapat istilah populasi yaitu kumpulan dari kromosom tersebut, kemudian sebuah kromosom dapat dibentuk dari kromosom penyusun menjadi gen. kromosom tersebut akan melakukan evolusi berturut-turut yang biasa disebut dengan generasi. Setelah itu, tiap generasi kromosom dievaluasi tingkat keberhasilannya untuk dijadikan sebuah solusi baru terhadap masalah (fungsi_objektif) menggunakan teknik yang disebut dengan nilai *fitness*. Pada algoritme genetika dapat memberikan hasil berupa nilai *real*, biner, simbol, serta karakter tergantung dari permasalahannya. Pada tahap pemilihan kromosom supaya dapat dipertahankan untuk generasi selanjutnya maka akan dilakukan pada proses seleksi. Pada tahap seleksi, proses yang dilakukan menggunakan teori Darwin yaitu kromosom yang memiliki nilai *fitness* yang tinggi maka akan memiliki peluang besar untuk dipilih pada generasi berikutnya (Hermawanto, 2013).

Pada tahap selanjutnya, metode algoritme genetika akan mempunyai kromosom baru yang disebut dengan nilai *offspring.* Selanjutnya kromosom tersebut dibentuk melalui perkawinan dalam satu generasi yang disebut dengan *crossover.* Kemudian, jika terdapat jumlah kromosom dalam populasi mengalami crossover yang telah ditentukan oleh parameter maka disebut dengan *crossover_rate.* Untuk perubahan unsur penyusun akibat dari faktor alam yang direpresentasikan sebagai proses berubahnya nilai satu atau lebih nilai gen dalam kromosom dengan nilai acak disebut dengan mutasi. Setelah beberapa tahap generasi maka akan dihasilkan kromosom yang mempunyai nilai gen konvergen terhadap suatu nilai tertentu untuk dijadikan sebuah solusi optimal yang dihasilkan oleh metode algorima genetika (Hermawanto, 2013).

1.5 Tahapan dalam Proses Algoritme Genetika

Kemampuan kerja algoritme genetika didasarkan pada teori Darwin, yaitu berdasarkan ketahanan hidup yang paling kuat (Malhotra, et al., 2011). Di dalam algoritme genetika mengandung kromosom, gen, kumpulan populasi, *fitness*, fungsi *fitness*, berkembangbiak, mutasi, serta serangkaian solusi yang ditunjukkan oleh kromosom. Solusi dari satu populasi akan digunakan untuk membentuk populasi baru, dimana tidak menutup kemungkinan bahwa populasi baru lebih baik dari populasi yang lama. Kemudian solusi akan dipilih dengan membentuk solusi baru yaitu *offspring*.

Secara garis besar, langkah-langkah dalam metode algoritme genetika sebagai berikut (Malhotra, et al., 2011):

1. Melakukan pengkodean

   Pengkodean pada Algoritme Genetika bermacam-macam sesuai dengan permasalahan yang diteliti, diantaranya:

   a. Pengkodean biner

      Pada pengkodean biner terdapat angka 1 dan 0 untuk membangkitkan himpunan solusi secara *random*.

   b. Pengkodean bilangan real (*real code*)

      Pengkodean bilangan *real* digunakan untuk menyelesaikan masalah kompleks dan membutuhkan banyak generasi dalam optimasinya. Pada pengkodean bilangan *real* dapat melakukan transformasi dari bilangan biner ke bilangan decimal atau sebaliknya.

2. Menginisialisasi populasi awal

   Dalam menginisialisasi dilakukan dengan membangkitkan sejumlah kromosom (sesuai dengan ukuran populasi) yang akan dijadikan menjadi anggota populasi awal.

3. Pembentukan individu baru

   Berikut ini merupakan proses reproduksi pada Algoritme Genetika sebagai berikut:

   a. Crossover

      Crossover berfungsi untuk mengkombinasikan dua kromosom *parent* (induk) berdasarkan probabilitas pada *crossover* yang akan menghasilkan *offspring*. Metode *crossover* yang digunakan adalah *extended intermediate crossover* untuk bilangan *real code*. Persamaan 2.8 dibawah ini menunjukkan proses perhitungan pada *extended intermediate crossover.*

$$C_1 = P_1 + \alpha\,(P_2 - P_1) \qquad (2.8)$$
$$C_2 = P_2 + \alpha\,(P_1 - P_2)$$

Keterangan:

C_1 = *Crossover* untuk *parent* 1

C_2 = *Crossover* untuk *parent* 2

P_1 = Nilai dari *parent* 1

P_2 = Nilai dari *parent* 2

α = Nilai alpha dengan *range* [-0.25;1.25]

b. Mutasi

Pada tahap mutasi mengubah jumlah gen berdasarkan nilai probabilitas mutasinya untuk menghasilkan kromosom baru. Proses perhitungan mutase untuk bilangan *real code* menggunakan metode *random mutation*. Berikut ini merupakan proses perhitungan mutase ditunjukkan pada Persamaan 2.9.

$$x_1' = x_1' + r\,(max_i - min_i) \tag{2.9}$$

Keterangan:

x_1' = Nilai parent yang akan dilakukan mutasi

r = Nilai *random* dengan *range* [-0.1;0.1]

max_i = Nilai maksimal dari batas yang telah ditentukan

min_i = Nilai minimum dari batas yang telah ditentukan

4. Mengevaluasi nilai *fitness*

Untuk mengevaluasi nilai *fitness* maka setiap kromosom pada populasi dihitung nilai *fitness*-nya berdasarkan fungsi *fitness*. Berikut merupakan fungsi *fitness* yang ditunjukkan pada Persamaan 2.10 (Istiqara, et al., 2018).

$$fitness = \frac{1}{(1+ MAPE)} \tag{2.10}$$

Dari persamaan 2.5 menyatakan bahwa nilai *fitness* ditentukan oleh nilai dari pinalti (jumlah pelanggaran). Semakin tinggi nilai *fitness* maka akan semakin tinggi untuk dipilih ke genearasi selanjutnya.

5. Seleksi individu

Tujuan dari seleksi individu adalah untuk mendapatkan individu terbaik dari populasi dan akan dipertahankan untuk generasi selanjutnya sesui dengan jumlah *offspring*. Pada proses seleksi terdapat beberapa metode diantaranya:

a. Seleksi *Elitism*

Cara kerja pada seleksi *elitism* yaitu dengan mengurutkan individu yang memiliki nilai *fitness* tertinggi ke nilai *fitness* terendah kemudian akan diambil sejumlah *popsize*.

b. Seleksi *Roulette Wheel*

Seleksi *roulette wheel* adalah seleksi dengan menghitung probabilitas pada setiap individu berdasarkan nilai *fitness*. Nilai pada probabilitas tersebut merupakan nilai peluang yang dimiliki oleh individu terpilih. Perhitungan pada probabilitas dapat menggunakan probabilitas komulatif untuk seleksi individu pada populasi.

Berikut ini Persamaan yang digunakan untuk menghitung total nilai *fitness* dari seluruh individu ditunjukkan oleh Persamaan 2.11.

$$Total\ Fitnes = \sum_{i=1}^{popsize} fitness(P_i) \quad (2.11)$$

Dan selanjutnya pada Persamaan 2.12 menunjukkan proses perhitungan nilai probabilitias seleksi pada masing-masing individu.

$$prob_i = \frac{fitness(P_i)}{total\ fitness},\ k = 1,2,3,\dots,popsize \quad (2.12)$$

Setelah menghitung probabilitas, maka dilakukan perhitungan probabilitas komulatif untuk setiap individu yang ditunjunkkan oleh Persamaan 2.13.

$$probCum_i = \sum_{j=1}^{i} Prob_j\ , k = 1,2,3,\dots,popsize \quad (2.13)$$

Kemudian lakukan pemutaran *roulette wheel* untuk memilih individu dengan cara sebagai berikut:

- Menentukan nilai r secara *random* dengan *range* [0;1]
- Setelah mendapatkan nilai r, individu yang terpilih adalah individu yang memiliki nilai probabilitas dari individu tersebut yang nilainya mendekati nilai r.

c. Seleksi *Replacement Selection*

Pada seleksi *replacement selection* dilakukan dengan menggantikan nilai induk dengan nilai *offspring* jika nilai *fitness* pada *offspring* lebih besar dibandingkan dengan nilai *fitness* pada induknya.

6. Meng-update Generasi

Tahap ini digunakan untuk memperbarui kromosom yang terdapat pada populasi.

Generasi pada Algoritme Genetika diulang secara terus menerus sampai mencapai kondisi berhenti. Beberapa kondisi yang digunakan untuk menghentikan generasi sebagai berikut:

- Iterasi berhenti pada generasi ke-n yang sudah ditentukan sebelumnya.
- Iterasi berhenti setelah t satuan waktu diperoleh.
- Dalam beberapa generasi yang berurutan, tidak ada peningkatan nilai *fitness* yang diharapkan.

1.6 Metode *Extreme Learning Machine* (ELM) dengan optimasi Algoritme Genetika

Metode *Extreme Learning Machine* (ELM) adalah sebuah metode yang digunakan untuk peramalan dalam sebuah permasalahan, pada metode ini memiliki keunggulan dalam kecepatan komputasi dan keakuratan dengan akurasi yang lebih baik dibandingkan dengan metode lain. Akan tetapi pada keakuratan ELM dipengaruhi oleh beberapa faktor, diantaranya banyaknya fitur, banyaknya data training, bobot *random*. Hal yang sangat mempengaruhi dalam komputasi ELM adalah menentukan bobot *random*. Dalam setiap kali dijalankan pada metode ELM memiliki hasil peramalan yang berbeda-beda karena bobot *random* akan selalu berubah-ubah. Sehingga akurasi dalam ELM bisa berubah-ubah sesuai dengan bobot *random* yang diberikan diawal. Dengan beberapa alasan tersebut, metode ELM perlu dikembangkan dengan tujuan agar bisa memiliki akurasi yang tinggi serta bobot *random* yang optimal.

Dengan permasalahan diatas, metode optimasi seperti algoritme genetika mampu mengatasi hal tersebut. Algoritme genetika digunakan sebagai metode optimasi yang dapat memberikan beberapa solusi dari solusi yang telah diberikan sebelumnya. Pada algoritme genetika sudah terbukti dalam penyelesaian optimasi seperti yang telah dijelaskan pada Sub Bab sebelumnya. Pada penggabungan metode ELM dengan algoritme genetika digunakan untuk mendapatkan bobot terbaik supaya hasil akurasi dari peramalan bisa optimal.

Berikut ini langkah-langkah perhitungan menggunakan metode *Extreme Learning Machine* dengan optimasi algoritme genetika (Yang, et al., 2013):

1. Menginisialisasi *input weight, cr, mr, popsize,* jumlah generasi.
2. Mengkodekan dengan menentukan jenis *encoding*-nya.
3. Melakukan proses reproduksi dengan cara *crossover* dan mutasi.
4. Proses seleksi kromosom berdasarkan nilai *fitness* tertinggi.
5. Setelah mendapatkan *input weight* optimal maka dilakukan perhitungan menggunakan ELM (*Extreme Learning Machine)*.
6. Melakukan proses *training* pada peramalan dengan metode ELM sebagai berikut:

   a. Menghitung *output* pada *hidden layer* (*H*).

      - Melakukan proses *weight transpose* (W^T)
      - Memulai perhitungan H_{init} dengan cara mengalikan data yang telah dipilih sebagai data *training* dengan hasil dari *weight transpose* (W^T).
      - Menghitung *output* pada *hidden layer* (*H*) dengan menggunakan Persamaan 2.2.

   b. Menghitung matrik *moore-penrose invers* sebagai berikut:

      - Melakukan proses *transpose* pada *H* yang akan disimbolkan H^T.

- Kemudian menghitung H^T dikalikan dengan nilai dari *H*.
- Setelah melakukan langkah tersebut, kemudian dilakukan *invers*.
- Menghitung matrik *moore-penrose invers* dengan Persamaan 2.3.

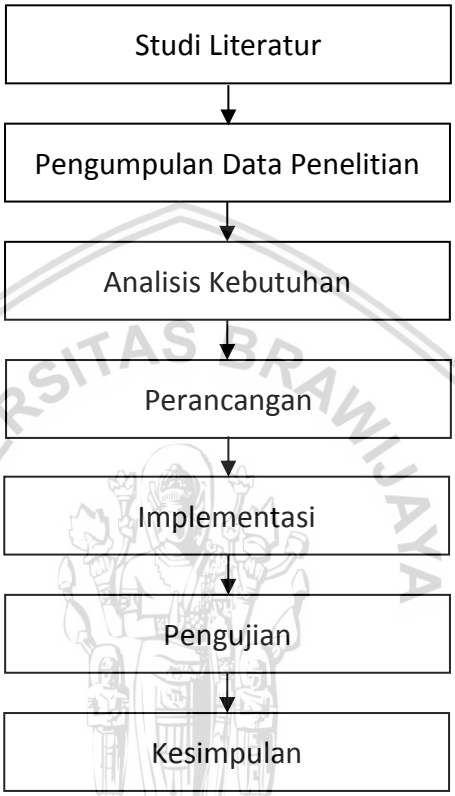
c. Menghitung nilai *output weight* dengan menggunakan Persamaan 2.4.

7. Setelah dilakukan proses *training*, maka akan dilakukan proses *testing*. Berikut ini langkah-langkah dalam proses testing pada ELM:
   a. Menghitung *output* pada *hidden layer* (*H*).
      - Melakukan proses *weight transpose* (W^T)
      - Memulai perhitungan H_{init} dengan cara mengalikan data yang telah dipilih sebagai data *training* dengan hasil dari *weight transpose* (W^T).
      - Menghitung *output* pada *hidden layer* (*H*) dengan menggunakan Persamaan 2.2.

   b. Menghitung proses peramalan pada ELM yang disimbolkan dengan $\hat{Y}$ dengan menggunakan Persamaan 2.5.
8. Menghitung nilai akurasi dengan menggunakan nilai pada MAPE yang akan digunakan sebagai nilai *fitness* untuk setiap induk pada algoritme genetika.
9. Melakukan proses evaluasi pada setiap individu.
10. Melakukan proses seleksi untuk mendapatkan nilai *fitness* tertinggi.
11. Mengulang langkah diatas sampai kondisi generasi maksimum.
12. Selesai

BAB 3 METODOLOGI

Bab ini menjelaskan langkah-langkah dalam metodologi penelitian yang terdiri dari tahapan studi literatur, pengumpulan data, analisis kebutuhan, perancangan, implementasi, pengujian, serta kesimpulan. Berikut adalah tahapan metodologi penelitian yang ditunjukkan pada Gambar 3.1.



Gambar 3.1 Diagram alir metode penelitian

1.1 Studi Literatur

Pada tahap ini, penulis melakukan studi literature yang berguna untuk referensi dalam penelitian Peramalan Pemakaian Air Pada Pembangkitan Listrik Unit Gresik Menggunakan Metode *Extreme Learning Machine* (ELM) Dengan Optimasi Algoritme Genetika. Teori pendukung diperoleh dari buku, jurnal, paper, karya tulis ilmiah, website mengenai topik-topik yang berkaitan dengan metode *Extreme Learning Machine* (ELM) dan Algoritme Genetika beserta Pemakaian Air. Referensi mengenai teori yang diperlukan untuk mendukung peneliti adalah sebagai berikut ini:

1. Peramalan (*Forcasting*)
2. Metode *Extreme Learning Machine* (ELM)
3. Algoritme Genetika
4. Pemakaian Air

1.2 Pengumpulan Data Penelitian

Pada tahap pengumpulan data digunakan untuk mengumpulkan data-data yang akan diolah untuk dilakukan penelitian serta pengujian pada sistem. Data yang digunakan untuk penelitian Peramalan Pemakaian Air Pada Pembangkitan Listrik Unit Gresik Menggunakan Metode *Extreme Learning Machine* (ELM) Dengan Optimasi Algoritme Genetika adalah sebagai berikut:

1. Data yang digunakan berasal dari jumlah produksi pemakaian air pada Pembangkitan Listrik Tenaga Gas Dan Uap (PLTGU) dimulai pada tanggal 1 januari 2017 sampai dengan 31 juli 2017.
2. Data pada penelitian ini dibagi menjadi data latih (*training)* dan data uji (*testing*).

1.3 Analisis Kebutuhan Sistem

Analisa kebutuhan sistem adalah tahap yang digunakan untuk mengetahui kebutuhan sistem dalam mengimplementasikan Peramalan Pemakaian Air Pada Pembangkitan Listrik Unit Gresik Menggunakan Metode *Extreme Learning Machine* (ELM) Dengan Optimasi Algoritme Genetika. Secara keseluruhan kebutuhan yang diperlukan dalam pembuatan sistem sebagai berikut:

1. Spesifikasi kebutuhan hardware
   - Processor intel Core™ i5-4210
   - RAM 4 GB
   - Harddisk 500 GB
2. Spesifikasi kebutuhan software
   - Sistem Operasi yang digunakan adalah Windaw 8.1
   - Bahasa pemrograman berupa java
   - Microsoft office 2010 yang digunakan sebagai pengolahan dokumen.

1.4 Perancangan

Perancangan penelitian dilakukan dengan menggunakan 2 metode, yaitu metode *extreme learning machine* (ELM) dengan Algoritme Genetika. Pada tahap perancangan akan dilakukan identifikasi formulasi permasalahan, perancangan data, perancangan algoritme, perhitungan secara manual, perancangan antarmuka.

1.5 Implementasi

Implementasi adalah tahap untuk membangun sebuah sistem dengan menerapkan pengetahuan yang telah didapatkan pada tahap studi literatur. Implementasi sistem dilakukan sesuai dengan perancangan sistem. Implementasi pada penelitian ini menggunakan bahasa java, berikut adalah implementasi sistem pada Peramalan Pemakaian Air Pada Pembangkitan Listrik Unit Gresik

Menggunakan Metode *Extreme Learning Machine* (ELM) Dengan Optimasi Algoritme Genetika sebagai berikut:

1. *User Interface* pada sistem menggunakan GUI
2. Implementasi metode Metode *Extreme Learning Machine* (ELM) Dengan Optimasi Algoritme Genetika menggunakan Bahasa java.

1.6 Pengujian

Pengujian pada penelitian ini dilakukan untuk melihat apakah sistem yang telah dibuat sesuai yang diharapakan serta untuk menguji akurasi dari sistem. Pengujian sistem yang dilakukan sebagai berikut:

1. Menguji vasiasi jumlah data *training* dan data *testing.*
2. Pengujian untuk menentukan nilai bobot *random* dari metode *Extreme Learning Machine* (ELM) yang paling optimal dalam implementasi sistem.
3. Menguji nilai akurasi dari implementasi sistem.

Untuk menguji hasil dari peramalan pemakaian air dapat dilihat dari nilai kesalahannya (*error*). Perhitungan nilai kesalahan (*error*) menggunakan teknik MAPE (*Mean Absolute Precentage Error*). Semakin kecil nilai kesalahannya maka nilai peramalan dari pemakaian air tersebut semakin baik.

1.7 Kesimpulan

Pada tahap kesimpulan dilakukan setelah tahap perancangan, implementasi sistem, dan pengujian sistem telah dilakukan. Kesimpulan diambil dari hasil analisis terhadap hasil pengujian sistem dan analisis terhadap metode *Extreme Learning Machine* (ELM) dengan optimasi Algoritme Genetika. Tahap terakhir dari penulisan pada penelitian ini adalah saran yang digunakan untuk memperbaiki kekurangan dan kesalahan serta pengembangan penelitian selanjutnya.

BAB 4 PERANCANGAN

Pada Bab ini menjelaskan deskripsi permasalahan, alur pada algoritme *Extreme Learning Machine* (ELM) dengan optimasi algoritme genetika, perhitungan manual, perancangan antarmuka, dan perancangan skenario pengujian .

1.1 Formulasi Permasalahan

Permasalahan yang diselesaikan adalah peramalan pemakaian air dengan menggunakan metode *Extreme Learning Machine* (ELM) dengan optimasi algoritme genetika. Hasil dari peramalan tersebut akan dievaluasi tingkat keakuratan menggunakan MAPE (*Mean Absolute Percentage Error*) untuk mengetahui kualitas dari hasil peramalan. Hasil peramalan yang bagus ditunjukkan dengan nilai *error* kecil.

Pada solusi yang diberikan menggunakan metode ELM yang memiliki parameter *input weight* yang didapatkan secara *random* sehingga akan di optimasi menggunakan algoritme genetika dengan tujuan bisa memberikan hasil optimal pada pembentukan nilai pada *input weight*. Nilai *input weight* pada ELM sangat mempengaruhi hasil peramalan yang didapatkan pada metode ini beserta nilai akurasinya. Pada peramalan menggunakan ELM ini akan menggunakan fitur dari jumlah produksi air dalam PLTGU UP Gresik tiap seminggu sekali. Sedangkan untuk parameter yang digunakan pada perhitungan meliputi: nilai *input weight*, jumlah generasi, bias, Cr, Mr, popsize, data *training,* serta data *testing*. Pada tahapan ELM dibagi menjadi 2 yaitu tahap *training* dan tahap *testing*. Sebelum dilakukan proses *training,* nilai *input weight* akan dilakukan optimasi terlebih dahulu menggunakan metode algoritme genetika. Pada pembentukan kromosom yang digunakan pada algoritme genetika menggunakan bilangan *real code* sehingga pada tahap *crossover* menggunakan metode *extended intermediate crossover* serta tahap mutasinya menggunakan metode *random mutation* kemudian untuk menghitung nilai *fitness* dari masing-masing individu menggunkan nilai MAPE. Kemudian pada tahap *training* akan dihitung nilai *hidden layer* dengan fungsi aktivasinya yang menghasilkan data berupa matrik. Hasil dari *output hidden layer* digunakan untuk mendapatkan nilai *output weight*. Setelah tahap tersebut maka dilakukan proses *testing* untuk mengetahui keluaran dari sistem beserta nilai kesalahannya.

1.2 Kebutuhan Data

Proses pengumpulan data didapatkan dari database pemakaian air pada PLTGU mulai tanggal 1 januari 2017 sampai 31 juli 2017 yang berjumlah 103. Pada peng-*input*-an database pemakaian air dilakukan setiap hari sekali dan peramalan pemakaian air dilakukan seminggu sekali. Terdapat 3 fitur yang akan digunakan pada proses peramalan yaitu x1, x2, x3 dan *output* berupa pemakaian air. Pada baris pertama dengan fitur x1 menandakan tanggal 2 januari 2017, x2 menandakan tanggal 4 januari, x3 menandakan tanggal 6 januari, dan seterusnya untuk baris $n + 1$.

Berikut ini adalah sampel data yang digunakan ditunjukkan pada Tabel 4.1 sebanyak 5 data.

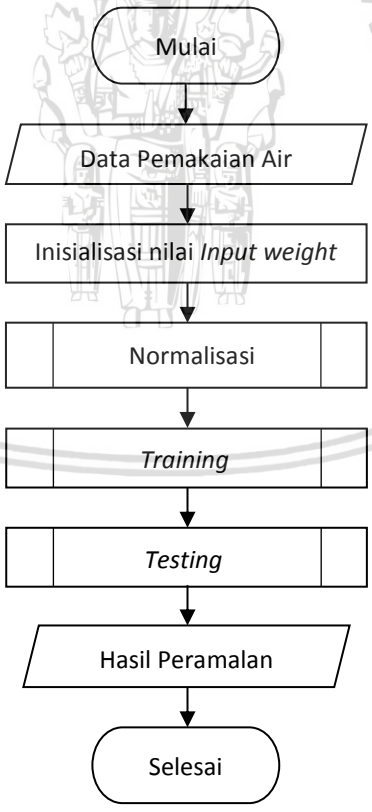
Tabel 4.1 Data jumlah pemakaian air pada PLTGU

No	Data	X1	X2	X3	Pemakaian Air
1	8 /2/2018	955867.6	867670	1905061.982	912284.2
2	10/2/2018	867670	1905061.982	912284.2	628216.8
3	12/2/2018	1905061.982	912284.2	628216.8	832743
4	14/2/2018	912284.2	628216.8	832743	827919
5	16/2/2018	628216.8	832743	827919	876182

1.3 Perancangan Algoritme

1.3.1 Alur Metode *Extreme Learning Machine* (ELM)

Pada alur metode *Extreme Learning Machine* (ELM) adalah tahapan untuk menentukan hasil peramalan pemakaian air. Berikut ini diagram alir peramalan menggunakan ELM ditunjukkan pada Gambar 4.1.



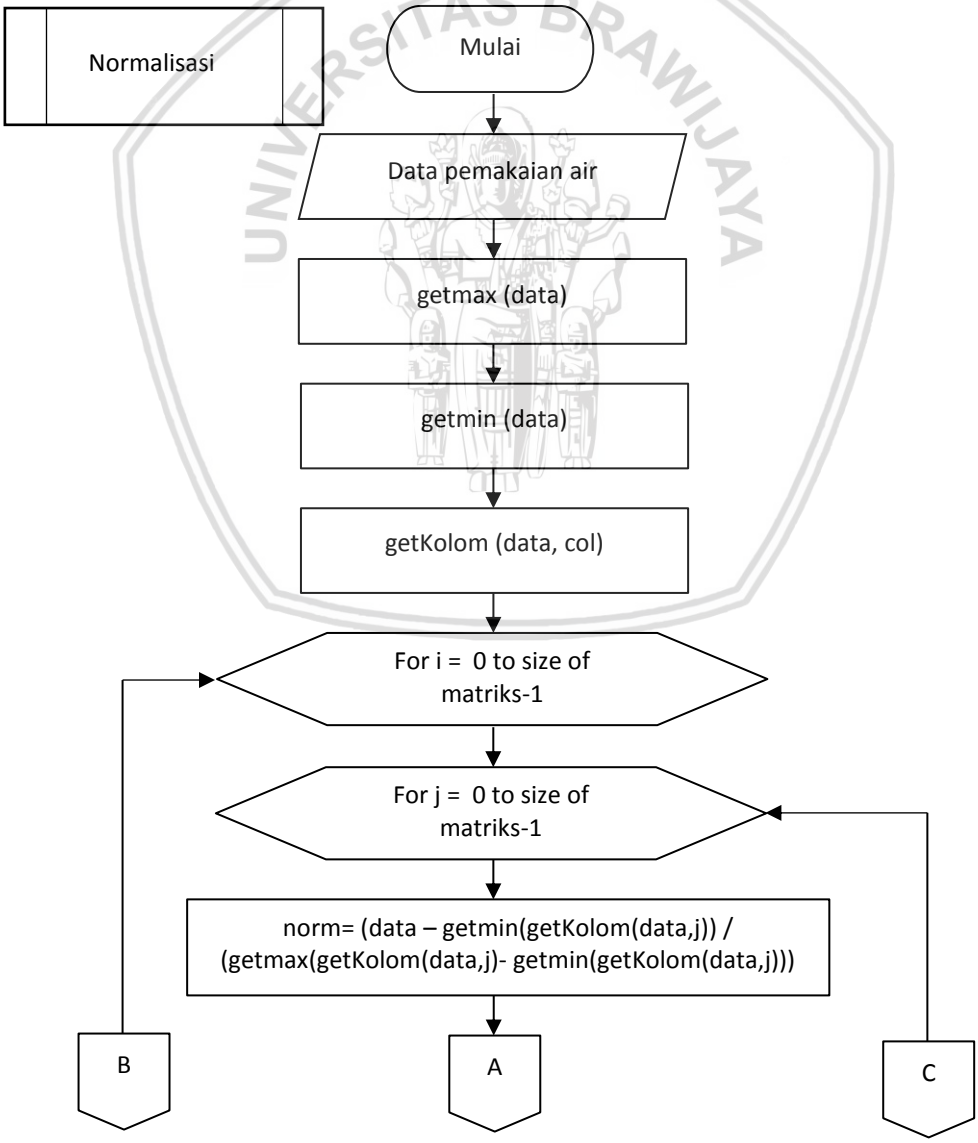
Gambar 4.1 Diagram alir sistem peramalan pemakaian air

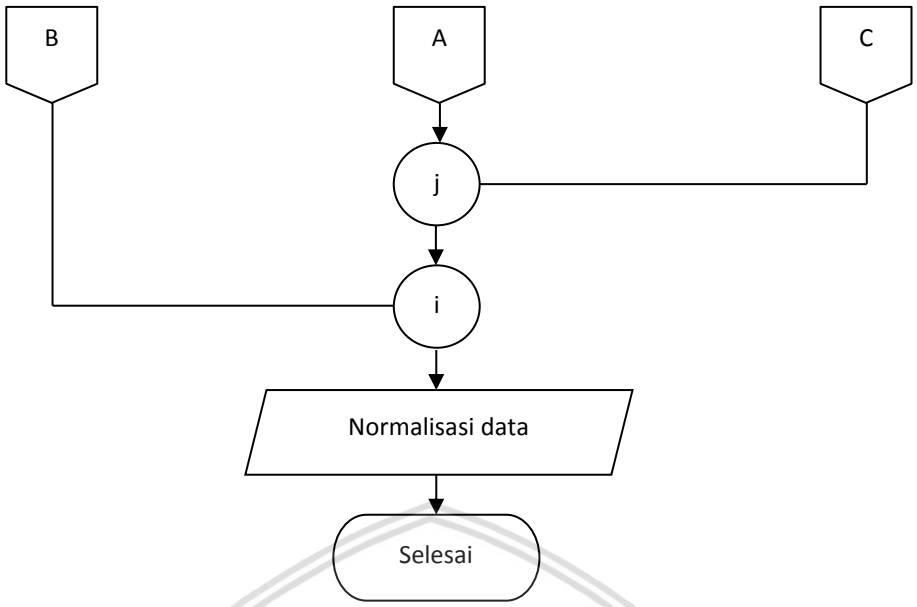
Berdasarkan Gambar 4.1 pada pemakaian air menggunakan metode ELM dapat dijelaskan langkah-langkahnya sebagai berikut:

1. Sistem menerima *input* berupa data pemakaian air mulai tanggal 1 januari 2017 sampai 31 juli 2017.
2. Melakukan normalisasi dataset dengan Persamaan 2.2 dalam *range* [0;1].
3. Memberikan inisialisasi nilai *input weight* secara *random* dengan *range* [-1;1].
4. Melakukan proses *training* dataset menggunakan ELM.
5. Melakukan proses *testing* dataset untuk mendapatkan nilai peramalannya.

1.3.1.2 Normalisasi Data dengan Min-Max Normalization

Normalisasi data dilakukan karena rentang nilai *input* tidak sama. Pada tahap normalisasi bertujuan untuk menstandarisasikan semua dataset yang digunakan untuk proses perhitungan peramalan pemakian air sehingga terdapat pada *range* tertentu. Pada normalisasi ini menggunakan Persamaan 2.1 yang telah dijelaskan pada bab sebelumnya. Berikut ini adalah diagram alir pada proses normalisasi ditunjukkan pada Gambar 4.2.





Gambar 4.2 Diagram alir normalisasi data dengan *Min-Max Normalization*

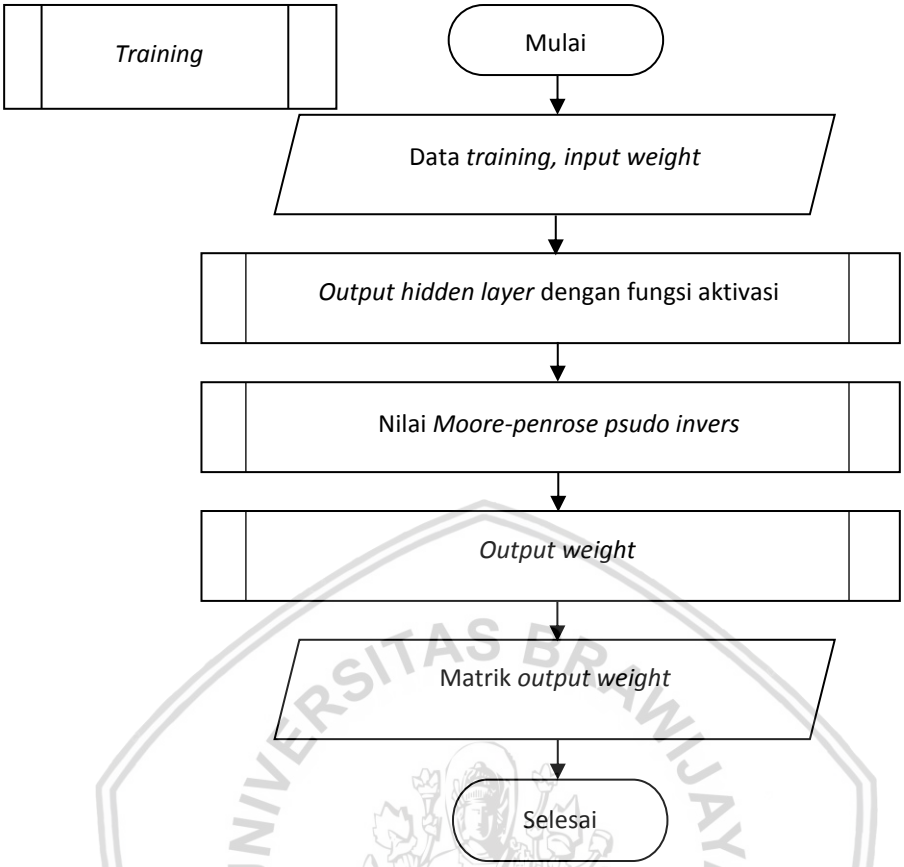
Berdasarkan Gambar 4.2 proses normalisasi, langkah-langkah pada normalisasi data pemakaian air sebagai berikut:

1. Masukan data pemakaian air pada PLTGU.
2. Mencari nilai maksimal dari masing – masing fitur.
3. Mencari nilai minimal dari masing – masing fitur.
4. Melakukan proses perhitungan normalisasi dengan Persamaan 2.1.
5. Hasil normalisasi dengan *range* [0;1]

1.3.1.3 Proses Training Extreme Learning Machine (ELM)

Pada tahap *training* data dilakukan untuk mendapatkan nilai *output weight* yang akan dilanjutkan pada proses *testing*. Langkah awal yang dilakukan pada *training* adalah mencari *output hidden layer* dengan fungsi aktivasi dengan Persamaan 2.2. Data yang dihitung adalah *input weight* yang didapatkan secara *random* dalam *range* tertentu kemudian dikalikan dengan data pemakaian air yang telah dinormalisasi. Langkah berikutnya mencari nilai *output hidden layer* yang telah dijelaskan pada Sub Bab sebelumnya. Hasil perhitungan dari *output hidden layer* digunakan untuk mencari nilai *invers* dengan metode OBE. Setelah itu pada tahap berikutnya menentukan *output weight* berdasarkan perkalian matrik *invers* dan *output weight* ke tahap *testing* ELM.

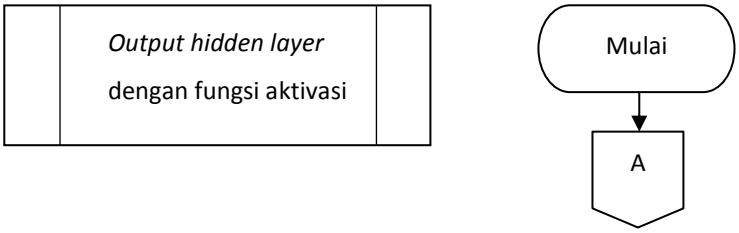
Berikut ini adalah langkah-langkah perhitungan *training* pada ELM yang ditunjukkan pada Gambar 4.3 sebagai berikut:

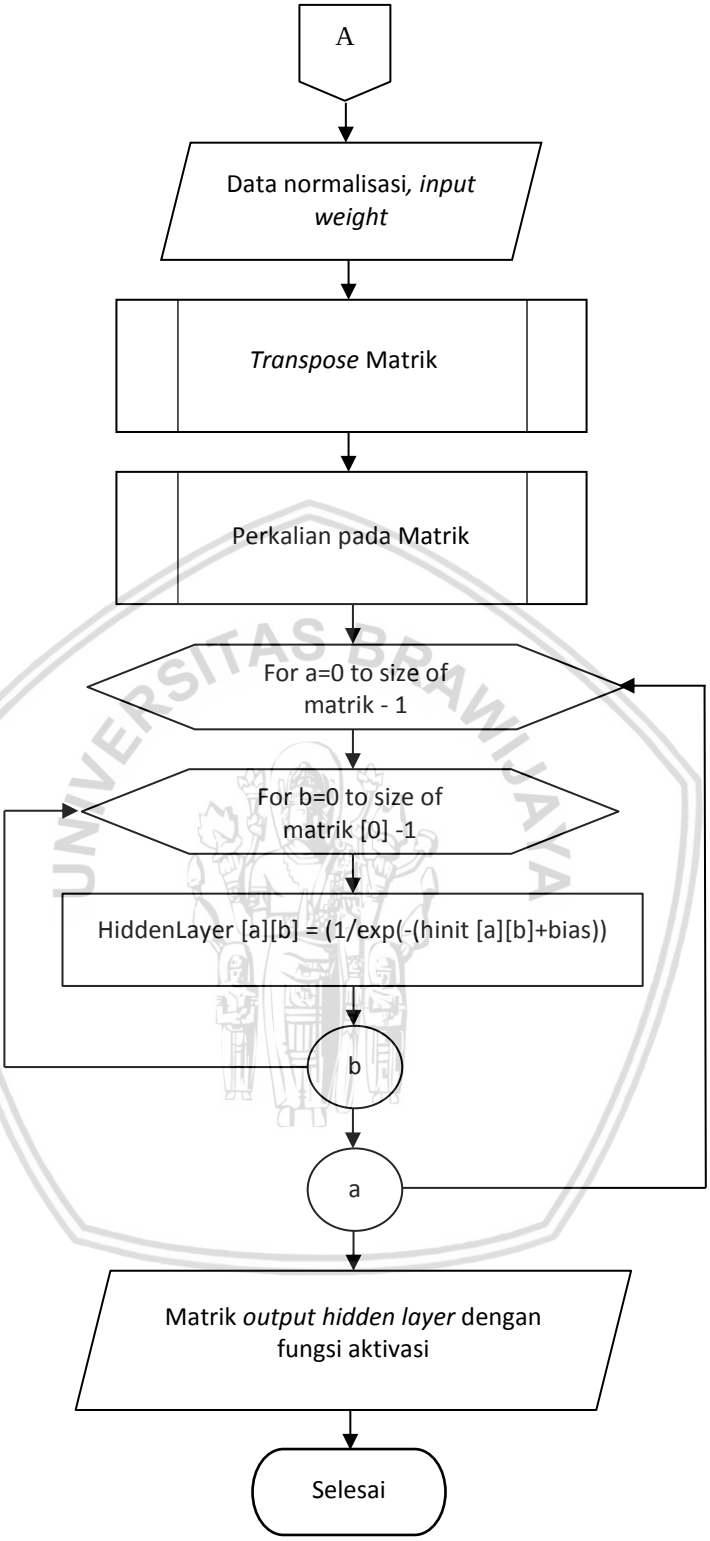


Gambar 4.3 Diagram alir *training* data pemakaian air

Pada Gambar 4.3 dijelaskan langkah-langkah proses *training* pada ELM sebagai berikut:

1. *Input* data pemakaian air yang telah dilakukan proses normalisasi, banyaknya data *training* disesuaikan dengan masukkan.
2. Memberikan inisialisasi pada nilai *input weight* yang didapatkan nilainya secara *random* dengan range tertentu.
3. Menghitung *output* pada *hidden layer* dengan fungsi aktivasi yang ditunjukkan pada Persamaan 2.2.
4. Menghitung *inverse* pada matriks dengan OBE (Operasi Basis Elementer).
5. Melakukan proses penghitungan nilai *output weight* yang didapatkan dari perkalian antara matrik invers dan matrik target. *Output weight* yang didapatkan nanti pada proses *testing* ditunjukkan pada Persamaan 2.4.



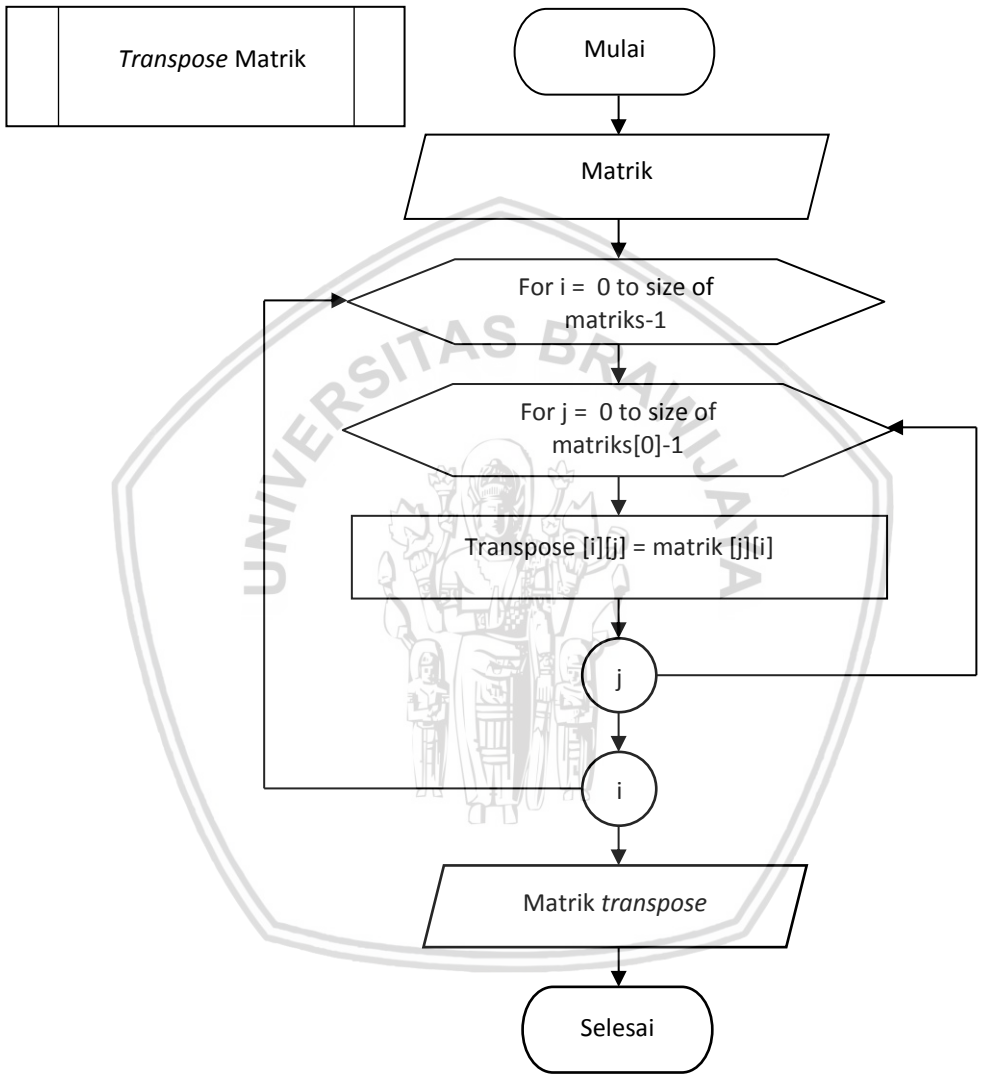


Gambar 4.4 Diagram alir proses *output hidden layer* dengan fungsi aktivasi

Pada Gambar 4.4 dijelaskan langkah-langkah proses *output hidden layer* dengan fungsi aktivasi pada ELM sebagai berikut:

1. Masukkan data normalisasi dan matrik hasil *input weight.*

2. Melakukan hasil matrik *transpose* dengan cara mengubah baris menjadi kolom dan sebaliknya pada matrik nilai *input weight.*
3. Melakukan perkalian dari data hasil normalisasi dikalikan dengan hasil *transpose input weight*.
4. Menghitung nilai *output hidden* layer dengan fungsi aktivasi ditunjukkan pada Persamaan 2.2.

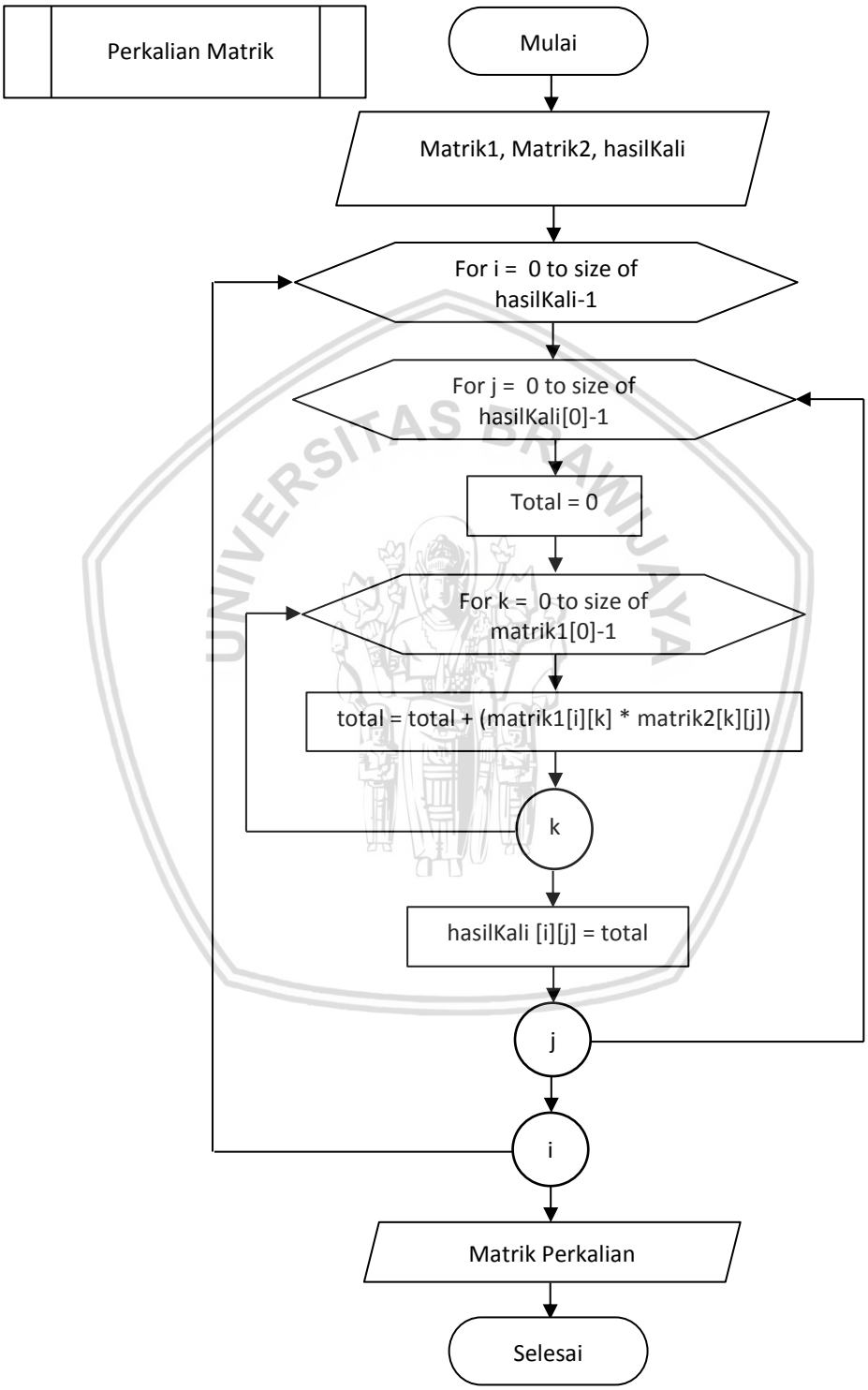


Gambar 4.5 Diagram alir proses *transpose* matrik

Pada Gambar 4.5 dijelaskan langkah-langkah proses *transpose* matrik sebagai berikut:

1. Masukkan data berupa sebuah matrik yang akan diubah menjadi *transpose* matrik.
2. Memberikan inisialisasi masukan yang terdiri dari baris dan kolom yang disesuaikan dengan panjang matrik inputan.

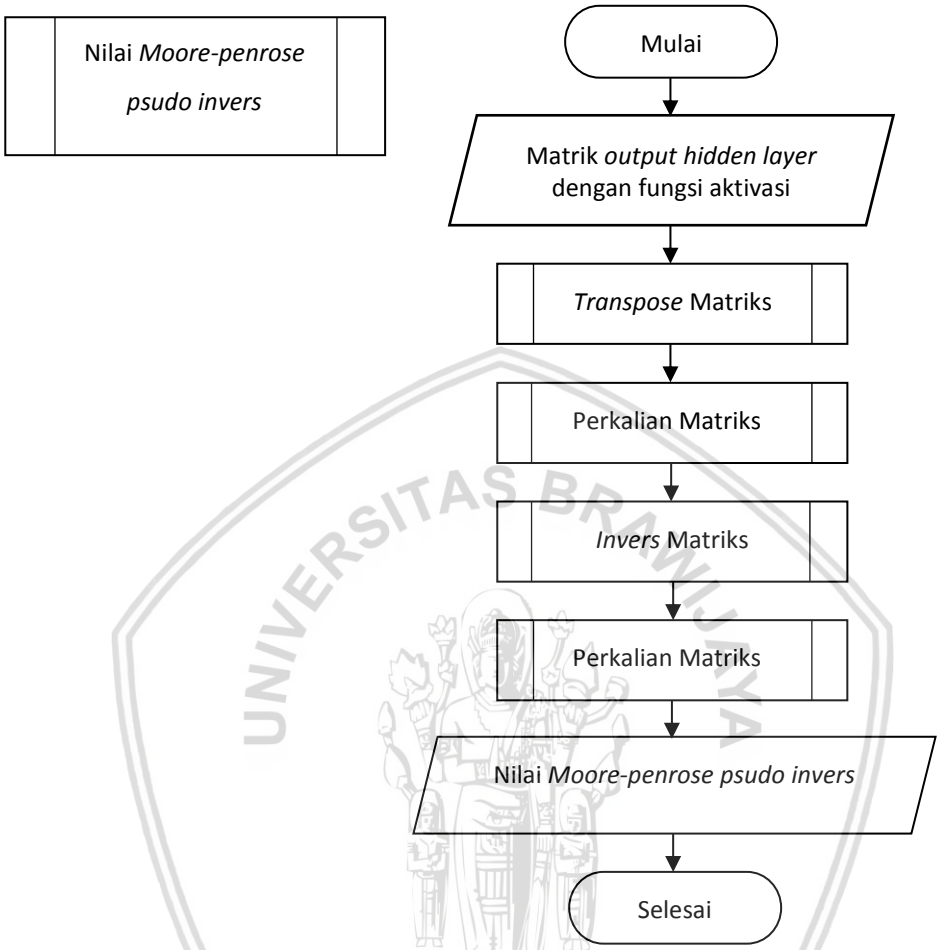
3. Melakukan perulangan dengan panjang matrik yang telah ditentukan dengan tujuan untuk memproses matrik awal menjadi *transpose* matrik.
4. Proses *transpose* matrik dengan cara mengubah baris dan kolom awal suatu variabel dengan ordo baris x kolom menjadi ordo kolom x baris.



Gambar 4.6 Diagram alir perkalian matrik

Pada Gambar 4.6 dijelaskan langkah-langkah proses perkalian matrik sebagai berikut:

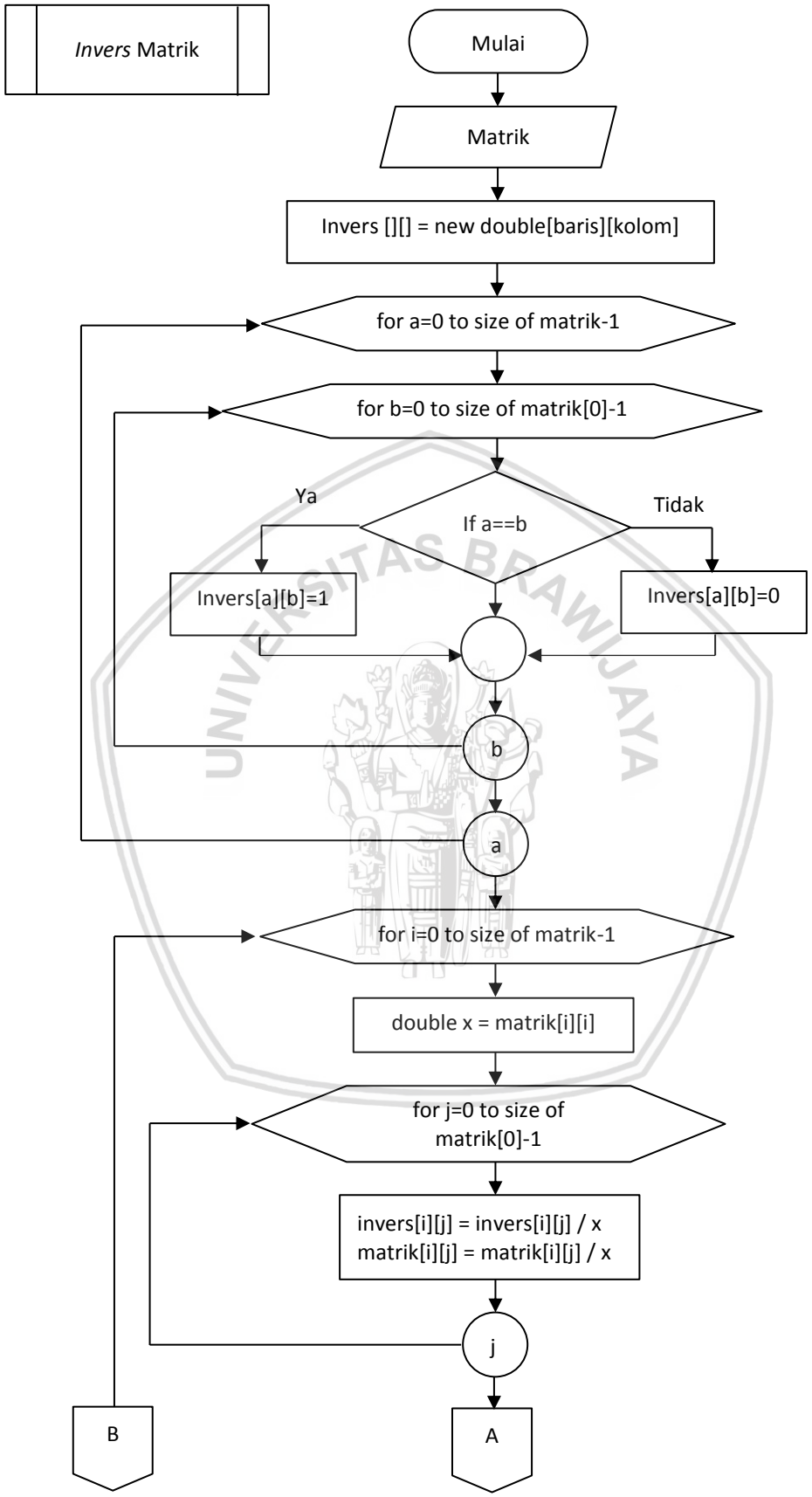
1. Masukan berupa matrik1 dengan matrik2. Kemudian kedua matrik di kalikan.

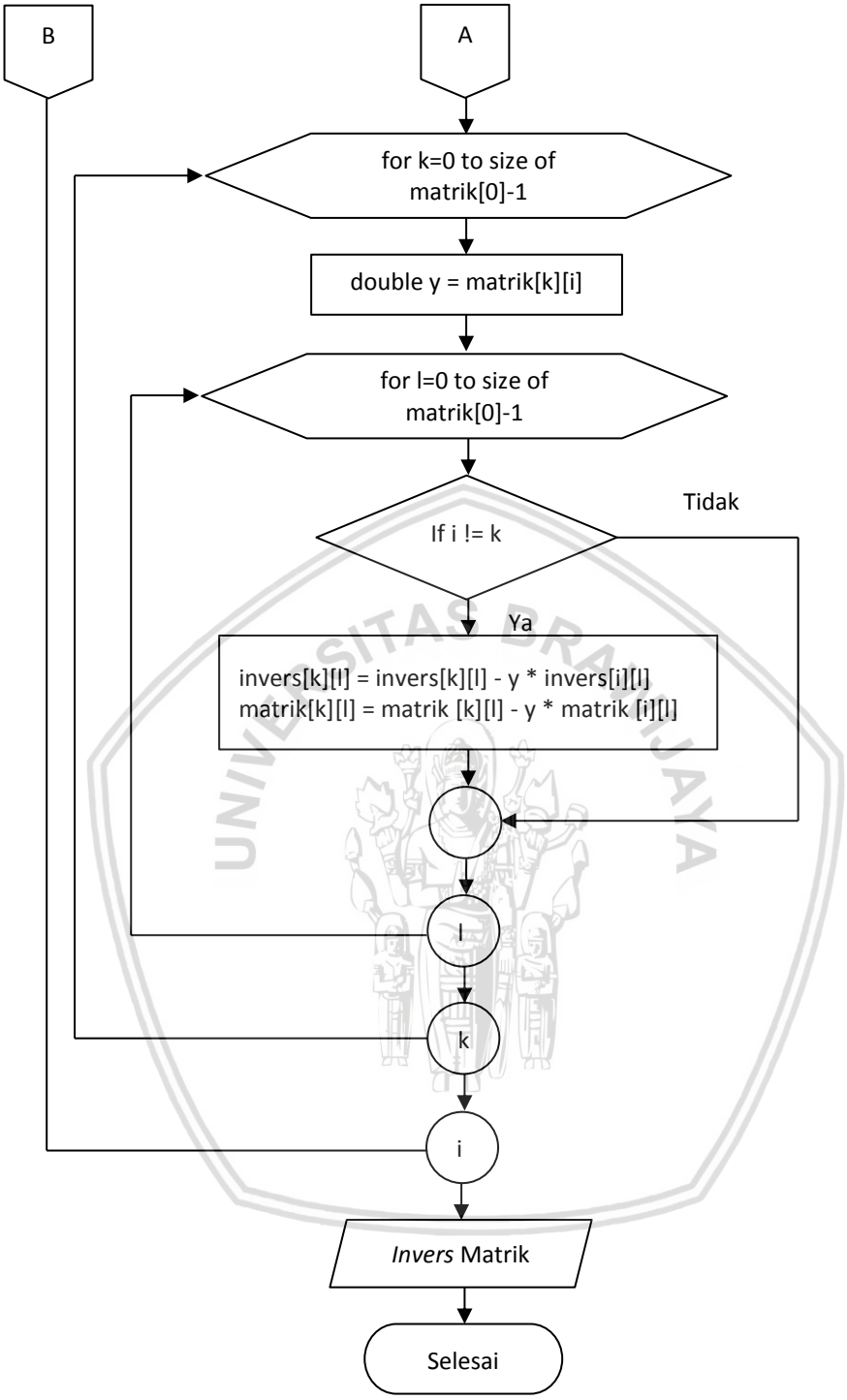


Gambar 4.7 Diagram alir *moore-penrose pseudo invers*

Pada Gambar 4.7 dijelaskan langkah-langkah *moore-penrose pseudo invers* sebagai berikut:

1. Masukan berasal dari hasil matrik *output hidden layer* dengan fungsi aktivasi.
2. Kemudian melakukan *transpose* matrik pada hasil matrik *output hidden layer* dengan fungsi aktivasi.
3. Hasil matrik *output hidden layer* dengan fungsi aktivasi dikalikan dengan hasil *transpose* matrik.
4. Melakukan *invers* matrik pada hasil perkalian matrik.
5. Melakukan perkalian matrik pada hasil *invers* matrik dengan *transpose* matrik.


Invers Matrik
Mulai
Matrik
Invers [][] = new double[baris][kolom]
for a=0 to size of matrik-1
for b=0 to size of matrik[0]-1
Ya
If a==b
Tidak
Invers[a][b]=1
Invers[a][b]=0
b
a
for i=0 to size of matrik-1
double x = matrik[i][i]
for j=0 to size of matrik[0]-1
invers[i][j] = invers[i][j] / x
matrik[i][j] = matrik[i][j] / x
j
B
A

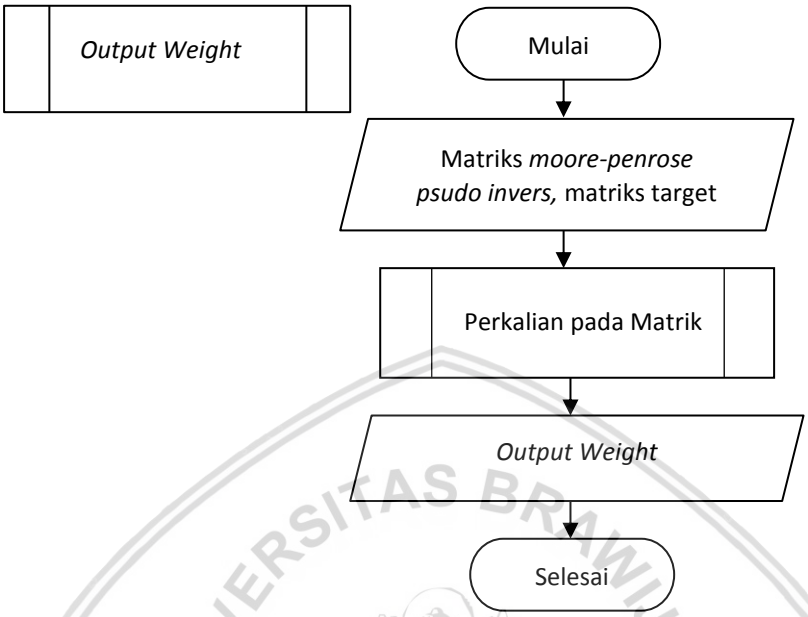


Gambar 4.8 Diagram alir *invers* matrik

Pada Gambar 4.8 dijelaskan proses invers matrik sebagai berikut:

1. Memberikan masukan berupa matrik hasil dari perkalian antara matrik *transpose output hidden layer* dengan matrik *output hidden layer*.
2. Melakukan pembuatan *array* berupa matrik identitas sepanjang baris dan kolom sesuai dengan panjang pada matrik *input*.

3. Melakukan matrik *invers* dengan cara merubah matrik identitas menjadi matrik baru dan matrik *input* menjadi matrik identitas.
4. Terbentuk matrik baru yang berasal dari matrik identitas.



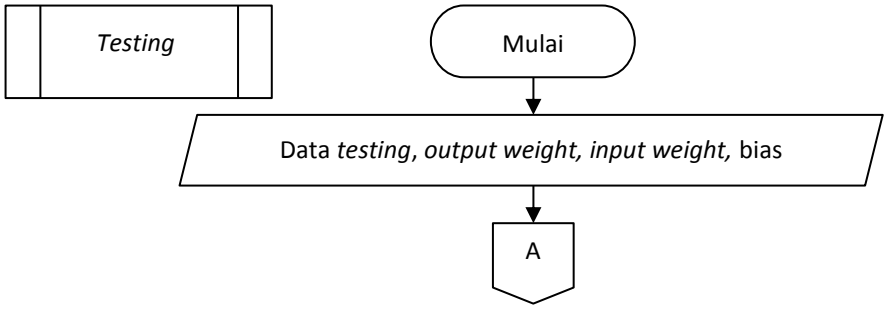
Gambar 4.9 Diagram alir proses *output weight*

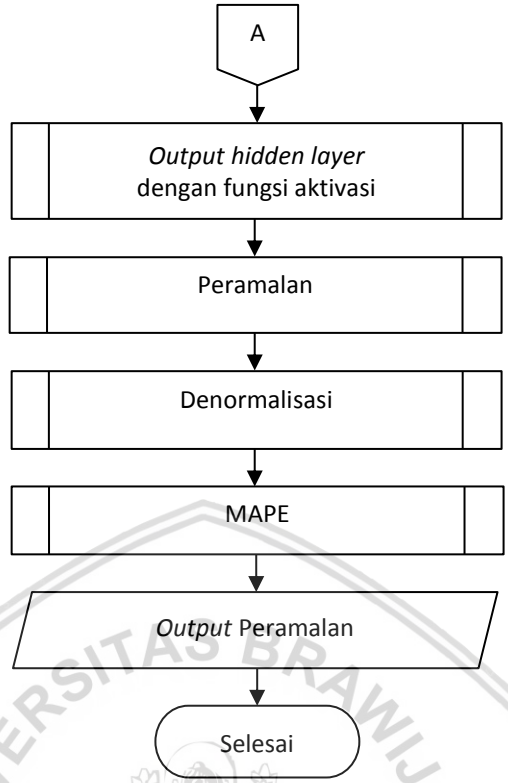
Pada Gambar 4.9 dijelaskan proses *output weight* sebagai berikut:

1. Memberikan masukan berupa nilai *invers* serta matrik target pada data *training.*
2. Melakukan perkalian matrik pada nilai *input.*

1.3.1.4 Proses Testing Extreme Learning Machine (ELM)

Pada proses *testing* dilakukan sesusia dengan tahapan pada proses sebelumnya yaitu proses *training*. *Testing* pada ELM dilakukan dengan mengambil nilai dari *output weight* pada proses *training*. Langkah awalnya sama seperti pada tahapan *training* yaitu dengan menormalisasi data *testing* kemudian menghitung nilai *hidden layer* dengan fungsi aktivasi. Kemudian dilakukan perkalian antara nilai pada *hidden layer* dengan nilai *output weight* yang didapatkan pada hasil akhir dari proses *training*. Pada Gambar 4.10 berikut ini adalah diagram alir dari proses *testing* pada ELM.

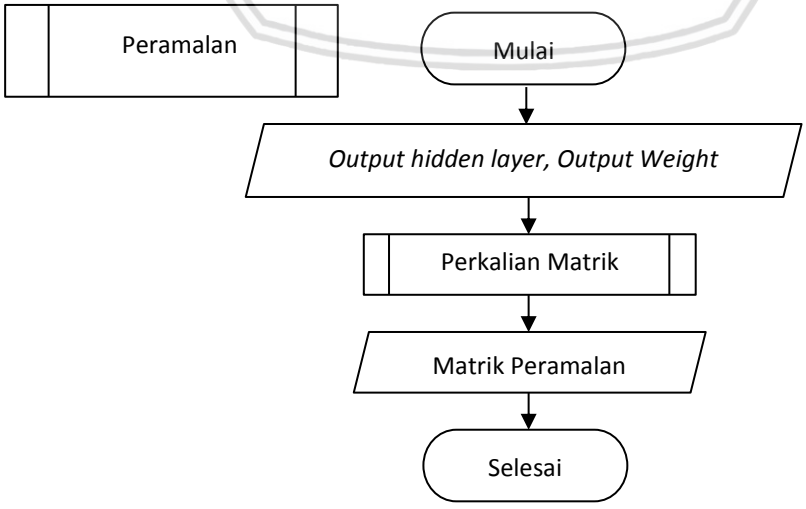




Gambar 4.10 Diagram alir proses *testing*

Pada Gambar 4.10 dijelaskan langkah-langkah perkalian matrik sebagai berikut:

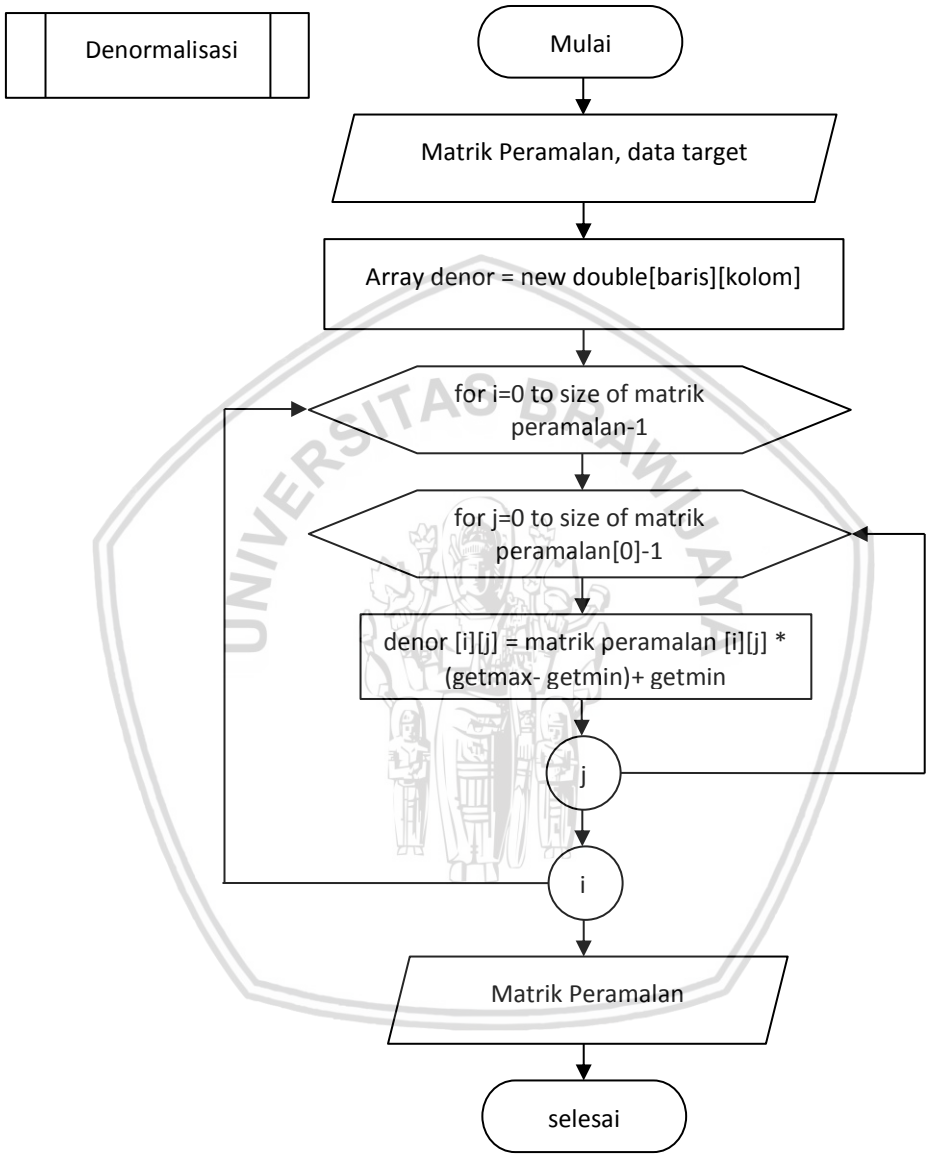
1. Masukan berupa data *testing, output weight,* dan *input weight.*
2. Melakukan proses perhitungan *output hidden layer* dengan fungsi aktivasi.
3. Melakukan peramalan pada hasil akhir dari sistem.
4. Melakukan denormalisasi pada hasil peramalan untuk dikembalikan ke nilai aslinya kemudian menentukan tingkat kesalahan sistem dengan MAPE.



Gambar 4.11 Diagram alir peramalan

Pada Gambar 4.11 dijelaskan langkah-langkah proses peramalan sebagai berikut:

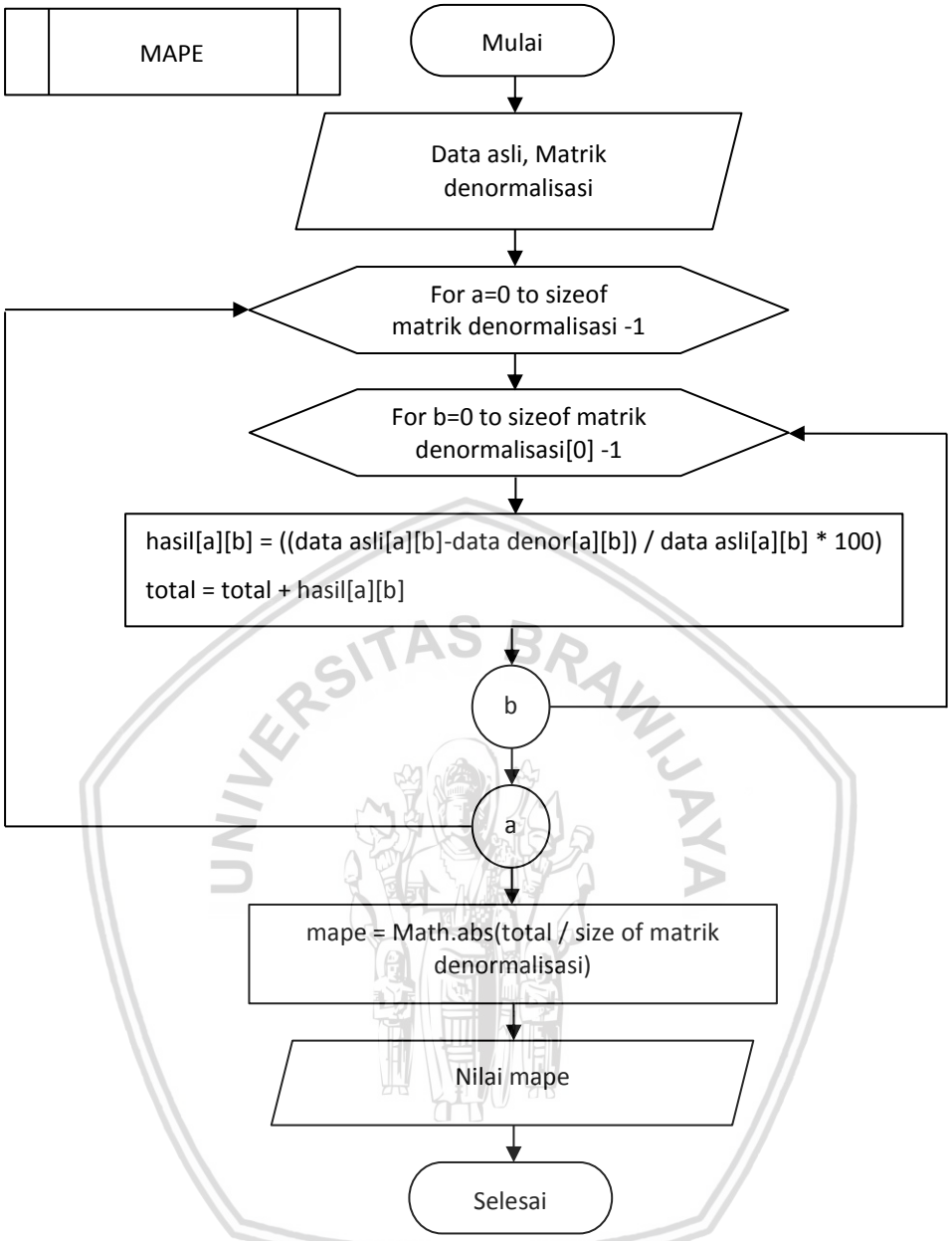
1. Melakukan *input* berupa *output hiden layer* dengan *output weight*.
2. Melakukan perkalian matrik antara matrik pada *output hiden layer* dengan *output weight* untuk menghitung peramalan.



Gambar 4.12 Diagram alir denormalisasi data

Tahapan pada proses denormalisasi data ditunjukkan oleh Gambar 4.12 sebagai berikut:

1. Melakukan *input* berupa matrik peramalan.
2. Melakukan perhitungan denormalisasi data untuk merubah data kembali menjadi data asli.



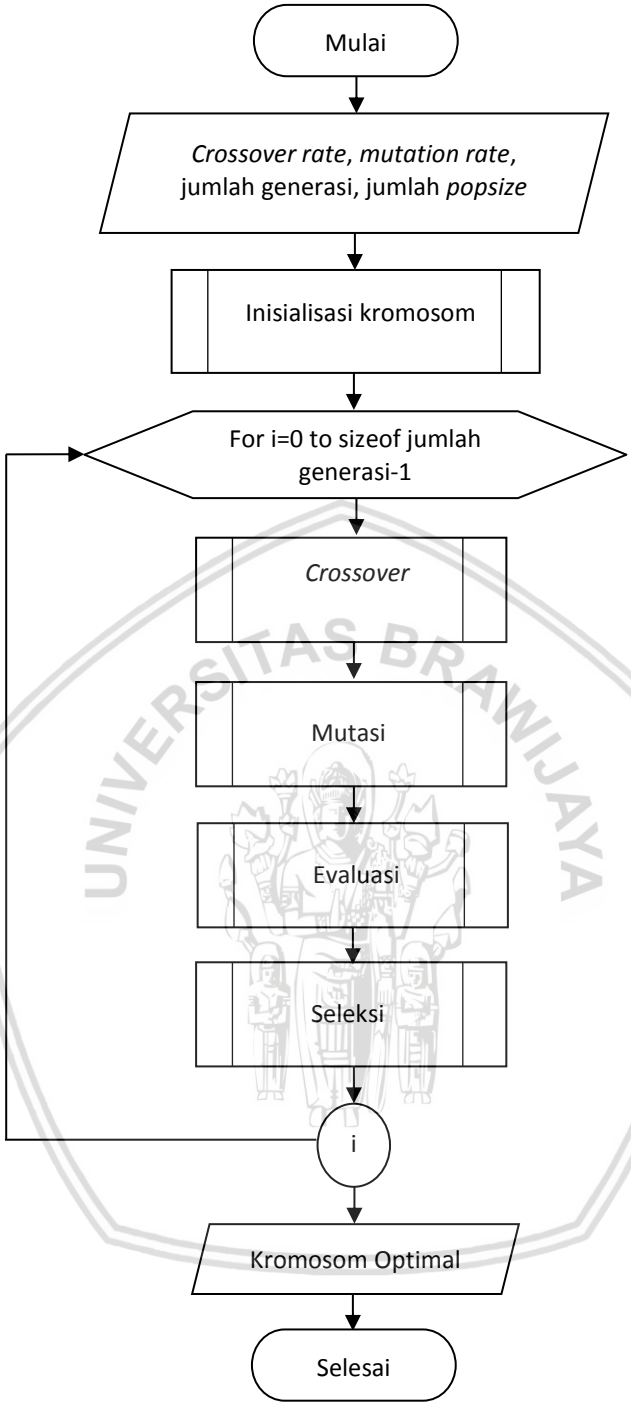
Gambar 4.13 Diagram alir proses MAPE

Tahapan pada proses perhitungan MAPE data ditunjukkan oleh Gambar 4.13 sebagai berikut:

1. Melakukan *input* berupa data asli dengan data yang telah di denormalisasi.
2. Melakukan perhitungan MAPE.

1.3.2 Alur Algoritme Genetika

Algoritme genetika digunakan untuk mengoptimalkan nilai *input weight* pada metode ELM. Langkah - langkah penyelesaian menggunakan algoritme genetika ditunjukkan pada Gambar 4.14 adalah sebagai berikut:

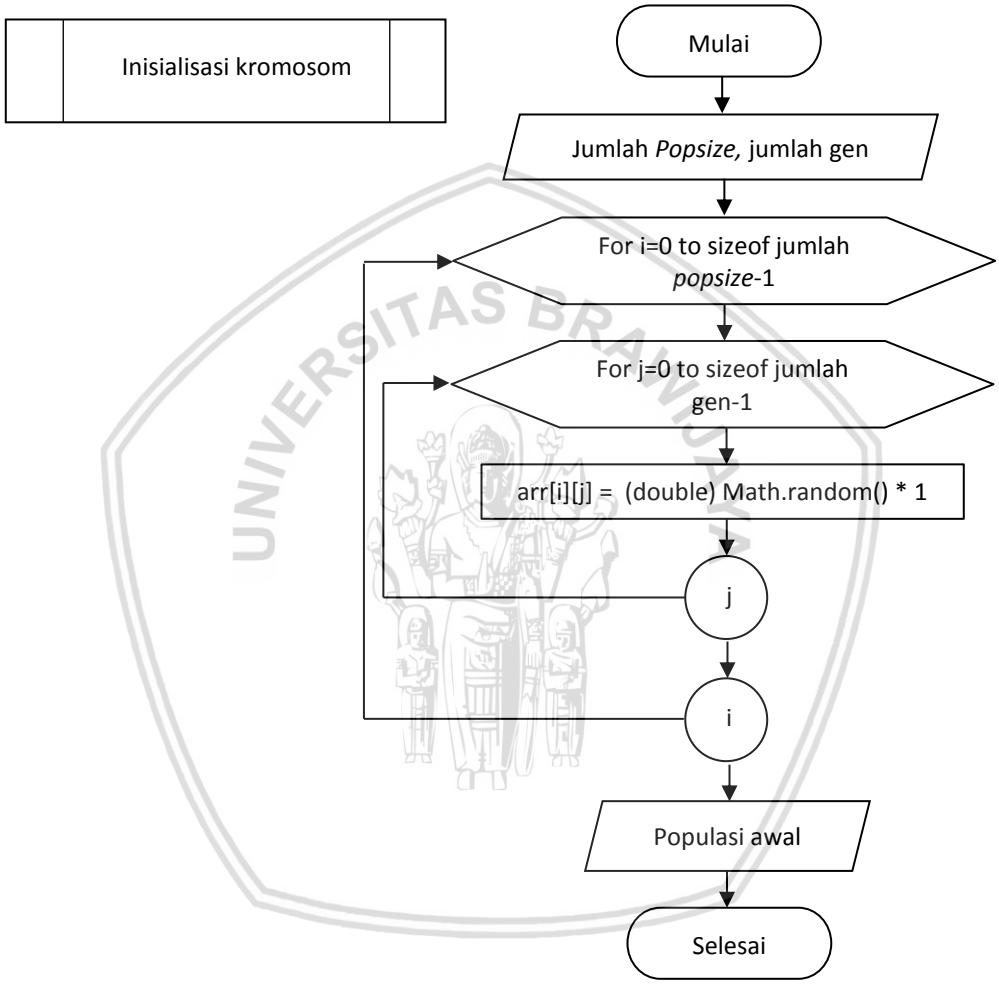


Gambar 4.14 Diagram alir proses algoritme genetika

Pada Gambar 4.14 dijelaskan langkah-langkah proses algoritme genetika sebagai berikut:

1. Memberikan masukan berupa inisialisasi nilai *crossover rate* dan *mutation rate* dan generasi maksimum.
2. Memberikan inisialisasi pada parent sejumlah *popsize*.
3. Melakukan proses *crossover* pada induk yang dipilih secara *random*.

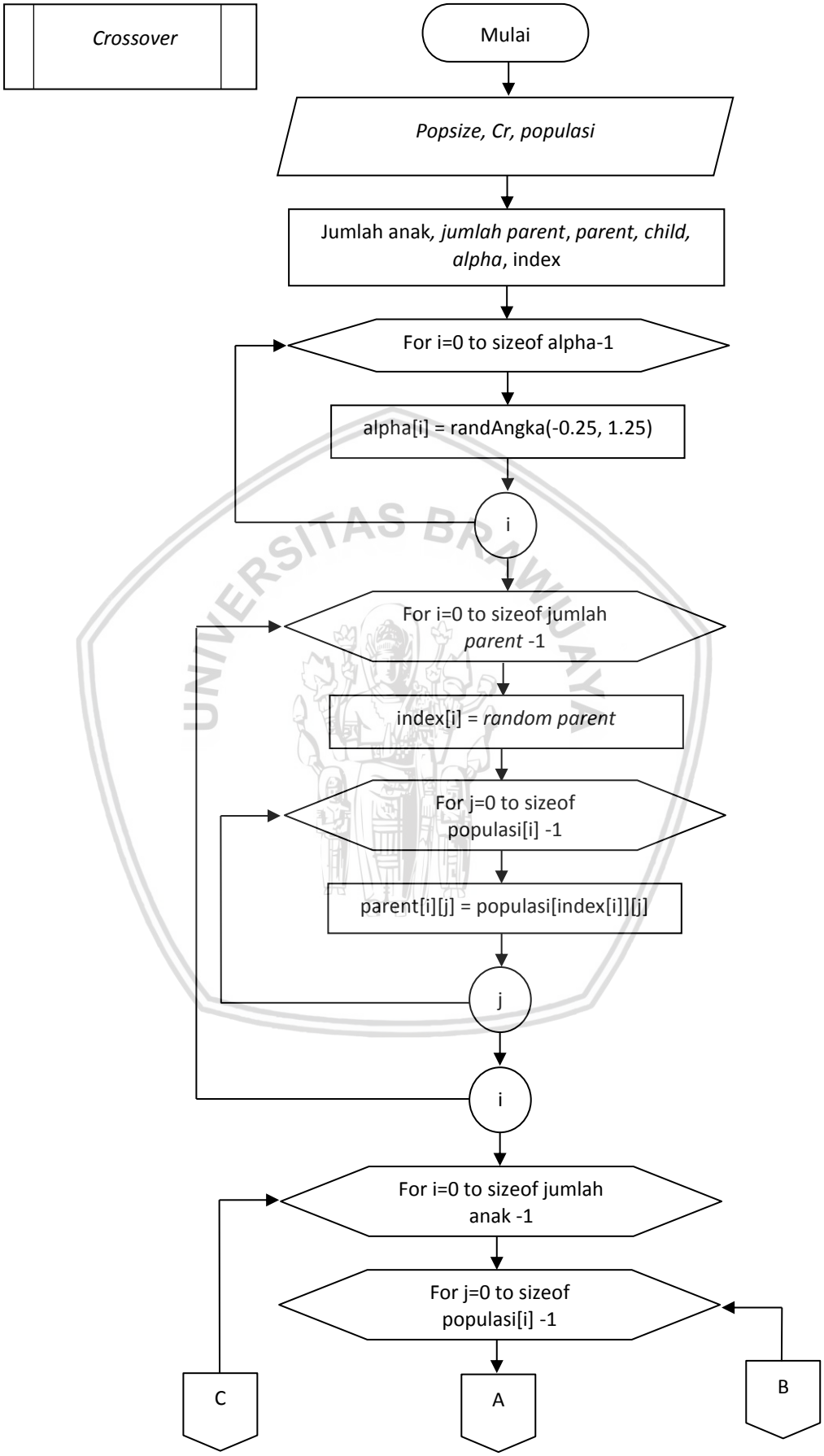
4. Melakukan proses mutasi dengan metode *random mutation* pada parent secara *random*.
5. Menghasilkan *offspring* yang didapatkan dari hasil *crossover* dan mutasi untuk menjadi populasi awal.
6. Menghitung nilai *fitness* setiap individu dengan cara dievaluasi.
7. Melakukan proses seleksi menggunakan metode *elitism* pada setiap individu untuk mendapatkan kromosom terbaik kemudian diambil sebanyak *popsize*.

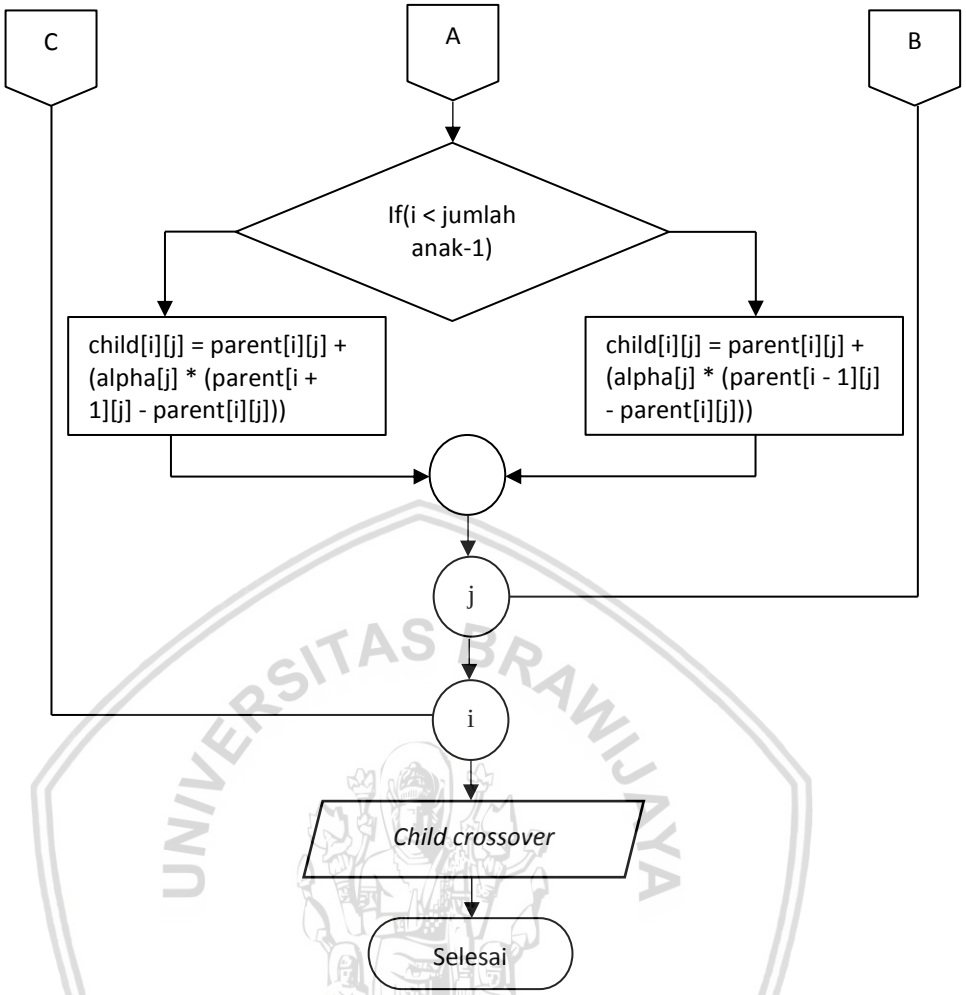


Gambar 4.15 Diagram alir inisialisasi kromosom

Pada Gambar 4.15 dijelaskan langkah-langkah proses inisialisasi *parent* sebagai berikut:

1. Memberikan jumlah *popsize* pada pembangkitan individu dan jumlah gen.
2. Melakukan *random* angka dengan interval [-1;1].
3. Hasil *random* dimasukkan kedalam gen sebanyak jumlah gen yang dimasukkan.

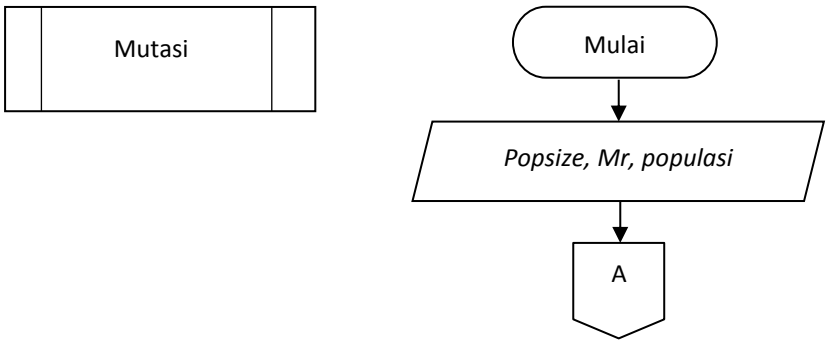

Crossover
Mulai
Popsize, Cr, populasi
Jumlah anak, jumlah parent, parent, child, alpha, index
For i=0 to sizeof alpha-1
alpha[i] = randAngka(-0.25, 1.25)
i
For i=0 to sizeof jumlah parent -1
index[i] = random parent
For j=0 to sizeof populasi[i] -1
parent[i][j] = populasi[index[i]][j]
j
i
For i=0 to sizeof jumlah anak -1
For j=0 to sizeof populasi[i] -1
C
A
B

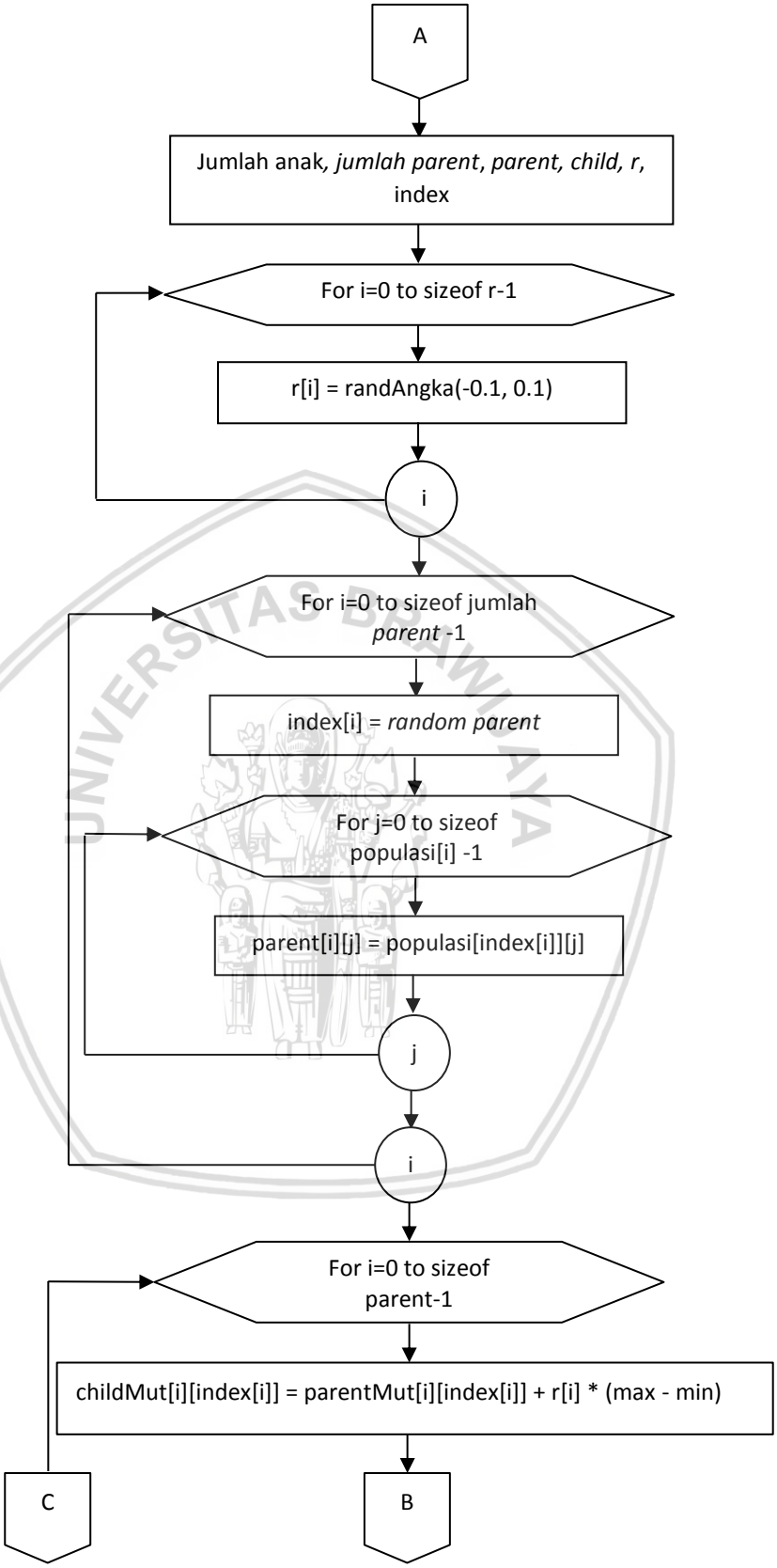


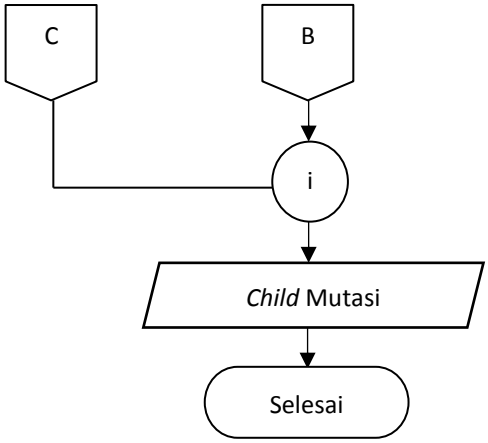
Gambar 4.16 Diagram alir *crossover*

Tahapan pada proses *crossover* ditunjukkan oleh Gambar 4.16 sebagai berikut:

1. Memberikan masukan berupa *cr, popsize,* dan populasi.
2. Perulangan untuk memberikan nilai alpha secara *random* dengan *range* tertentu.
3. Melakukan perulangan untuk memilih individu sebagai parent secara *random*.
4. Melakukan perhitungan pada anak *crossover*.



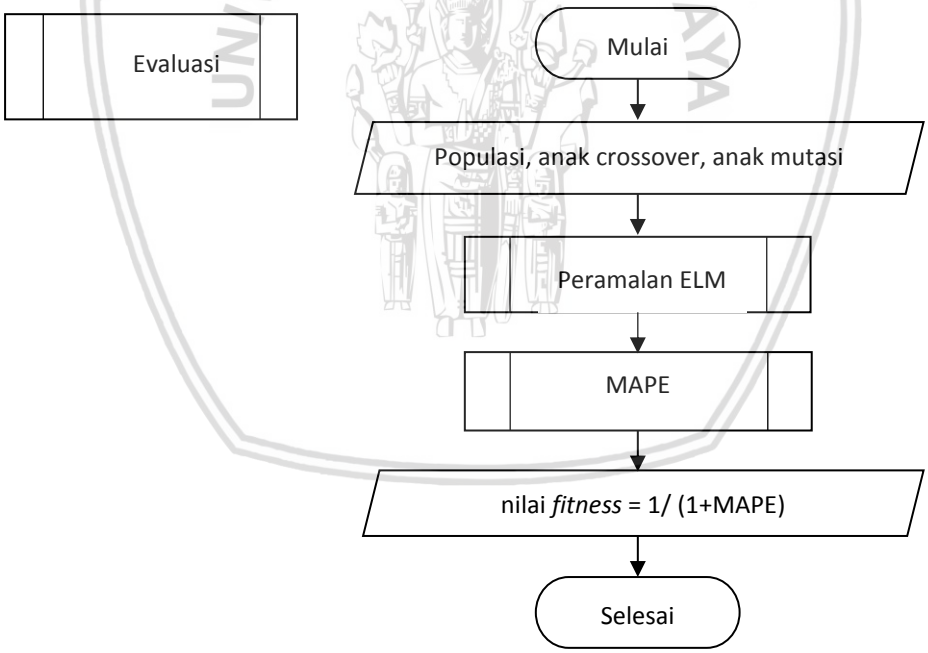

A
Jumlah anak, jumlah parent, parent, child, r, index
For i=0 to sizeof r-1
r[i] = randAngka(-0.1, 0.1)
i
For i=0 to sizeof jumlah parent -1
index[i] = random parent
For j=0 to sizeof populasi[i] -1
parent[i][j] = populasi[index[i]][j]
j
i
For i=0 to sizeof parent-1
childMut[i][index[i]] = parentMut[i][index[i]] + r[i] * (max - min)
C
B



Gambar 4.17 Diagram alir mutasi

Tahapan pada proses mutasi ditunjukkan oleh Gambar 4.17 sebagai berikut:

1. Memberikan masukan berupa *cr, popsize,* dan populasi.
2. Perulangan untuk memberikan nilai r secara *random* dengan *range* tertentu.
3. Melakukan perulangan untuk memilih individu sebagai parent secara *random*.
4. Menghitung anak mutasi.

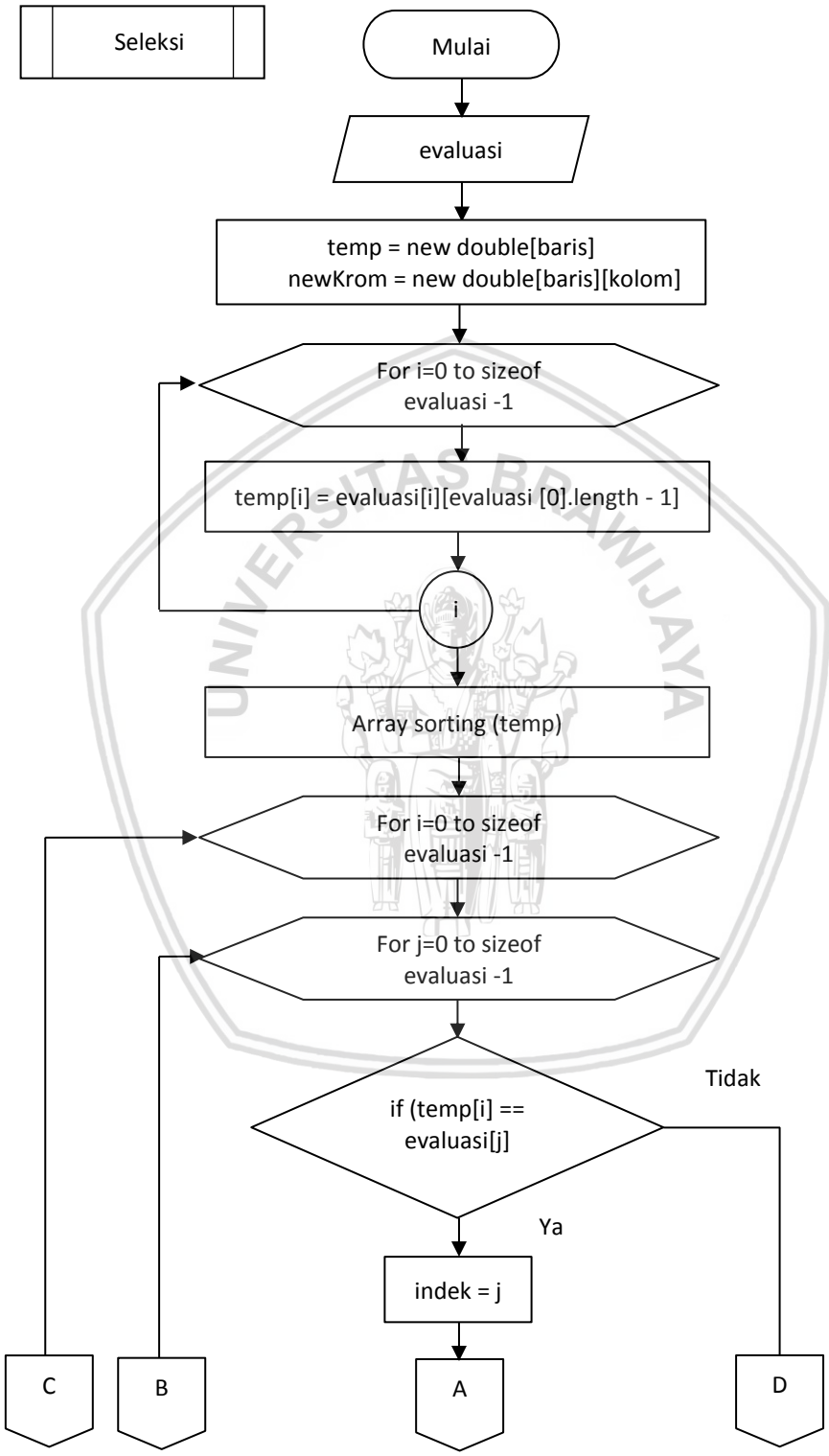


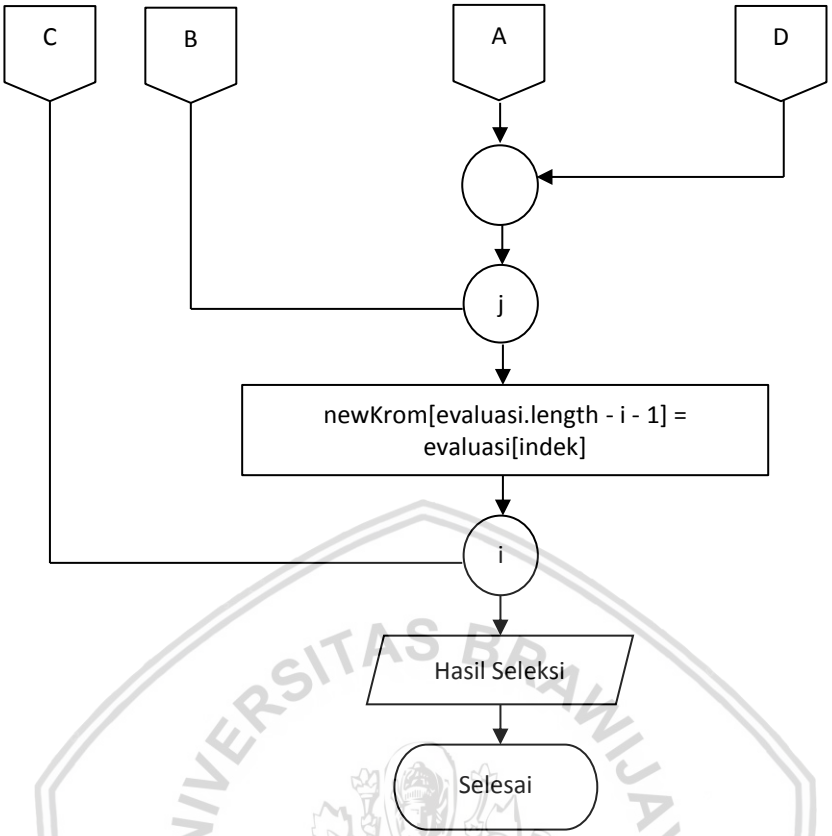
Gambar 4.18 Diagram alir evaluasi

Tahapan pada proses evaluasi ditunjukkan oleh Gambar 4.18 sebagai berikut:

1. Memberikan masukan populasi, anak *crossover*, dan anak mutasi.
2. Melakukan proses peralaman menggunakan metode ELM dengan cara yang sudah dijelaskan pada sub bab sebelumnya mulai dari tahap *training* sampai *testing*.

3. Melakukan perhitungan akurasi menggunakan MAPE untuk mengetahui tingkat kesalahan pada sistem.
4. Menghitung nilai *fitness* sama dengan nilai yang didapatkan dari proses MAPE.





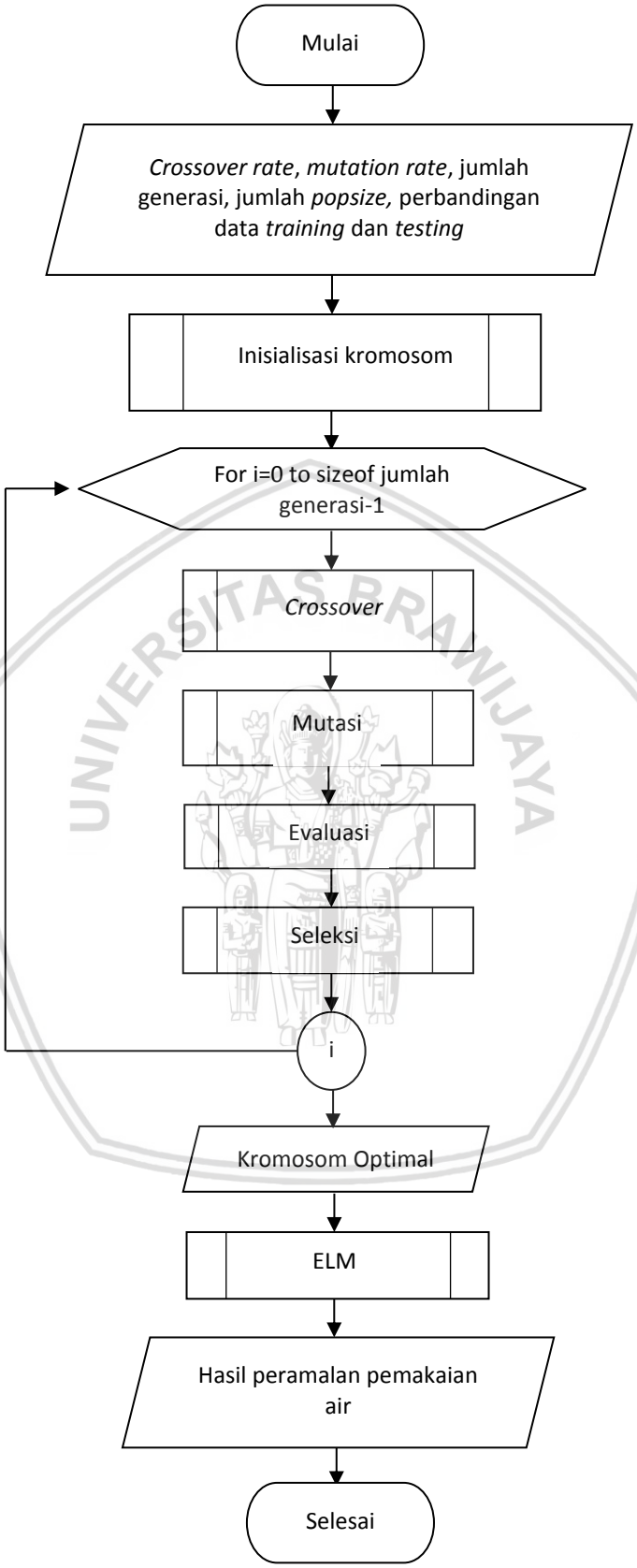
Gambar 4.19 Diagram alir seleksi *elitism*

Pada Gambar 4.19 dijelaskan langkah-langkah proses seleksi pada algoritme genetika sebagai berikut:

1. Imputan berupa hasil dari evaluasi.
2. Melakukan perulangan untuk meng-*copy* evaluasi ke dalam temp.
3. Melakukan sorting pada temp secara *ascending*.
4. Perulangan untuk sorting pada temp secara *ascending*.
5. Membalik *sorting* menjadi *descending*.

1.3.3 Metode *Extreme Learning Machine* (ELM) Dengan Optimasi Algoritme Genetika

Pada penelitian ini menggunakan perpaduan antara dua metode yaitu metode *extreme learning machine* (ELM) dengan optimasi algoritme genetika. Langkah-langkah dalam penerapan metode ini dilakukan seperti yang sudah dijelaskan pada Bab 2. Berikut ini alur dari penerapan metode ELM dengan algoritme genetika dijelaskan pada Gambar 4.20.



Gambar 4.20 Diagram alir proses ELM dengan algoritme genetika

1.4 Perhitungan Manual

Perhitungan manual adalah acuan utama dalam pengimplementasian metode ELM dengan optimasi algoritme genetika untuk mengetahui kebenaran dari perhitugan sistem. Langkah-langkah yang dilakukan dalam penyelesaian ini sesuai dengan diagram alir yang telah dijelaskan pada sub bab sebelumnya.

1.4.1 Normalisasi Data

Pada perhitungan ini langkah pertama yang dilakukan adalah menjadikan data asli untuk dinormalisasi untuk membatasi *range* pada data satu dengan data yang lainnya. *Range* yang digunakan adalah mulai 0 sampai dengan 1. Tujuan utama dilakukan normalisasi selain mengatur *range* data adalah menghindari terjadinya hasil perhitungan tidak terdefinisi pada tahap perhitungan *invers*. Langkah-langkahnya sudah dijelaskan pada Gambar 4.2. Berikut ini adalah contoh perhitungan manual pada normalisasi dengan menggunakan *Min-Max Normalization* yang mengacu pada Persamaan 2.1.

- Normalisasi dengan fitur yang memiliki nilai maksimal = 1905062 dan minimal = 514373.7

$$X = \frac{(x_p - \min x_p)}{(\max x_p - \min x_p)} = \frac{955867.6 - 514373.7}{1905062 - 514373.7} = \frac{441493.946}{1390688.282} = 0.256$$

Berikut ini adalah hasil Normalisasi ditunjukkan oleh Tabel 4.2 dan Tabel 4.3.

Tabel 4.2 Normalisasi data *training*

No	X1	X2	X3	Y
1	0.256	0.187	1	0.717
2	0.187	1	0.222	0
3	1	0.222	0	0.516
4	0.222	0	0.160	0.504
5	0	0.160	0.156	0.626
6	0.160	0.156	0.194	0.559

Tabel 4.3 Normalisasi data *testing*

No	X1	X2	X3	Y
1	0.156	0.194	0.173	0.539
2	0.194	0.173	0.167	1

1.4.2 Inisialisasi Kromosom Pada Populasi Awal

Inisialisasi parent pada algoritme genetika diberikan dengan range [-1;1] untuk memberikan jarak kecil setiap antar data. Pemberian inisialisasi ini dilakukan secara *random*. Pada langkah ini memberikan inisialisasi pada *input weight* pada ELM. Sedangkan untuk dimensinya dilakukan dengan jumlah yang *random* juga. Untuk panjang gen-nya dilakukan sebanyak 9 karena jumlah *input weight* ditentukan sebanya matrik 3 x 3. Untuk pengambilan gen pada setiap individu diambil setiap 3 angka dalam kromosomnya. Berikut ini Tabel 4.4 merupakan *inisialisasi parent* untuk algoritme genetika.

Tabel 4.3 Inisialisasi kromosom pada populasi awal

	X1	X2	X3	X4	X5	X6	X7	X8	X9
P1	0.521	0.533	0.630	0.200	0.175	0.872	0.777	0.084	0.640
P2	0.833	0.098	0.171	0.134	0.235	0.458	0.259	0.304	0.193
P3	0.477	0.556	0.556	0.203	0.169	0.597	0.944	0.156	0.279

1.4.3 Melakukan Reproduksi

Pada langkah ini yaitu melakukan reproduksi dengan melalui dua tahapan yaitu *crossover* dengan mutasi. Hal ini dilakukan untuk menjaga keragaman pada setiap individu. Pada *crossover* menggunakan metode *extended intermediate crossover* karena bilanganya berupa *real code*. Berikut ini adalah contoh perhitungan dari *crossover* dengan rumus yang sudah dijelaskan pada Persamaan 2.8 dan Persamaan 2.9 pada Bab sebelumnya.

- *Crossover*
  - Cr (*Crossover Rate*) = 0.7, sehingga *offspring*-nya adalah cr x *popsize* = 2.1 atau 2 serta $\alpha = 0.4$
  - Misal: P1 dan P3 untuk *parent* yang terpilih. Maka:

    $C_1 = P_1 + \alpha\,(P_2 - P_1)$

    = 0.521 + 0.4 (0.477 – 0.521)

    = 0.503

    $C_2 = P_2 + \alpha\,(P_1 - P_2)$

    = 0.494 + 0.4 (0.521 – 0.494)

    = 0.494

    Sehingga hasil dari *crossover* dapat ditunjukkan oleh Tabel 4.4.

Tabel 4.3 *Crossover*

C1	**X1**	**X2**	**X3**	**X4**	**X5**	**X6**	**X7**	**X8**	**X9**
	0.503	0.542	0.600	0.201	0.172	0.762	0.843	0.112	0.495
C2	**X1**	**X2**	**X3**	**X4**	**X5**	**X6**	**X7**	**X8**	**X9**
	0.494	0.546	0.585	0.201	0.171	0.707	0.127	0.127	0.423

- Mutasi
  - Mr (*Mutation Rate*) = 0.3, sehingga *offspring*-nya adalah cr x *popsize* = 0.9 atau 1 serta r = 0.03.
  - Misal: P2 untuk *parent* yang terpilih. Sehingga dapat dihitung sebagai berikut:

    P2 dengan gen ke-2, C3= x’ + r* (max - min)

    = 0.098 + 0.03*(0.556 - 0.098) = 0.111

    Sehingga menghasilkan *offspring* pada Tabel 4.4 sebagai berikut:

Tabel 4.4 *Offspring*

	X1	X2	X3	X4	X5	X6	X7	X8	X9
P1	0.521	0.533	0.630	0.200	0.175	0.872	0.777	0.084	0.640
P2	0.833	0.118	0.171	0.134	0.235	0.458	0.304	0.304	0.193
P3	0.477	0.551	0.556	0.203	0.169	0.597	0.156	0.156	0.279
C1	0.503	0.542	0.600	0.201	0.172	0.762	0.843	0.112	0.495
C2	0.494	0.546	0.585	0.201	0.171	0.707	0.127	0.127	0.423
C3	0.833	0.111	0.171	0.134	0.235	0.458	0.304	0.304	0.193

1.4.4 Inisialisasi Nilai Bias

Pemberian inisialisasi nilai bias dilakukan secara *random* dengan *range* [0;1] menghasilkan matrik 1x*j*. Berikut ini inisialisasi nilai bias yang digunakan ditunjukkan pada Tabel 4.5.

Tabel 4.5 Inisialisasi nilai bias

b	1	2	3
1	0.118	0.338	0.149

1.4.5 Proses *Training*

Tahap awal yang dilakukan pada proses ELM adalah proses *training*, yaitu dengan menerapkan langkah-langkah yang dijelaskan pada Sub Bab 2.3.2.

1. Menentukan fungsi aktivasi dan jumlah *hidden layer*

   a. Menghitung *Hinit*

   $$Hinit = (X \,.\, W^T) + b$$

   = (0.256 * 0.521 +0.187* 0.533 + 1* 0.630) + 0.118

   = 0.981

   Berikut ini hasil dari proses Hinit yang ditunjukkan pada Tabel 4.6.

Tabel 4.6 *Hinit*

Hinit	1	2	3
1	0.981	1.294	1.004
2	0.888	0.744	0.521
3	0.757	0.576	0.944
4	0.334	0.522	0.424
5	0.301	0.502	0.262
6	0.407	0.566	0.410

   b. Menghitung *hidden layer* dengan fungsi aktivasi

   $$H = 1/(1 + EXP((-X_{train} * W^T) + b))$$

   = 1 / (1 + EXP (-Hinit)

= 1 / (1 + (2.718 ^ (-0.981)))

= 0.727

Hasil *hidden layer* dengan fungsi aktivasi yang ditunjukkan pada Tabel 4.7.

Tabel 4.7 *Hidden layer* dengan fungsi aktivasi

H	1	2	3
1	0.727	0.784	0.731
2	0.708	0.677	0.627
3	0.680	0.640	0.720
4	0.580	0.627	0.604
5	0.574	0.623	0.565
6	0.600	0.638	0.601

2. Menghitung matrik *invers*

   a. Melakukan *Transpose* Matrik (H^T)
   Hasil dari *transpose* matrik ditunjukkan padaTabel 4.8 sebagai berikut:

Tabel 4.8 Hasil *transpose* matrik

H^T	1	2	3	4	5	6
1	0.727	0.708	0.680	0.582	0.574	0.600
2	0.784	0.677	0.640	0.627	0.623	0.638
3	0.731	0.627	0.720	0.604	0.565	0.601

   b. Menghitung perkalian *Transpose* matrik (H^T) dengan matrik *H*.
   Berikut ini hasil perkalian ditunjukkan pada Tabel 4.9 dibawah ini.

Tabel 4.9 Hasil H^T kali *H*

H^T.H	1	2	3
1	2.525	2.594	2.505
2	2.594	2.674	2.575
3	2.505	2.576	2.492

   c. Menghitung *invers* dari hasil perkalian H^T dengan *H* menggunakan Persamaan Operasi Baris Elementer (OBE). Berikut ini proses perhitungan *invers* matrik menggunakan OBE:
   Langkah 1: Membentuk matrik identitas (*I*)

$$[H \mid I] = \begin{bmatrix} 2.525 & 2.594 & 2.505 \\ 2.594 & 2.674 & 2.575 \\ 2.505 & 2.576 & 2.492 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

   Langkah 2: R1 ÷ 2.525 → R1

$$[H \mid I] = \begin{bmatrix} 1 & 1.027 & 0.991 \\ 2.594 & 2.674 & 2.575 \\ 2.505 & 2.576 & 2.492 \end{bmatrix} \begin{bmatrix} 0.395 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Langkah 3: R2 – (2.594 * R1)→ R2

$$[H \mid I] = \begin{bmatrix} 1 & 1.027 & 0.991 \\ 0 & 0.009 & 0.002 \\ 2.505 & 2.576 & 2.492 \end{bmatrix} \begin{bmatrix} 0.395 & 0 & 0 \\ -1.027 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Langkah 4: R3 – (2.505*R1) → R3

$$[H \mid I] = \begin{bmatrix} 1 & 1.027 & 0.991 \\ 0 & 0.009 & 0.002 \\ 0 & 0.002 & 0.007 \end{bmatrix} \begin{bmatrix} 0.395 & 0 & 0 \\ -1.027 & 1 & 0 \\ -0.992 & 0 & 1 \end{bmatrix}$$

Langkah 5: R2 ÷ 0.009 → R2

$$[H \mid I] = \begin{bmatrix} 1 & 1.027 & 0.991 \\ 0 & 1 & 0.244 \\ 0 & 0.002 & 0.007 \end{bmatrix} \begin{bmatrix} 0.395 & 0 & 0 \\ -105.564 & 102.766 & 0 \\ -0.992 & 0 & 1 \end{bmatrix}$$

Langkah 6: R1 – (1.027 * R2) → R1

$$[H \mid I] = \begin{bmatrix} 1 & 0 & 0.704 \\ 0 & 1 & 0.244 \\ 0 & 0.002 & 0.007 \end{bmatrix} \begin{bmatrix} 108.835 & -105.564 & 0 \\ -105.564 & 102.766 & 0 \\ -0.992 & 0 & 1 \end{bmatrix}$$

Langkah 7: R3 – (0.002 * R2) → R3

$$[H \mid I] = \begin{bmatrix} 1 & 0 & 0.704 \\ 0 & 1 & 0.244 \\ 0 & 0 & 0.006 \end{bmatrix} \begin{bmatrix} 108.835 & -105.564 & 0 \\ -105.564 & 102.766 & 0 \\ -0.755 & -0.230 & 1 \end{bmatrix}$$

Langkah 8: R3 ÷ (0.006)→ R3

$$[H \mid I] = \begin{bmatrix} 1 & 0 & 0.704 \\ 0 & 1 & 0.244 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 108.835 & -105.564 & 0 \\ -105.564 & 102.766 & 0 \\ -111.384 & -33.976 & 147.457 \end{bmatrix}$$

Langkah 9: R1 – (0.704 * R3) → R1

$$[H \mid I] = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0.244 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 191.289 & -80.413 & -109.157 \\ -105.564 & 102.766 & 0 \\ -111.384 & -33.976 & 147.457 \end{bmatrix}$$

Langkah 10: R2 – (0.244 * R3) → R2

$$[H \mid I] = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 191.289 & -80.413 & -109.157 \\ -74.276 & 111.09 & -36.125 \\ -111.384 & -33.976 & 147.457 \end{bmatrix}$$

Hasil *invers* pada Tabel 4.10 sebagai berikut:

Tabel 4.10 Hasil nilai *invers*

$(H^T.H)^{-1}$	1	2	3
1	191.289	-80.413	-109.157
2	-78.276	111.089	-36.125
3	-111.384	-33.976	147.457

d. Menghitung matrik moore-penrose pseudo pada invers (H^+)

$$H^+ = (H^T.H)^{-1}.H^T$$

Hasil H^+ pada Tabel 4.11 sebagai berikut:

Tabel 4.11 Hasil H^+

H^+	1	2	3	4	5	6
1	-3.850	12.553	0.143	-4.952	-1.828	-2.089
2	3.807	-2.819	-8.167	2.257	3.789	2.156
3	0.229	-9.452	8.585	2.887	-1.851	0.113

3. Menghitung *Output Weight* ($\hat{\beta}$) Pada ELM

Pada perhitungan *output weight* dilakukan dengan rumus yang telah di jelaskan pada Persamaan 2.4. Hasil dari *output weight* nantinya akan dijadikan nilai masukan pada proses *testing*. Pada Tabel 4.12 berikut ini hasil dari *output weight*.

Tabel 4.12 Hasil *output weight*

$\hat{\beta}$	1
1	-7.498
2	3.230
3	4.957

1.4.6 Proses *Testing*

Pada proses *testing* ELM ini dilakukan dengan tahapan yang sudah dijelaskan oleh Sub Bab sebelumnya yaitu pada Bab 2. Berikut ini tahapan manualisasi untuk proses *testing* pada ELM.

1. Menentukan fungsi aktivasi dan jumlah *hidden layer*
   a. Menghitung *Hinit*

$$Hinit = (X \,.\, W^T) + b$$

= (0.156 * 0.521 + 0.194* 0.533+ 0.173* 0.630) + 0.118

= 0.412

Tabel 4.13 berikut ini hasil dari perhitungan Hinit sebagai berikut:

Tabel 4.13 *Hinit* pada *testing*

Hinit	1	2	3
1	0.412	0.554	0.397
2	0.417	0.553	0.421

   b. Menghitung *hidden layer* dengan fungsi aktivasi

$$H = 1/(1 + EXP((-X_{test} * W^T) + b))$$

= 1 / (1 + EXP (-Hinit)

= (1 / (1+ 2.718^ (-0.412))) = 0.601

Tabel 4.14 merupakan Hasil *hidden layer* dengan fungsi aktivasi sebagai berikut:

Tabel 4.14 *Hidden layer* dengan fungsi aktivasi pada *testing*

H	1	2	3
1	0.601	0.635	0.598
2	0.602	0.634	0.603

2. Melakukan Perhitungan Hasil Peramalan

Untuk menghitung hasil peramalan menggunakan Persamaan 2.5. Berikut ini adalah perhitungan untuk melakukan peramalan.

$\hat{Y} = H * \hat{\beta}$

= (0.616* 3.503 + 0.628 * 1.240 + 0.662 * (-4.045))

= 0.259

Tabel 4.15 berikut ini hasil peramalan dengan perhitungan yang sudah dilakukan sebagai berikut:

Tabel 4.15 Hasil perhitungan peramalan

$\hat{Y}$	1
1	0.506
2	0.524

1.4.7 Denormalisasi Data

Tahap ini bertujuan untuk mengembalikan nilai ke angka sebenarnya tanpa dibatasi nilai minimal ataupun maksimalnya. Untuk perhitungan denormalisasi telah di jelaskan pada Persamaan 2.6. Berikut ini contoh perhitungan denormalisasi.

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

= (0.506 × (1024312 – 628216.78)) + 628216.78

= 828723.703

Tabel 4.16 merupakan Hasil dari perhitungan denormalisasi data sebagai berikut:

Tabel 4.16 Hasil denormalisasi

x	1
1	828723.703
2	836017.191

1.4.8 Menghitung MAPE

Pada perhitungan MAPE bertujuan untuk mengetahui seberapa akurat sistem dengan metode ELM dan algoritme genetika serta mengetahui tingkat kesalahan pada sistem. Berikut ini hasil perhitungan untuk mencari nilai MAPE dengan Persamaan 2.7 yang sudah dijelaskan pada sub bab sebelumnya.

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{Y_i - \hat{Y}}{Y_i}\right| \times 100$$

= (19.961 / 2)

= 9.980

1.4.9 Melakukan Evaluasi

Untuk melakukan evaluasi dengan cara menghitung nilai *fitness* setiap individu pada populasi. Untuk menghitung nilai *fitness* telah dijelaskan pada Persamaan 2.8 yang dibahas pada bab sebelumnya. Tabel 4.17 berikut ini hasil perhitungan evaluasi setiap individu.

Tabel 4.17 Hasil perhitungan evaluasi

	X1	X2	X3	X4	X5	X6	X7	X8	X9	*fitness*
P1	0.521	0.533	0.630	0.200	0.175	0.872	0.777	0.084	0.640	0.091
P2	0.833	0.118	0.171	0.134	0.235	0.458	0.304	0.304	0.193	0.057
P3	0.477	0.551	0.556	0.203	0.169	0.597	0.156	0.156	0.279	0.095
C1	0.503	0.542	0.600	0.201	0.172	0.762	0.843	0.112	0.495	0.084
C2	0.494	0.546	0.585	0.201	0.171	0.707	0.127	0.127	0.423	0.088
C3	0.833	0.111	0.171	0.134	0.235	0.458	0.304	0.304	0.193	0.057

1.4.10 Melakukan Seleksi

Untuk melakukan seleksi dengan cara *elitism* dengan mengurutkan nilai *fitness* dari angka terbesar sampai angka terkecil, kemudian diambil dari *parent* dan *child* sebanyak *popsize*. Tabel 4.18 berikut ini hasil seleksi berdasarkan nilai *fitness*.

Tabel 4.18 Hasil seleksi

	X1	X2	X3	X4	X5	X6	X7	X8	X9	*fitness*
P1	0.477	0.551	0.556	0.203	0.169	0.597	0.156	0.156	0.279	0.095
P2	0.521	0.533	0.630	0.200	0.175	0.872	0.777	0.084	0.640	0.091
P3	0.494	0.546	0.585	0.201	0.171	0.707	0.127	0.127	0.423	0.088

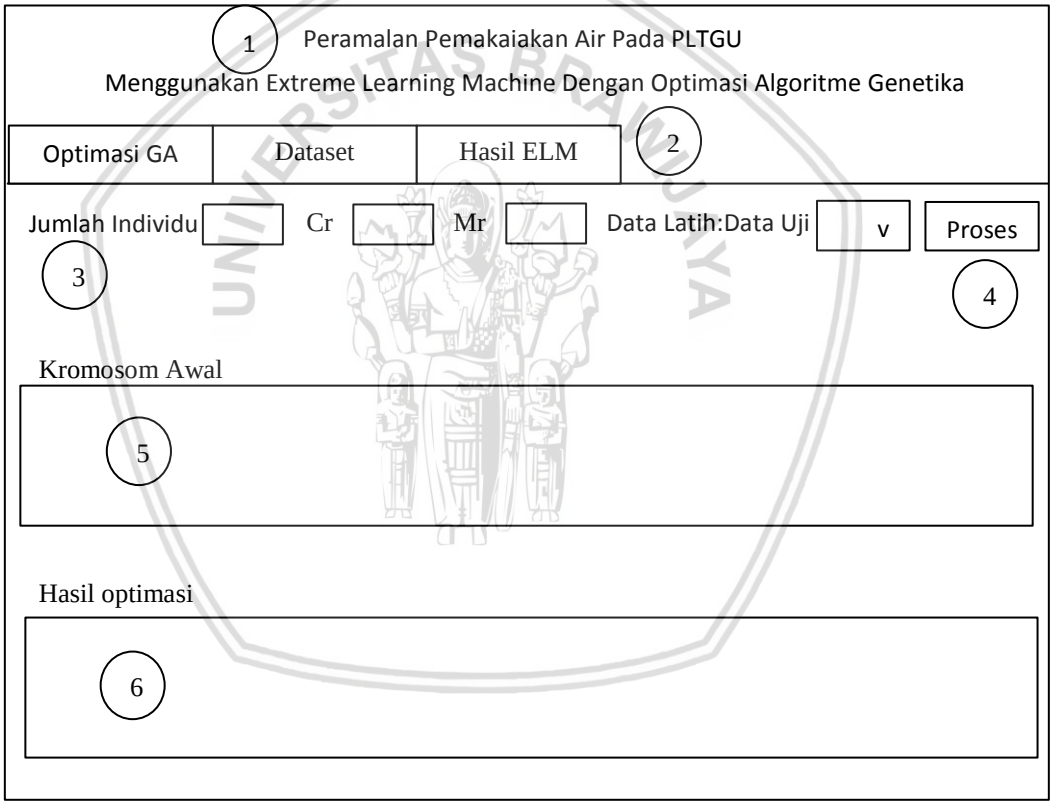
Sehingga dapat ditarik kesimpulkan bahwa individu yang terpilih adalah P1 karena memiliki *fitness* optimal.

1.5 Perancangan Antarmuka

Pada perancangan antarmuka ini digunakan untuk memberikan gambaran dalam mengimplementasi sistem peramalan dengan menggunakan metode ELM dengan optimasi algoritme genetika. Pada antarmuka sistem ini terdapat dua halaman diantaranya antarmuka untuk proses ELM dan algoritme genetika serta antarmuka untuk peramalan air.

1.5.1 Perancangan Antarmuka Proses ELM Dengan Algorime Genetika

Pada perancangan antarmuka ELM dengan algoritme genetika digunakan untuk proses perhitungan mengoptimasi *input weight* pada ELM dengan menggunakan algoritme genetika. Berikut ini antarmuka yang ditunjukkan oleh Gambar 4.21 untuk perancangan proses ELM dan algoritme genetika.



Gambar 4.21 Perancangan antarmuka hasil ELM dengan optimasi Algoritme Genetika

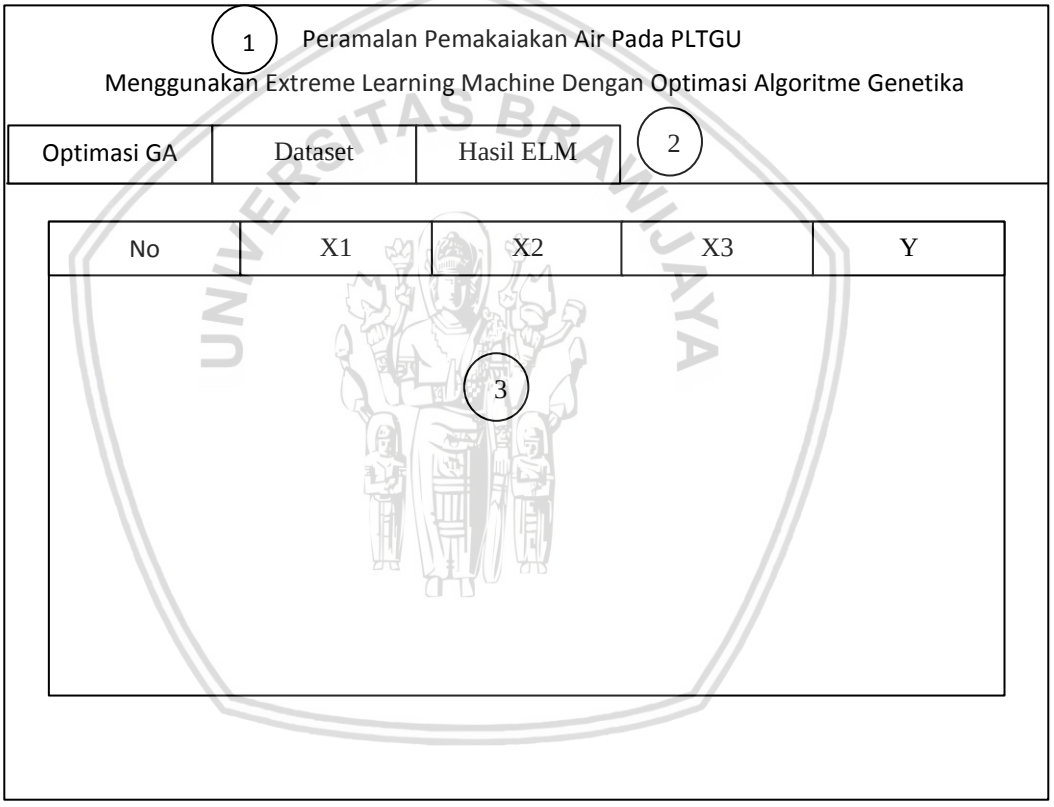
Berikut ini penjelasan dari perancangan antarmuka ELM dengan algoritme genetika yang ditunjukkan pada Gambar 4.21

1. *Header* untuk sistem peramalan.
2. *Tab* untuk halaman Optimasi GA, Dataset, dan Hasil ELM.
3. *Text field* untuk parameter pada algoritme genetika dan Extreme Learning Machine (ELM).

4. *Buttom* proses untuk melakukan tombol pemrosesan pada metode untuk memulai mengoptimasi *input weight* pada ELM dengan algoritme genetika.
5. *Table* untuk menampilkan individu awal.
6. *Table* untuk menampilkan hasil dari optimasi *input weight* pada ELM dengan algoritme genetika.

1.5.2 Perancangan Antarmuka Dataset

Perancangan antarmuka dataset digunakan untuk menampilkan data hasil produksi air. terdapat 3 fitur diantaranya fitur x1 untuk tanggal 2 januari 2017, x2 menandakan tanggal 4 januari, x3 menandakan tanggal 6 januari, dan seterusnya untuk baris n + 1. Sedangkan untuk Y menandakan hasil produksi air. Berikut ini perancangan antarmuka dataset ditunjukkan pada Gambar 4.22.



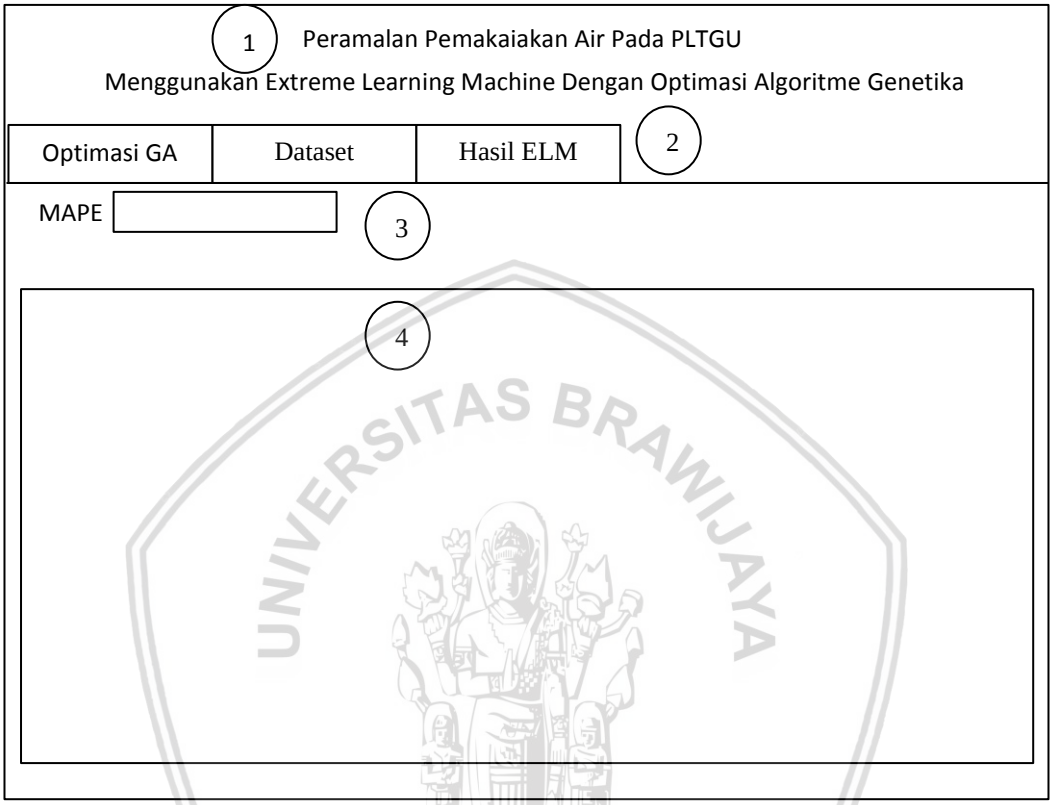
Gambar 4.22 Perancangan antarmuka dataset

Berikut ini penjelasan dari perancangan antarmuka dataset yang telah ditunjukkan oleh Gambar 4.22.

1. *Header* untuk sistem peramalan.
2. *Tab* untuk halaman Optimasi GA, Dataset, dan Hasil ELM.
3. *Table* untuk menampilkan dataset yang akan digunakan untuk proses data *training* dan data *testing*.

1.5.3 Perancangan Antarmuka Dari Hasil ELM

Perancangan antarmuka ini digunakan untuk menampilkan hasil peramalan air setiap seminggu sekali dengan optimasi yang telah dilakukan pada algoritme genetika. Perancangan antarmuka hasil peramalan ditunjukkan pada Gambar 4.23 berikut ini.



Gambar 4.23 Perancangan antarmuka dari hasil peramalan

Berikut ini penjelasan dari perancangan antarmuka ELM dengan algoritme genetika yang ditunjukkan pada Gambar 4.23.

1. *Header* untuk sistem peramalan.
2. *Tab* untuk halaman Optimasi GA, Dataset, dan Hasil ELM.
3. *Text field* untuk menampilkan hasil MAPE.
4. *Table* untuk menampilkan data aktual dan data hasil peramalan.

1.6 Perancangan Skenario Pengujian

Pengujian pada penelitian ini merupakan pengujian yang terkait dengan ELM dengan algoritme genetika. Pada pengujian ini digunakan untuk mengetahui seberapa tinggi nilai akurasi serta kinerja dari metode yang telah dipilih oleh peneliti. Berikut ini beberapa skenario pengujian pada metode ELM dan algoritme genetika.

a. Pengujian Nilai *Crossover Rate* dan Nilai *Mutation Rate*

b. Pengujian Jumlah *Popsize*

c. Pengujian Jumlah Generasi

d. Pengujian Jumlah Data *Training*

e. Pengujian Jumlah Fitur

1.6.1 Pengujian Kombinasi Nilai *Crossover Rate* (Cr) Dan *Mutation Rate* (Mr)

Pengujian nilai *crossover rate* digunakan untuk mengetahui nilai yang optimal dalam memberikan proses reproduksi sehingga mendapatkan anak yang terbaik. Pada pengujian ini dilakukan sebanyak 10 kali untuk mengetahui yang terbaik batas nilai *crossover rate*. Berikut ini adalah Tabel 4.19 untuk pengujian nilai *crossover rate*.

Tabel 4.19 Pengujian kombinasi nilai *Cr* dan *Mr*

Cr:Mr	Nilai *Fitness*							Rata-rata *fitness*
	Percobaan ke-							
	1	2	3	4	5	…	10	
0.1:0.9								
0.2:0.8								
0.3:0.7								
0.4:0.6								
0.5:0.5								
0.6:0.4								
0.7:0.3								
0.8:0.2								
0.9:0.1								

1.6.2 Pengujian Jumlah *Popsize*

Pengujian pada jumlah *popsize* bertujuan untuk mengetahui pengaruh jumlah *popsize* terhadap keberagaman solusi nilai *fitness* yang dihasilkan. Pengujian ini dilakukan dengan kelipatan 10 sampai dengan 200 dilakukan 10 kali percobaan. Tabel 4.20 berikut ini perancangan pengujian pada jumlah *popsize.*

Tabel 4.20 Pengujian jumlah *popsize*

Popsize	Nilai *Fitness*							Rata-rata *fitness*
	Percobaan ke-							
	1	2	3	4	5	…	10	
10								
20								

Tabel 4.20 Pengujian jumlah popsize (lanjutan)

Popsize	Nilai *Fitness*							Rata-rata *fitness*
	Percobaan ke-							
	1	2	3	4	5	…	10	
30								
40								
50								
60								
70								
…								
200								

1.6.3 Pengujian Jumlah Generasi

Pengujian pada jumlah generasi algoritme genetika bertujuan untuk mengetahui jumlah generasi yang optimal dalam memberikan hasil optimasi. Pengujian ini dilakukan dengan memberikan kelipatan 100, dimulai dari 100 sampai dengan 1200. Berikut ini Tabel 4.21 merupakan rancangan pengujian jumlah generasi.

Tabel 4.21 Pengujian jumlah generasi

Generasi ke-	Nilai *Fitness*							Rata-rata *fitness*
	Percobaan ke-							
	1	2	3	4	5	…	10	
100								
200								
300								
400								
500								
600								
700								
800								
900								
1000								
1100								
1200								

1.6.4 Pengujian Jumlah Data *Training*

Pengujian pada jumlah data *training* bertujuan untuk mengetahui pengaruh jumlah data *training* terhadapat metode ELM dalam mengenali pola data. Untuk perbandingan pengujian data *training* dan *testing* menggunakan 5 perbandingan, diantaranya 80%:20%, 70%:20%, 60%:20%, 50%:20%, 40%:20% dan dilakukan percobaan sebanyak 10 kali. Berikut ini Tabel 4.22 merupakan rancangan pengujian pada jumlah data *training*.

Tabel 4.22 Pengujian jumlah data *training*

Training:Testing	**Nilai *Fitness***							**Rata-rata *fitness***
	Percobaan ke-							
	1	**2**	**3**	**4**	**5**	**…**	**10**	
80%:20%								
70%:20%								
60%:20%								
50%:20%								
40%:20%								

1.6.5 Pengujian Jumlah Fitur

Pengujian pada jumlah fitur digunakan untuk mengetahui pengaruh banyaknya fitur terhadap hasil peramalan dan tingkat keakuratan yang dihasilkan. Pengujian ini dilakukan sebanyak 10 kali percobaan dengan menguji jumlah fitur sebanyak 2, 3, 4, dan 5. Berikut ini Tabel 4.23 merupakan rancangan pengujian pada jumlah fitur proses ELM.

Tabel 4.23 Pengujian jumlah fitur

Jumlah Fitur	**Nilai MAPE**							**Rata-rata MAPE**
	Percobaan ke-							
	1	**2**	**3**	**4**	**5**	**…**	**10**	
2								
3								
4								
5								

BAB 5 IMPLEMENTASI

1.1 Implementasi Metode Extreme Learning Machine Dengan Algoritme Genetika

Implementasi ini digunakan untuk mengimplementasi perancangan yang dibahas pada Bab sebelumnya yaitu Bab 4. Pada implementasi peramalan pemakaian air ini menggunakan bahasa java dengan antarmuka berupa GUI.

1.1.1 Implementasi Normalisasi Data

Implementasi proses normalisasi data dilakukan dengan menggunakan *Min-Max Normalization*. Data yang dinormalisasi akan berada dalam *range* [0;1], sehingga data satu dengan data yang lainnya memiliki jarak yang kecil. Berikut ini adalah implementasi proses normalisasi data ditunjukkkan pada Kode Program 5.1.

```
1   public double[][] normalisasi(double[][] data) {
2     double norm[][] = new double[row][col];
3       for (int i = 0; i < row; i++) {
4        for (int j = 0; j < col; j++) {
5          norm[i][j] = (data[i][j] - getmin(getKolom(data,
6                        j))) / (getmax(getKolom(data, j)) -
7                        getmin(getKolom(data, j)));
8              }
9           }
10          return norm;
11      }
```

Kode Program 5.1 Proses normalisasi data

Pada Kode Program 5.1 terdapat kode program normalisasi data yang dapat dijelaskan sebagai berikut:

1. Baris 2 yaitu inisialisasi variabel normalisasi dengan dengan ukuran baris dan kolom yang sudah di inisialisasi.
2. Baris 3 s.d 11 yaitu perulangan untuk proses normalisasi data.

1.1.2 Implementasi Proses *Transpose* Matrik

Implementasi pada proses transpose matrik dilakukan dengan mengubah baris menjadi kolom dan sebaliknya. Berikut ini adalah implementasi dari proses transpose matrik pada ELM ditunjukkan pada Kode Program 5.2.

```
1   public double[][] transposeMatrik(double[][] matrik) {
2     double[][] HasilTranspose = new
3                     double[matrik[0].length][matrik.length];
4       for (int i = 0; i < HasilTranspose.length; i++) {
5         for (int j = 0; j < HasilTranspose[i].length; j++) {
6            HasilTranspose[i][j] = matrik[j][i];
7         }}
8       return HasilTranspose;
9   }
```

Kode Program 5.2 Proses *transpose* matrik

Pada Kode Program 5.2 terdapat kode program *Transpose* Matrik yang dapat dijelaskan sebagai berikut:

1. Baris 2 yaitu inisialisasi variabel hasil *transpose* matrik.
2. Baris 3 s.d 10 yaitu perulangan untuk proses *transpose* matrik dengan merubah kolom menjadi baris dan baris menjadi kolom.

1.1.3 Implementasi Proses *Output Hidden Layer* Dengan Fungsi Aktivasi Dan Perkalian Matrik

Implementasi pada proses *output hidden layer* yang dilakukan dengan fungsi aktivasi ini secara perhitungan sudah dijelaskan pada Bab 4 dengan fungsi persamaannya. Pada proses ini perhitungannya dilakukan sama dengan pada proses *testing.* Sedangkan perkalian matrik merupakan perkalian antara dua buah matrik. Berikut ini merupakan implementasi dari proses *output hidden layer* dengan fungsi aktivasi dan perkalian matrik ditunjukkan pada Kode Program 5.3 dan Gambar 5.4.

```
1  public double[][] Hinit(double[][] data, double[][]
2                                  tranBobot, double[] bias) {
3     double[][] hinit = new
4                     double[data.length][tranBobot[0].length];
5       for (int i = 0; i < data.length; i++) {
6         for (int j = 0; j < tranBobot[0].length; j++) {
7            double total = 0;
8              for (int k = 0; k < tranBobot[0].length; k++) {
9                        total += (data[i][k] *tranBobot[k][j]);
10                    }
11                   hinit[i][j] = total + bias[j];
12             }
13          }
14          return hinit;
15       }
16
17 public double[][] fungsiHidenLayer(double[][] Hinit) {
18    double[][] HidenLayer = new
19                        double[Hinit.length][Hinit[0].length];
20       for (int i = 0; i < Hinit.length; i++) {
21          for (int j = 0; j < Hinit[0].length; j++) {
22             HidenLayer[i][j] = 1 / (1 + Math.exp(-
23                                  Hinit[i][j]));
24             }
25          }
26          return HidenLayer;
27       }
28
```

Kode Program 5.3 Proses *output hidden layer* dengan fungsi aktivasi

Pada Kode Program 5.3 terdapat kode program *output hidden layer* dengan fungsi aktivasi yang dapat dijelaskan sebagai berikut:

1. Baris 2 yaitu inisialisasi variabel hinit dengan ukuran panjang data.

2. Baris 3 s.d 14 yaitu perulangan untuk proses perhitungan hinit dengan mengkalikan *transpose* bobot matrik dengan data kemudian di tambah dengan nilai bias.
3. Baris 18 yaitu inisialisasi variabel fungsi hidden layer dengan ukuran panjang dari matrik hinit.

```
1  public double[][] perkalianMatrik(double[][] matrik1,
2                    double[][] matrik2) {
3    double[][] hasilKali = new
4                   double[matrik1.length][matrik2[0].length];
5      for (int i = 0; i < hasilKali.length; i++) {
6         for (int j = 0; j < hasilKali[0].length; j++) {
7             double total = 0;
8            for (int k = 0; k < matrik1[0].length; k++) {
9             total = total + (matrik1[i][k] * matrik2[k][j]);
10                 }
11                 hasilKali[i][j] = total;
12              }
13          }
14          return hasilKali;
15     }
```

Kode Program 5.4 Proses perkalian matrik

Pada Kode Program 5.4 terdapat kode program perkalian matrik yang dapat dijelaskan sebagai berikut:

1. Baris 3 yaitu inisialisasi variabel hasil kali dengan ukuran panjang matrik 1 dan matrik 2.
2. Baris 5 s.d 14 yaitu perulangan untuk proses perkalian antara dua buah matrik.

1.1.4 Implementasi Perhitungan *Transpose Output Hidden Layer* Dengan Fungsi Aktivasi

Implementasi pada perhitungan *transpose output hidden layer* dilakukan dengan mengubah hasil *output hidden layer* dari kolom menjadi baris dan sebaliknya. Berikut ini merupakan hasil dari *transpose output hidden layer* dengan fungsi aktivasi ditunjukkan oleh Kode Program 5.5.

```
1  public double[][] HT(double[][] fungsiHiddenLayer) {
2    double[][] hasilHT = transposeMatrik(fungsiHiddenLayer);
3       return hasilHT;
4  }
```

Kode Program 5.5 Proses *transpose hidden layer*

Pada Kode Program 5.5 terdapat kode program *transpose hidden layer* yang dapat dijelaskan sebagai berikut:

1. Baris 1 yaitu inisialisasi variabel hasil *transpose hidden layer* (H^T) dengan memanggil metod *transpose* matrik.

1.1.5 Implementasi Perhitungan Nilai *Invers* Matrik

Implementasi pada perhitungan nilai *invert* matrik dilakukan sesuai dengan persamaan yang telah dijelaskan pada Bab 4. Pada perhitungan *moore-penrose*

psudo invers didapatkan dari perhitungan *transpose hidden layer* (H^T), perkalian matrik H^T dengan matrik *hidden layer* (H), dan perkalian matrik H^T dengan hasil invers dari perkalian matrik H^T dengan matrik *hidden layer* (H). Berikut ini merupakan implementasi proses perhitungan niai *invert* matrik ditunjukkkan pada Kode Program 5.6.

```
1   public double[][] HT(double[][] fungsiHiddenLayer) {
2     double[][] hasilHT = transposeMatrik(fungsiHiddenLayer);
3        return hasilHT;
4   }
5
6   public double[][] HTkaliH(double[][] tranHidLayer,
7                     double[][] fungsiHiddenLayer) {
8     double[][] HTkaliH = perkalianMatrik(tranHidLayer,
9                          fungsiHiddenLayer);
10       return HTkaliH;
11  }
12
13   public double[][] Invers(double[][] HtkaliH) {
14    double[][] invers = new
15                    double[HtkaliH.length][HtkaliH[0].length];
16      for (int a = 0; a < HtkaliH.length; a++) {
17         for (int b = 0; b < HtkaliH[0].length; b++) {
18            if (a == b) {
19               invers[a][b] = 1;
20            } else {
21                  invers[a][b] = 0;
22               }
23            }
24       }
25       for (int i = 0; i < HtkaliH.length; i++) {
26         double x = HtkaliH[i][i];
27           for (int j = 0; j < HtkaliH[0].length; j++) {
28                invers[i][j] = invers[i][j] / x;
29               HtkaliH[i][j] = HtkaliH[i][j] / x;
30             }
31            for (int k = 0; k < HtkaliH[0].length; k++) {
32              double y = HtkaliH[k][i];
33               for (int l = 0; l < HtkaliH[0].length; l++) {
34                 if (i != k) {
35                  invers[k][l] = invers[k][l] - y *
36                                       invers[i][l];
37                   HtkaliH[k][l] = HtkaliH[k][l] - y *
38                                       HtkaliH[i][l];
39              }
40           }
41         }
42     }
43      return invers;
44  }
45
46  public double[][] Hplus(double[][] invers, double[][]
47                          tranHidLayer) {
48    double[][] hPlus = perkalianMatrik(invers, tranHidLayer);
49       return hPlus;
50  }
```

Kode Program 5.6 Proses perhitungan nilai *invers* matrik

Pada Kode Program 5.6 terdapat kode program perhitungan nilai *invers* matrik yang dapat dijelaskan sebagai berikut:

1. Baris 2 yaitu inisialisasi variabel hasil *transpose hidden layer* (H^T) dengan memanggil metod *transpose* matrik.
2. Baris 8 yaitu inisialisasi variabel hasil perkalian *transpose hidden layer* (H^T) dengan *hidden layer* (H).
3. Baris 14 s.d 24 yaitu inisialisasi variabel *invers* matrik untuk menampung hasil dari inisialisasi matrik identitas.
4. Baris 25 s.d 44 yaitu proses perhitungan nilai *invers* matrik dengan menggunakan OBE (Operasi Baris Elementer). Dengan cara OBE yaitu akan mengubah matrik yang di-*invers* menjadi matrik identitas, sedangkan matrik identitas sebelumnya akan di ubah menjadi sebuah matrik baru. Sehingga matrik baru tersebut merupakan hasil dari invers matrik.
5. Baris 48 yaitu proses perhitungan *moore-penrose pseudo invers* dengan melakukan perkalian hasil dari *invers* matrik dengan *transpose hidden layer*.

1.1.6 Implementasi Perhitungan Nilai *Output Weight*

Implementasi pada perhitungan nilai *output weight* didapatkan dari hasil perhitungan dari nilai *transpose hidden layer* dikalikan dengan matrik *invert*. Setelah itu, hasil dari perkalian tersebut dikalikan dengan nilai dari normalisasi pada nilai target. Berikut ini implementasi dari perhitungan nilai *output weight* ditunjukkan pada Kode Program 5.7.

```
1  public double[][] Btopi(double[][] Hplus, double[][] dataY)
2  {
3    double[][] y = new double[dataY.length][1];
4    double[][] hasilBT = new double[Hplus.length][1];
5       for (int i = 0; i < dataY.length; i++) {
6          y[i][0] = dataY[i][3];
10      }
11      for (int i = 0; i < Hplus.length; i++) {
12         for (int j = 0; j < y[0].length; j++) {
13            double total = 0;
14               for (int k = 0; k < Hplus[0].length; k++) {
15                 total = total + (Hplus[i][k] * y[k][j]);
16                 }
17                 hasilBT[i][j] = total;
18           }
19       }
20    return hasilBT;
21 }
```

Kode Program 5.7 Proses perhitungan nilai *output weight*

Pada Kode Program 5.7 terdapat kode program perhitungan *output weight* yang dapat dijelaskan sebagai berikut:

1. Baris 3 yaitu inisialisasi variabel (y) yang akan digunakan untuk menampung data.

2. Baris 4 yaitu inisialisasi variabel hasil beta.
3. Baris 5 s.d 6 yaitu perulangan untuk proses pengambilan data Y.
4. Baris 11 s.d 21 yaitu perulangan untuk proses perkalian antara hasil dari *moore-penrose pseudo invers* dengan data Y.

1.1.7 Implementasi Perhitungan Hasil *Output* Peramalan

Implementasi pada perhitungan *output* peramalan didapatkan dari hasil perkalian antara nilai *output weight* dengan *output hidden layer*. Berikut ini hasil implementasi dari output peramalan ditunjukkan pada Kode Program 5.8.

```
1  public double[][] Ytes(double[][] fungsiHidLayer,double[][]
2                         bTopi) {
3     double[][] hasilYtes = perkalianMatrik(fungsiHidLayer,
4                                bTopi);
5     return hasilYtes;
6  }
```

Kode Program 5.8 Proses perhitungan nilai *output* peramalan

Pada Kode Program 5.7 terdapat kode program perhitungan *output weight* yang dapat dijelaskan sebagai berikut:

1. Baris 3 yaitu proses untuk menghitung peramalan dengan mengkalikan hasil *fungsi hidden layer* dengan hasil dari matrik beta.

1.1.8 Implementasi Denormalisasi Data

Implementasi pada perhitungan denormalisasi data dilakukan dengan persamaan yang telah dijelaskan pada bab sebelumnya yaitu Bab 4. Berikut ini merupakan hasil implementasi dari proses denormalisasi data ditunjukkan pada Kode Program 5.9.

```
1  public double[][] Denormalisasi(double[][] ytes, double[][]
2                                  data) {
3     double[][] denor = new
4                        double[ytes.length][ytes[0].length];
5       for (int i = 0; i < ytes.length; i++) {
6         for (int j = 0; j < ytes[0].length; j++) {
7           denor[i][j] = ytes[i][j] * (getmax(getKolom(data,
8                         3)) - getmin(getKolom(data, 3)))
9                         +(getmin(getKolom(data, 3)));
10          }
11        }
12    return denor;
13 }
```

Kode Program 5.9 Proses perhitungan nilai denormalisasi data

Pada Kode Program 5.9 terdapat kode program perhitungan denormalisasi data yang dapat dijelaskan sebagai berikut:

1. Baris 3 yaitu inisialisasi variabel denormalisasi.
2. Baris 5 s.d 12 yaitu perulangan untuk proses perhitungan denormalisasi data.

1.1.9 Implementasi Perhitungan Nilai MAPE

Nilai MAPE digunakan untuk mengetahui tingkat kesalahan dalam metode dalam menyelesaikan permasalahan. Berikut ini implementasi dari perhitungan nilai MAPE ditunjukkan pada Kode Program 5.10.

```
1  public double MAPE(double[][] ytesdata, double[][] dataY,
2  int jumlah_dataLatih) {
3    double total = 0, mape;
4    double[][] hasil = new
5                       double[ytesdata.length][ytesdata[0].len
6                       gth];
7    for (int a = 0; a < ytesdata.length; a++) {
8       for (int b = 0; b < ytesdata[0].length; b++) {
9          hasil[a][b] = ((ytesdata[a][b] / dataY[a +
10 jumlah_dataLatih][3])*100);
11             total = total + hasil[a][b];
12       }
13   }
14   mape = Math.abs(total / ytesdata.length);
15   return mape;
16 }
```

Kode Program 5.10 Proses perhitungan nilai MAPE

Pada Kode Program 5.10 terdapat kode program perhitungan *output weight* yang dapat dijelaskan sebagai berikut:

1. Baris 4 yaitu inisialisasi variabel hasil mape dengan ukuran panjang matrik peramalan.
2. Baris 6 s.d 14 yaitu perulangan untuk proses perhitungan nilai mape.

1.1.10 Implementasi Inisialisasi Kromosom Pada Algoritme Genetika

Implementasi pada pemberian nilai inisialisasi kromosom pada algoritme genetika menggunakan *range* nilai [0-1]. Dimana nilai tersebut yang akan digunakan sebagai nilai bobot *random* pada ELM. Setiap gen yang diberikan pada kromosom sepanjang sembilan gen. Berikut ini adalah proses pemberian inisisalisasi kromosom pada algoritme genetika ditunjukkan pada Kode Program 5.11.

```
1   public double[][] inisialisasiPopulasiAwal(int individu,
2  int gen) {
3          double        randomKromosom[][]        =        new
4  double[individu][gen];
5          for (int i = 0; i < individu; i++) {
6              for (int j = 0; j < gen - 1; j++) {
7                  randomKromosom[i][j]       =        (double)
8  Math.random() * 1;
9              }
10         }
11         return randomKromosom;
12     }
```

Kode Program 5.11 Proses inisialisasi kromosom

Pada Kode Program 5.11 terdapat kode program perhitungan *output weight* yang dapat dijelaskan sebagai berikut:

1. Baris 3 yaitu inisialisasi variabel *random* kromosom.
2. Baris 5 s.d 12 yaitu perulangan untuk proses *random* kromosom sebanyak individu yang telah ditentukan.

1.1.11 Implementasi Proses *Crossover* Pada Algoritme Genetika

Implementasi pada proses *crossover* dilakukan dengan menggunakan metode *extended intermediate crossover*. Pembangkitan anak pada crossover dilakukan berdasarkan jumlah *crossover rate* (cr) dikalikan dengan nilai *popsize*. Sedangkan pemilihan *parent* dilakukan secara *random* dengan menggunakan fungsi *random*. Berikut ini proses crossover pada algoritme genetika ditunjukkan pada Kode Program 5.12.

```
1  public double[][] crossover(int popsize, double cr,
2                              double[][] populasi) {
3    int jumlahAnak = (int) Math.round (cr * popsize);
4    int jumlahParent = (jumlahAnak % 2 == 1 ? jumlahAnak + 1
5                        : jumlahAnak);
6    double parent[][] = new
7                        double[jumlahParent][populasi[0].lengt
8                        h];
9    double child[][] = new
10                       double[jumlahAnak][populasi[0].length]
11                       ;
12   double alpha[] = new double[populasi[0].length - 1];
13   int index[] = new int[jumlahParent];
14   for (int i = 0; i < alpha.length; i++) {
15      alpha[i] = randAngka(-0.25, 1.25);
16   }
17   for (int i = 0; i < jumlahParent; i++) {
18      index[i] = ThreadLocalRandom.current().nextInt(0,
19                 populasi.length - 1);
20     for (int j = 0; j < populasi[i].length - 1; j++) {
21        parent[i][j] = populasi[index[i]][j];
22     }
23   }
24   for (int i = 0; i < jumlahAnak; i++) {
25     for (int j = 0; j < populasi[i].length - 1; j++) {
26        if (i < jumlahAnak - 1) {
27          child[i][j] = parent[i][j] + (alpha[j] * (parent[i
28                        + 1][j] - parent[i][j]));
29        } else {
30          child[i][j] = parent[i][j] + (alpha[j] * (parent[i
31                        - 1][j] - parent[i][j]));
32        }
33      }
34   }
35
36     return child;
37 }
```

Kode Program 5.12 Proses *crossover*

Pada Kode Program 5.12 terdapat kode program perhitungan *crossover* yang dapat dijelaskan sebagai berikut:

1. Baris 3 yaitu inisialisasi jumlah anak dengan mengkalikan nilai cr dengan popsize.
2. Baris 4 yaitu inisialisasi jumlah *parent* dengan pengondisian jika *parent* ganjil maka akan ditambah satu parent lagi.
3. Baris 6 yaitu inisialisasi variabel *parent*.
4. Baris 9 yaitu inisialisasi variabel *child*.
5. Baris 12 yaitu inisialisasi variabel *alpha*.
6. Baris 13 s.d 16 yaitu perulangan untuk proses nilai *random alpha*.
7. Baris 17 s.d 23 yaitu perulangan untuk proses penunjukkan kromosom sebagai *parent*.
8. Baris 24 s.d 27 yaitu perulangan untuk proses menghitung *child* dengan metode *extended intermediate crossover*.

1.1.12 Implementasi Proses Mutasi Pada Algoritme Genetika

Implementasi pada proses mutase dilakukan dengan menggunakan metode *random mutation*. Pemilihan *parent* dilakukan secara acak dan nilai *offspring* didapatkan dari jumlah *mutation rate* (mr) dikalikan dengan jumlah *popsize*. Berikut ini proses mutase pada algoritme genetika ditunjukkan pada Kode Program 5.13.

```
1  public double[][] mutasi(int popsize, double mr, double[][]
2  populasi) {
3     int jumlahAnakMut = (int) Math.round (mr * popsize);
4     double parentMut[][] = new
5                   double[jumlahAnakMut][populasi[0].length];
6     int index[] = new int[jumlahAnakMut];
7     double max = populasi[0][0], min = populasi[0][0];
8      double r = randAngka(-0.1, 0.1);
9      for (int i = 0; i < jumlahAnakMut; i++) {
10        index[i] = ThreadLocalRandom.current().nextInt(0,
11                        populasi.length - 1);
12         for (int j = 0; j < populasi[i].length - 1; j++) {
13             parentMut[i][j] = populasi[index[i]][j];
14         }
15     }
16      for (int i = 0; i < parentMut.length; i++) {
17         index = ThreadLocalRandom.current().nextInt(0,
18                   parentMut[0].length - 1);
19          for (int j = 0; j < populasi.length; j++) {
20             if(j==0){
21                     max = populasi[j][index];
22                     min = populasi[j][index];
23             }
24             if (populasi[j][index] > max) {
25                     max = populasi[j][index];
26             }
27             if (populasi[j][index] < min) {
```

```
28                      min = populasi[j][index];
29                  }
30              }
31          childMut[i][index] = parentMut[i][index] + r * (max-
32                                  min);
33      } return parentMut;
34  }
```

Kode Program 5.13 Proses mutasi

Pada Kode Program 5.13 terdapat kode program perhitungan mutasi yang dapat dijelaskan sebagai berikut:

1. Baris 3 yaitu inisialisasi jumlah anak mutasi dengan mengkalikan nilai mr dengan popsize.
2. Baris 4 yaitu inisialisasi variabel *parent* mutasi.
3. Baris 6 yaitu inisialisasi variabel index.
4. Baris 7 yaitu inisialisasi variabel max dan min.
5. Baris 8 yaitu inisialisasi nilai *random* pada variabel r.
6. Baris 9 s.d 15 yaitu perulangan untuk proses penunjukkan kromosom sebagai *parent* mutasi.
7. Baris 16 s.d 32 yaitu perulangan untuk proses menghitung *child* mutasi dengan metode *random mutation.*

1.1.13 Implementasi Evaluasi Pada Algoritme Genetika

Implementasi evaluasi ini menggunakan nilai *fitness* yang diperoleh dari nilai MAPE. Berikut ini hasil dari implementasi evaluasi pada algoritme genetika ditunjukkan pada Kode Program 5.14.

```
1   public double[][] Totalpopulasi(double[][] populasi,
2   double[][] anakCros, double[][] anakMut, int data_latih) {
3       double[][] totalPopulasi = new double[populasi.length +
4                                   anakCros.length +
5                                   anakMut.length][populasi[0].
6                                   length];
7       for (int i = 0; i < populasi.length; i++) {
8          for (int j = 0; j < totalPopulasi[0].length; j++) {
9             totalPopulasi[i][j] = populasi[i][j];
10         }
11      }
12
13      for (int i = populasi.length; i < populasi.length +
14            anakCros.length; i++) {
15        for (int j = 0; j < populasi[0].length - 1; j++) {
16           totalPopulasi[i][j] = anakCros[i
17                                populasi.length][j];
18          }
19      }
20       for (int i = populasi.length + anakCros.length; i <
21                    totalPopulasi.length; i++) {
22         for (int j = 0; j < populasi[0].length - 1; j++) {
23
```

```
            totalPopulasi[i][j] = anakMut[i - populasi.length
                                    - anakCros.length][j];

        }
      }
      for (int i = 0; i < totalPopulasi.length; i++) {
        double temp[] = new double[totalPopulasi[i].length -
                        1];
        for (int j = 0; j < temp.length; j++) {
           temp[j] = totalPopulasi[i][j];
        }
        totalPopulasi[i][totalPopulasi[i].length - 1] =
        fitness(temp, data_latih);
       }
       return totalPopulasi;
   }

public double fitness(double[] kromosom, int data_latih) {
   double fitnes = 0;
   double[][] bobot = konvetBobot(kromosom);
   double[][] transBobot = transposeMatrik(bobot);
   double[] bias = randBias();
   double normal[][] = normalisasi(data);
   double normal_latih[][] = new
                             double[data_latih][normal[0].len
                             gth];
   double normal_uji[][] = new double[normal.length -
                           data_latih][normal[0].length];
   for (int i = 0; i < data_latih; i++) {
      for (int j = 0; j < normal_latih[0].length; j++) {
        normal_latih[i][j] = normal[i][j];
      }
   }
   for (int i = 0; i < normal.length-data_latih; i++) {
      for (int j = 0; j < normal_latih[0].length; j++) {
        normal_uji[i][j] = normal[i+data_latih][j];
      }
   }
   double[][] hinit = Hinit(normal_latih, transBobot, bias);
   double[][] fungsiHidLayer = fungsiHidenLayer(hinit);
   double[][] transHiddLayer = HT(fungsiHidLayer);
   double[][] HasilHTkaliH = HTkaliH(transHiddLayer,
                              fungsiHidLayer);
   double[][] hasilInvers = Invers(HasilHTkaliH);
   double[][] hasilHplus = Hplus(hasilInvers,
                           transHiddLayer);
   double[][] hasilBtopi = Btopi(hasilHplus, normal_latih);
   double[][] hinitTes = Hinit(normal_uji, transBobot,
bias);
   double[][] fungsiHidLayerTes =
fungsiHidenLayer(hinitTes);
   double[][] ytes = Ytes(fungsiHidLayerTes, hasilBtopi);
   double[][] denormalisasi = Denormalisasi(ytes, data);
   double[][] Y = YtesDataY(denormalisasi, data,
                  normal_latih.length);
   double hasil = MAPE(Y, denormalisasi);
   fitnes = 1 / (hasil + 1);
   return fitnes;
}
```

Kode Program 5.14 Proses perhitungan nilai evaluasi

Pada Kode Program 5.14 terdapat kode program perhitungan evaluasi yang dapat dijelaskan sebagai berikut:

1. Baris 3 yaitu inisialisasi variabel total populasi.
2. Baris 7 s.d 11 yaitu perulangan untuk penggabungan populasi awal.
3. Baris 13 s.d 19 yaitu perulangan untuk pengambilan anak dari proses *crossover*.
4. Baris 20 s.d 27 yaitu perulangan untuk pengambilan anak dari proses mutasi.
5. Baris 28 s.d 38 yaitu perulangan untuk proses penggabungan nilai *fitness* dengan kromosom.
6. Baris 41 yaitu inisialisasi variabel *fitness*.
7. Baris 42 yaitu inisialisasi variabel bobot.
8. Baris 43 yaitu inisialisasi variabel transbobot.
9. Baris 44 yaitu inisialisasi variabel bias.
10. Baris 45 yaitu inisialisasi variabel normalisasi.
11. Baris 46 yaitu inisialisasi variabel normalisasi data latih.
12. Baris 49 yaitu inisialisasi variabel normalisasi data uji.
13. Baris 51 s.d 55 yaitu perulangan untuk proses normalisasi data latih.
14. Baris 56 s.d 60 yaitu perulangan untuk proses normalisasi data uji.
15. Baris 61 yaitu inisialisasi variabel hinit.
16. Baris 62 yaitu inisialisasi variabel fungsi hidden layer (H).
17. Baris 63 yaitu inisialisasi variabel *transpose hidden layer* (H^T).
18. Baris 64 yaitu inisialisasi variabel hasil H^T dengan H.
19. Baris 66 yaitu inisialisasi variabel hasil *invers*.
20. Baris 67 yaitu inisialisasi variabel Hplus.
21. Baris 69 yaitu inisialisasi variabel beta.
22. Baris 70 yaitu inisialisasi variabel Hinit untuk *testing*.
23. Baris 72 yaitu inisialisasi variabel *fungsi hidden layer* pada *testing*.
24. Baris 74 yaitu inisialisasi variabel Y *output.*
25. Baris 75 yaitu inisialisasi variabel denormalisasi data.
26. Baris 76 s.d 78 yaitu proses perhitungan nilai mape.
27. Baris 75 yaitu proses perhitungan nilai *fitness*.

1.1.14 Implementasi Seleksi Pada Algoritme Genetika

Pada implementasi seleksi yang digunakan pada algoritme genetika adalah seleksi elitism. Dimana nilai *fitness* akan diurutkan mulai dari nilai terbesar sampai dengan nilai terkecil. Berikut ini adalah hasil implementasi seleksi pada algoritme genetika ditunjukkan oleh Kode Program 5.15.

```
1  public double[][] seleksi(double[][] kromosom) {
2    double temp[] = new double[kromosom.length];
3    double newKrom[][] = new
4                         double[kromosom.length][kromosom[0].
5                         length];
6    for (int i = 0; i < kromosom.length; i++) {
7       temp[i] = kromosom[i][kromosom[0].length - 1];
8    }
9    Arrays.sort(temp);
10   for (int i = 0; i < kromosom.length; i++) {
11     int indek = 0;
12     for (int j = 0; j < kromosom.length; j++) {
13       if (temp[i] == kromosom[j][kromosom[0].length - 1]) {
14          indek = j;
15       break;
16      }
17   }
18   newKrom[kromosom.length - i - 1] = kromosom[indek];
19  }
20
21    return newKrom;
22 }
```

Kode Program 5.15 Proses perhitungan nilai seleksi

Pada Kode Program 5.15 terdapat kode program perhitungan seleksi yang dapat dijelaskan sebagai berikut:

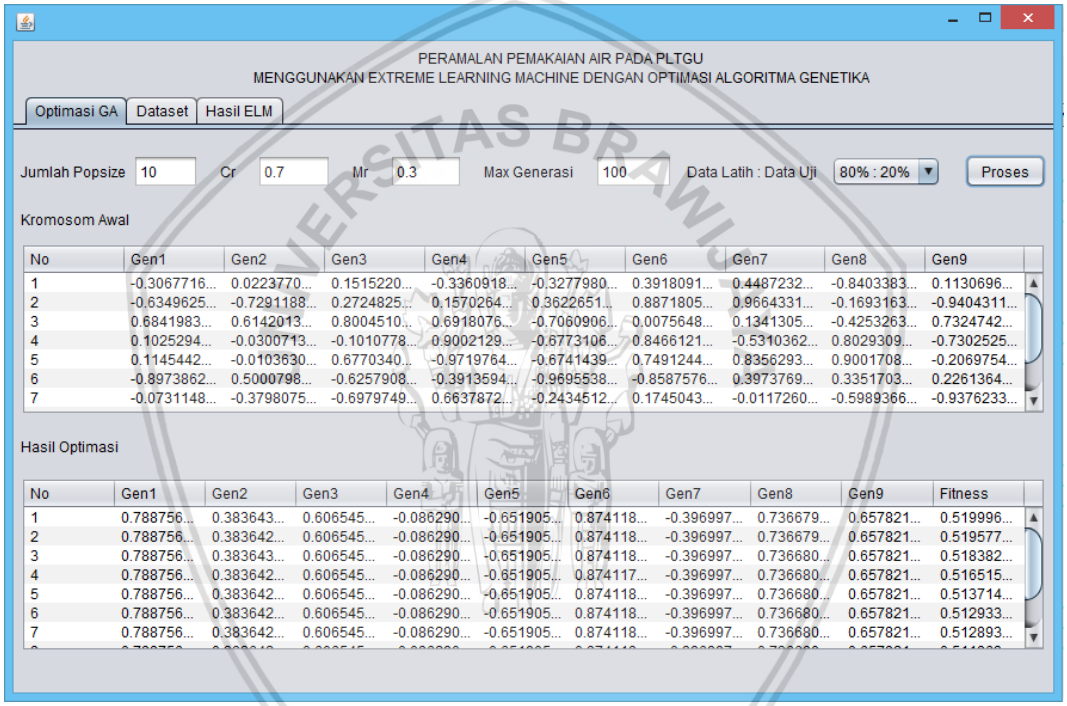
1. Baris 2 yaitu inisialisasi variabel temp.
2. Baris 3 yaitu inisialisasi variabel *new* kromosom.
3. Baris 6 s.d 8 yaitu perulangan untuk menyimpan kromosom kedalam variabel temp.
4. Baris 9 yaitu array yang digunakan untuk *sorting* data.
5. Baris 10 s.d 22 yaitu perulangan untuk proses mengurutkan data secara *descending*.

1.2 Implementasi Antarmuka

Implementasi antarmuka merupakan tampilan yang dibuat berdasarkan perancangan pada Bab 4. Pada impelementasi antarmuka pada peramalan pemakaian air terdapat 2 halaman yaitu halaman untuk optimasi bobot ELM, dan hasil ELM.

1.2.1 Implementasi Halaman Proses ELM Dengan Algoritme Genetika

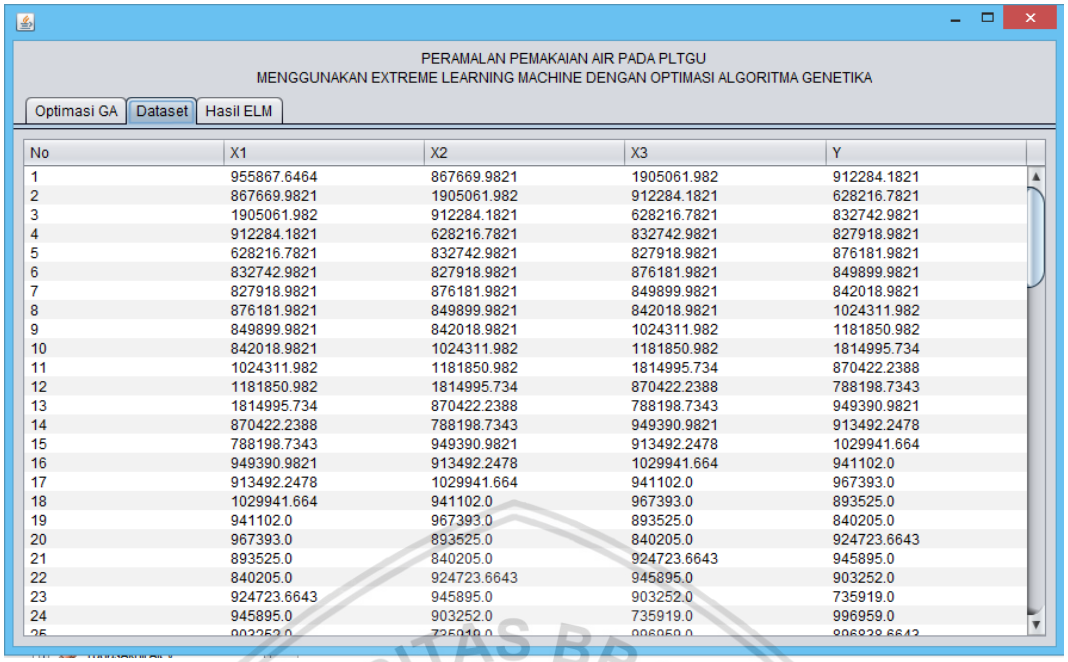
Implementasi halaman prose ELM dengan optimasi algoritme genetika menampilkan kromosom awal dan kromosom optimal. Dimana kromosom optimal didapatkan dari nilai nilai *fitness* tertinggi. Pada halaman utama ini terdapat beberapa *Text Field* parameter yang dapat diinputkan oleh pengguna diantaranya jumlah *popsize*, nilai *Crossover Rate* (*Cr*), nilai *Mutation Rate* (*Mr*), jumlah generasi, serta perbandingan jumlah data latih dan data uji untuk proses ELM. Kemudian terdapat tombol yang digunakan untuk memproses optimasi bobot yang didapatkan dari proses algoritme genetika. Berikut ini implementasi halaman prose ELM dengan optimasi algoritme genetika ditunjukkan pada Gambar 5.1.



Gambar 5.1 Halaman optimasi bobot ELM dengan Algoritme Genetika

1.2.2 Implementasi Halaman Dataset

Pada halaman dataset digunakan untuk menampilkan dataset hasil produksi air. Pada halaman ini terdapat *table* yang mempunyai 3 fitur diantaranya x1, x2, dan x3, sedangkan Y sebagai *output* hasil produksi air. Dataset yang digunakan mulai produksi air tanggal 1 januari 2017 sampai tanggal 31 juli 2017 yang diambil dari PT Pembangkitan Jawa Bali UP Gresik. Berikut ini halaman dataset ditunjukkan pada Gambar 5.2.

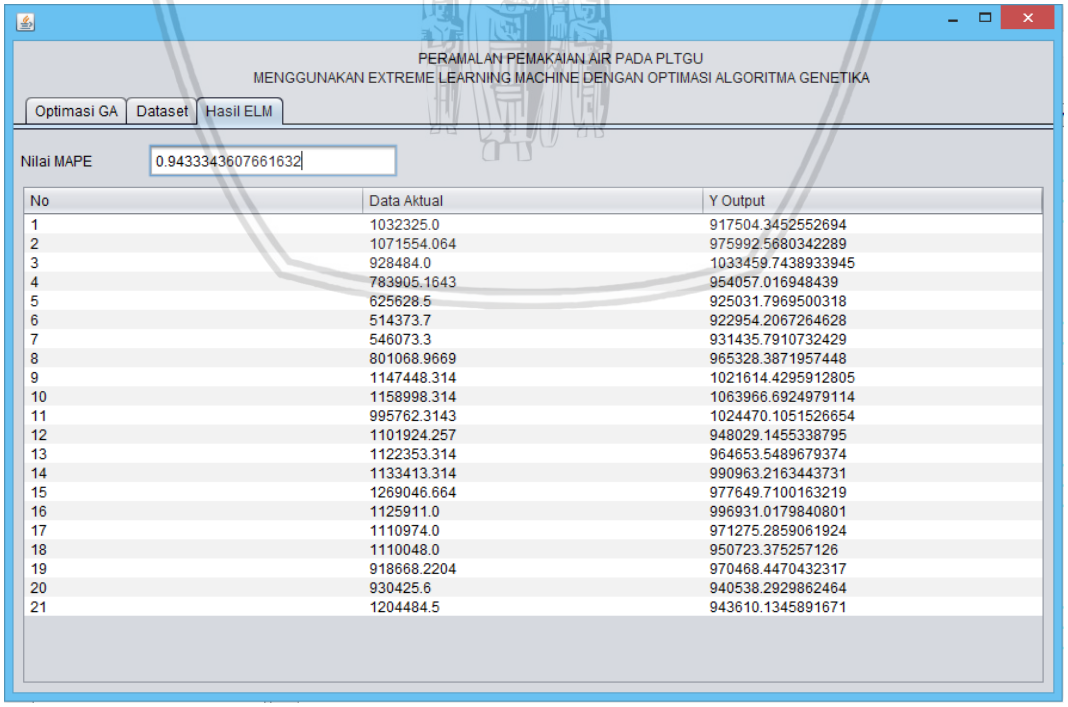


No	X1	X2	X3	Y
1	955867.6464	867669.9821	1905061.982	912284.1821
2	867669.9821	1905061.982	912284.1821	628216.7821
3	1905061.982	912284.1821	628216.7821	832742.9821
4	912284.1821	628216.7821	832742.9821	827918.9821
5	628216.7821	832742.9821	827918.9821	876181.9821
6	832742.9821	827918.9821	876181.9821	849899.9821
7	827918.9821	876181.9821	849899.9821	842018.9821
8	876181.9821	849899.9821	842018.9821	1024311.982
9	849899.9821	842018.9821	1024311.982	1181850.982
10	842018.9821	1024311.982	1181850.982	1814995.734
11	1024311.982	1181850.982	1814995.734	870422.2388
12	1181850.982	1814995.734	870422.2388	788198.7343
13	1814995.734	870422.2388	788198.7343	949390.9821
14	870422.2388	788198.7343	949390.9821	913492.2478
15	788198.7343	949390.9821	913492.2478	1029941.664
16	949390.9821	913492.2478	1029941.664	941102.0
17	913492.2478	1029941.664	941102.0	967393.0
18	1029941.664	941102.0	967393.0	893525.0
19	941102.0	967393.0	893525.0	840205.0
20	967393.0	893525.0	840205.0	924723.6643
21	893525.0	840205.0	924723.6643	945895.0
22	840205.0	924723.6643	945895.0	903252.0
23	924723.6643	945895.0	903252.0	735919.0
24	945895.0	903252.0	735919.0	996959.0

Gambar 5.2 Halaman dataset

1.2.3 Implementasi Halaman Hasil ELM

Pada implementasi halaman hasil ELM digunakan untuk menampilkan nilai MAPE dengan hasil peramalan. Pada halaman ini menampilkan data aktual dengan data hasil peramalan kemudian terdapat *text field* yang digunakan untuk menampilan MAPE. Berikut ini halaman hasil ELM ditunjukkan pada Gambar 5.3.



No	Data Aktual	Y Output
1	1032325.0	917504.3452552694
2	1071554.064	975992.5680342289
3	928484.0	1033459.7438933945
4	783905.1643	954057.016948439
5	625628.5	925031.7969500318
6	514373.7	922954.2067264628
7	546073.3	931435.7910732429
8	801068.9669	965328.3871957448
9	1147448.314	1021614.4295912805
10	1158998.314	1063966.6924979114
11	995762.3143	1024470.1051526654
12	1101924.257	948029.1455338795
13	1122353.314	964653.5489679374
14	1133413.314	990963.2163443731
15	1269046.664	977649.7100163219
16	1125911.0	996931.0179840801
17	1110974.0	971275.2859061924
18	1110048.0	950723.375257126
19	918668.2204	970468.4470432317
20	930425.6	940538.2929862464
21	1204484.5	943610.1345891671

Gambar 5.3 Halaman hasil ELM

BAB 6 PENGUJIAN DAN ANALISIS

Pada Bab pengujian menjelaskan tentang hasil dari pengujian sistem peramalan pemakaian air dengan menggunakan *extreme learning machine* dengan optimasi algoritme genetika. Pengujian sistem tersebut dilakukan dengan 5 tahap pengujian, diantaranya pengujian nilai *crossover rate* dan *mutation rate*, pengujian jumlah *popsize*, pengujian jumlah generasi, pengujian jumlah data *training*, dan pengujian jumlah fitur.

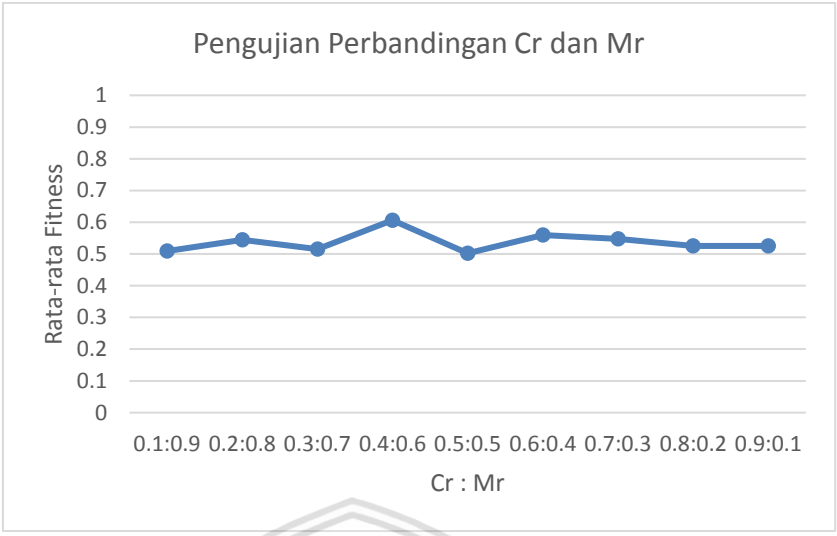
1.1 Pengujian Perbandingan Nilai *Crossover Rate* (*Cr*) Dan *Mutation Rate* (*Mr*)

Pada pengujian ini dilakukan untuk menemukan nilai perbandingan *crossover rate* dan *mutation rate* yang tepat untuk mendapatkan solusi yang optimal. Hasil dari solusi optimal didapatkan dari nilai rata-rata *fitness* yang dihasilkan melalui 10 kali percobaan dengan masing-masing perbandingan nilai *crossover rate* (*Cr*) dan *mutation rate* (*Mr*). Pada percobaan *Cr* dan *Mr* yang dilakukan menggunakan interval 0 sampai dengan 1. Parameter yang digunakan pada pengujian ini diantaranya jumlah data *training* dan data *testing* sebesar 80%:20%, jumlah generasi sebesar 100, serta jumlah *popsize* sebesar 20. Berikut ini hasil dari pengujian perbandingan *Cr* dan *Mr* di tunjukkan pada Tabel 6.1.

Tabel 6.1 Pengujian perbandingan *Cr* dan *Mr*

Cr:Mr	Nilai *Fitness* Percobaan ke-							Rata-rata *fitness*
	1	2	3	4	5	…	10	
0.1:0.9	0.506	0.496	0.499	0.494	0.521	…	0.514	0.508
0.2:0.8	0.523	0.572	0.629	0.538	0.523	…	0.503	0.544
0.3:0.7	0.511	0.517	0.522	0.506	0.412	…	0.500	0.515
0.4:0.6	0.429	0.539	0.538	0.513	0.549	…	0.629	0.606
0.5:0.5	0.511	0.500	0.488	0.501	0.516	…	0.518	0.502
0.6:0.4	0.522	0.563	0.500	0.543	0.515	…	0.523	0.559
0.7:0.3	0.542	0.639	0.540	0.571	0.507	…	0.519	0.547
0.8:0.2	0.550	0.514	0.505	0.538	0.515	…	0.514	0.525
0.9:0.1	0.521	0.578	0.525	0.526	0.517	…	0.502	0.525

Berdasarkan hasil pengujian perbandingan nilai *Cr* dan *Mr* yang ditunjukkan pada Tabel 6.1 menghasilkan nilai rata-rata *fitness* dengan masing-masing perbandingan *Cr* dan *Mr*. Berikut ini Gambar 6.1 merupakan grafik pengujian hasil perbandingan *Cr* dan *Mr*.



Gambar 6.1 Pengujian perbandingan Cr dan Mr

Berdasarkan hasil pengujian yang telah ditunjukkan oleh grafik pada Gambar 6.1 dapat disimpulkan bahwa perbandingan nilai *Cr* dan nilai *Mr* pada algoritme genetika mempengaruhi hasil *fitness*. Pada hasil tersebut, nilai *Cr* dan *Mr* dengan rata-rata *fitness* tertinggi yaitu *Cr* sebesar 0.4 dan *Mr* sebesar 0.6 dengan rata-rata *fitness* sebesar 0.606. Sedangkan nilai rata-rata *fitness* terendah didapatkan dengan perbandingan nilai *Cr* sebesar 0.5 dan *Mr* sebesar 0.5 dengan rata-rata *fitness* 0.502.

Pada grafik tersebut, nilai rata-rata *fitness* tertinggi diperoleh dengan nilai *Cr* kecil dan nilai *Mr* yang besar. Hal tersebut dikarenakan pada proses *mutation rate* (Mr) yang tinggi memungkinkan algoritme genetika melakukan eksplorasi dan diversitas (keragaman) populasi yang tinggi, sedangkan untuk *crossover rate* (*Cr*) yang rendah memungkinkan tidak bisa secara efektif belajar dari generasi sebelumnya. Sehingga menyebabkan ruang pencarian tidak bisa dieksploitasi secara efektif.

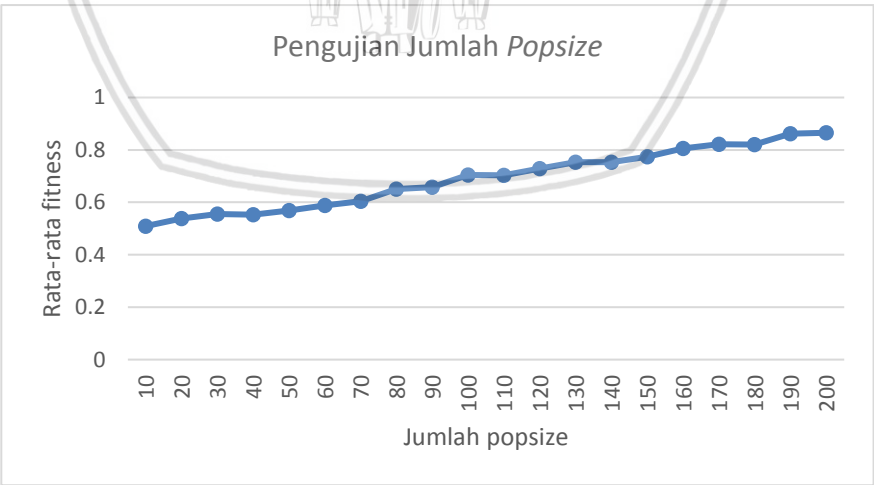
1.2 Pengujian Jumlah *Popsize*

Pengujian pada jumlah *popsize* bertujuan untuk mengetahui banyaknya *popsize* yang digunakan untuk mendapatkan solusi optimal. Hasil solusi optimal didapatkan dari perhitungan nilai rata-rata *fitness* yang dihasilkan setiap percobaan 10 kali dengan masing-masing jumlah *popsize*. Pengujian ini menggunakan jumlah *popsize* dengan kelipatan 10 sampai dengan 200. Parameter yang digunakan diantaranya perbandingan *Cr* dan *Mr* sebesar 0.4:0.6, perbandingan data *training* dan data *testing* sebesar 80%:20%, serta jumlah generasi sebesar 100. Tabel 6.2 berikut ini merupakan hasil pengujian jumlah *popsize.*

Tabel 6.2 Pengujian jumlah *popsize*

Popsize	Nilai *Fitness*							Rata-rata *fitness*
	Percobaan ke-							
	1	2	3	4	5	...	10	
10	0.500	0.501	0.550	0.499	0.530	...	0.498	0.509
20	0.545	0.741	0.527	0.484	0.514	...	0.500	0.538
30	0.505	0.788	0.520	0.543	0.494	...	0.496	0.555
40	0.532	0.567	0.571	0.506	0.565	...	0.520	0.553
50	0.580	0.550	0.554	0.528	0.528	...	0.528	0.568
60	0.600	0.544	0.568	0.569	0.538	...	0.568	0.588
70	0.552	0.560	0.729	0.558	0.721	...	0.547	0.604
80	0.623	0.620	0.541	0.848	0.559	...	0.999	0.650
90	0.997	0.582	0.586	0.745	0.608	...	0.629	0.658
100	0.630	0.596	0.603	0.615	0.570	...	0.650	0.704
...	...	...	...	...	...	...	...	...
200	0.692	0.997	0.989	0.999	0.656	...	0.967	0.865

Berdasarkan hasil pengujian yang ditunjukkan oleh Tabel 6.2 yaitu pengujian jumlah *popsize* dengan masing-masing nilai *fitness* yang dihasilkan oleh setiap percobaan. Berikut ini adalah grafik hasil pengujian jumlah *popsize* ditunjukkan oleh Gambar 6.2.



Gambar 6.2 Pengujian jumlah *popsize*

Pada grafik yang ditunjukkan oleh Gambar 6.2 menunjukkan bahwa hasil pengujian jumlah popsize dapat mempengaruhi hasil *fitness* yang didapatkan. Pada pengujian tersebut diperoleh nilai rata-rata *fitness* tertinggi sebesar 0.865

dengan jumlah *popsize* sebesar 200. Sedangkan nilai rata-rata *fitness* terendah sebesar 0.509 dengan jumlah *popsize* sebesar 10. Pada grafik hasil pengujian menunjukkan bahwa semakin bertambahnya jumlah *popsize* maka akan semakin meningkatnya nilai rata-rata *fitness* yang didapatkan. Hal tersebut dikarenakan semakin banyak jumlah *popsize* yang digunakan maka semakin bertambahnya keragaman tingkat individu yang terdapat pada populasi.

1.3 Pengujian Jumlah Generasi

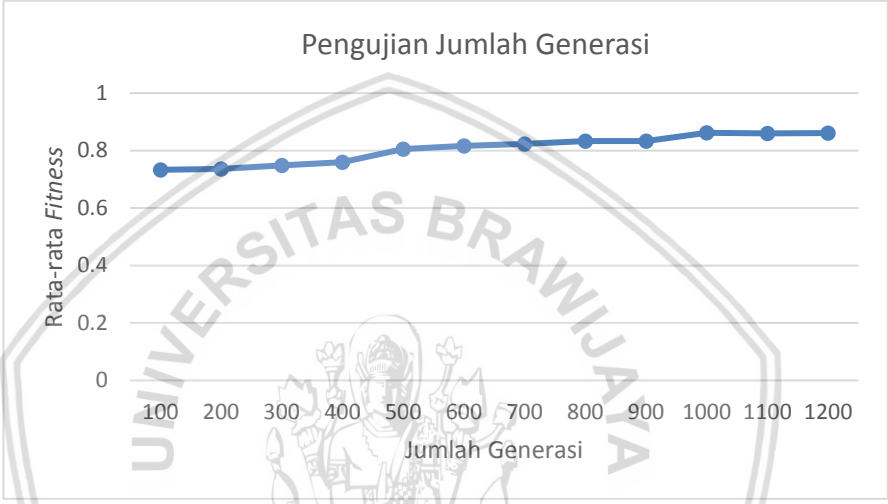
Pengujian pada jumlah generasi bertujuan untuk mengetahui hubungan antara jumlah generasi dengan *fitness* yang didapatkan. Hasil optimal didapatkan dari menghitung nilai rata-rata *fitness* yang dihasilkan setiap percobaan dengan masing-masing jumlah generasi. Berdasarnya jumlah generasi pada pengujian ini dimulai dari kelipatan 100 sampai 1200. Nilai *fitness* yang dihasilkan akan berubah-ubah karena terdapat nilai *random* pada proses algoritme genetika dan ELM. Pada pengujian ini menggunakan parameter jumlah *popsize* sebesar 200, nilai *Cr* dan *Mr* sebesar 0.4:0.6, serta perbandingan data *training* dan data *testing* sebesar 80%:20%. Pada Tabel 6.3 berikut ini merupakan hasil dari pengujian jumlah generasi.

Tabel 6.3 Pengujian jumlah generasi

Popsize	Nilai *Fitness*							Rata-rata *fitness*
	Percobaan ke-							
	1	2	3	4	5	…	10	
100	0.578	0.645	0.954	0.697	0.589	…	0.903	0.733
200	0.933	0.585	0.578	0.653	0.651	…	0.643	0.735
300	0.997	0.578	0.997	0.641	0.635	…	0.641	0.748
400	0.610	0.653	0.923	0.999	0.570	…	0.428	0.759
500	0.687	0.651	0.765	0.987	0.655	…	0.669	0.805
600	0.651	0.998	0.681	0.998	0.651	…	0.662	0.816
700	0.999	0.894	0.711	0.701	0.995	…	0.657	0.823
800	0.683	0.993	0.998	0.696	0.999	…	0.894	0.833
900	0.694	0.999	0.957	0.953	0.992	…	0.748	0.833
1000	0.670	0.998	0.754	0.999	0.999	…	0.619	0.860
1100	0.799	0.889	0.999	0.671	0.999	…	0.999	0.860
1200	0.979	0.889	0.999	0.671	0.999	…	0.999	0.860

Berdasarkan Tabel 6.3 pengujian generasi menunjukkan bahwa jumlah generasi dapat mempengaruhi nilai *fitness* yang didapatkan. Hasil pengujian tersebut dengan menggunakan jumlah generasi yang kecil menghasilkan nilai *fitness* yang kecil. Sedangkan meningkatnya jumlah generasi maka nilai *fitness*

cenderung meningkat. Dengan 10 kali percobaan pada pengujian jumlah maksimum generasi menghasilkan nilai rata-rata *fitness* tertinggi sebesar 0.861 terdapat pada jumlah generasi ke-1000. Untuk nilai rata-rata *fitness* terendah sebessar 0.733 terdapat pada jumlah generasi ke 100. Pada percobaan dengan jumlah generasi ke-1000 telah menemukan titik konvergen dan pada generasi berikutnya memiliki rata-rata *fitness* yang secara signifikan tidak jauh berbeda. Sehingga pada jumlah generasi ke-1000 telah menemukan solusi terbaik dan optimal. Hal tersebut karena semakin banyak jumlah generasi maka semakin banyak proses pencarian solusi untuk mendapatkan solusi yang terbaik. Gambar 6.3 berikut ini merupakan grafik hasil pengujian jumlah generasi.



Gambar 6.3 Pengujian jumlah generasi

1.4 Pengujian Jumlah Data *Training*

Pengujian ini bertujuan untuk mengetahui pengaruh jumlah data *training* terhadap pengenalan pola data, pelatihan serta nilai MAPE yang diperoleh. Pada pengujian ini dilakukan sebanyak 10 kali percobaan dengan masing-masing jumlah data *training* dan jumlah data *testing* yang telah ditetapkan. Parameter yang digunakan dari pengujian ini yaitu menggunakan perbandingan *Cr* dan *Mr* sebesar 0.4:0.6, jumlah *popsize* 200, serta jumlah generasi sebesar 1000. Hasil dari pengujian data *training* dan data *testing* ditunjukkan pada Tabel 6.4.

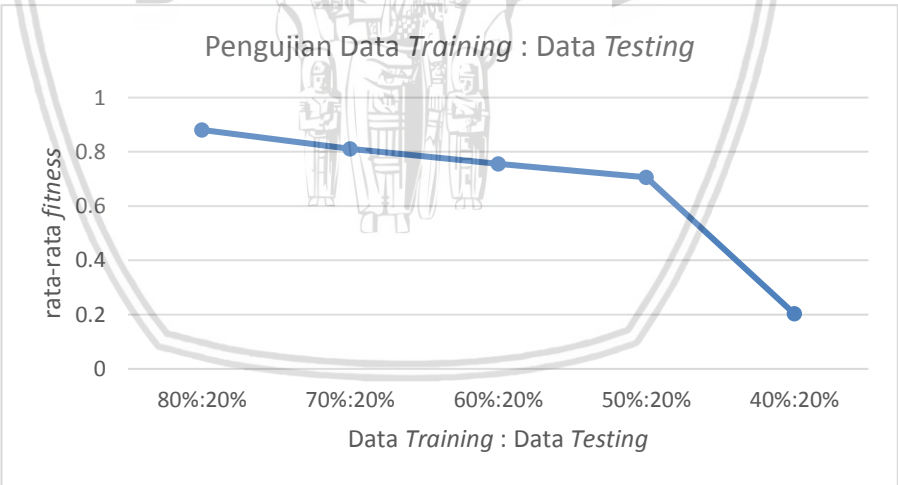
Tabel 6.4 Pengujian jumlah data *training*

Training:Testing	**Nilai *Fitness***							**Rata-rata *fitness***
	Percobaan ke-							
	1	**2**	**3**	**4**	**5**	**…**	**10**	
80%:20%	0.996	0.610	0.998	0.999	0.592	…	0.991	0.880
70%:20%	0.181	0.924	0.999	0.181	0.999	…	0.990	0.811
60%:20%	0.998	0.334	0.990	0.742	0.652	…	0.803	0.755

Tabel 6.4 Pengujian jumlah data *training* (lanjutan)

Training:Testing	Nilai *Fitness*							Rata-rata *fitness*
	Percobaan ke-							
	1	2	3	4	5	…	10	
50%:20%	0.999	0.268	0.996	0.266	0.999	…	0.997	0.706
40%:20%	0.215	0.215	0.283	0.283	0.174	…	0.261	0.203

Pada pengujian ini dilakukan pembagian data diantaranya data *training* dan data *testing* yang telah ditentukan. Pengujian pada data *training* bertujuan untuk melakukan pengenalan pola data dan pembelajaran pada data, sedangkan untuk data *testing* digunakan untuk proses pengujian pada hasil pengenalan pola dan pembelajaran data. Hasil pengujian perbandingan data yang ditunjukkan pada Tabel 6.4 mendapatkan nilai rata-rata *fitness* tertinggi sebesar 0.880 terdapat pada data *training* 80% sebesar 82 data dan data *testing* 20% sebesar 20 data. Untuk nilai rata-rata *fitness* terendah sebesar 0.203 yang terdapat pada perbandingan data *training* 40% dan data *testing* 20%. Hal tersebut dikarenakan pada ELM merupakan metode yang mengandalkan pelatihan pada data, sehingga perbandingan untuk data *training* harus lebih besar dari pada data *testing* untuk mendapatkan solusi yang optimal dan nilai MAPE yang kecil. Gambar 6.4 berikut ini merupakan grafik hasil pengujian perbandingan data *training* dan data *testing*.



Gambar 6.4 Pengujian jumlah data *training*

1.5 Pengujian Jumlah Fitur

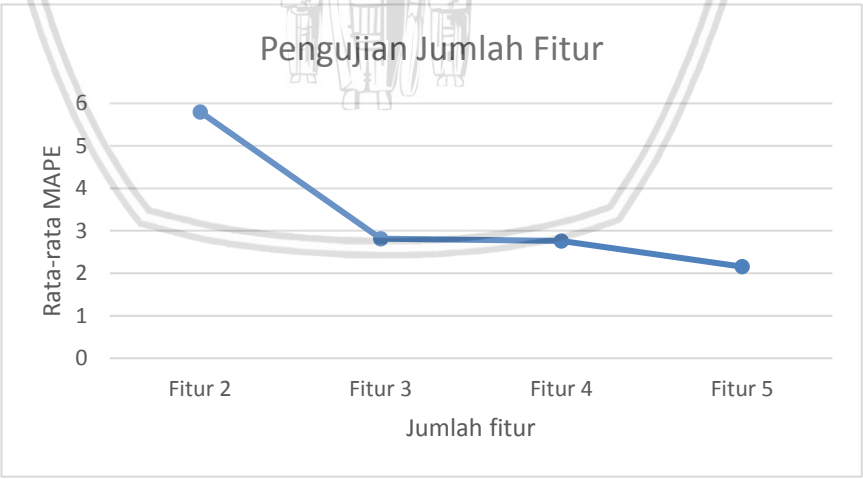
Pengujian jumlah fitur bertujuan untuk mengetahui pengaruh jumlah fitur terhadap pelatihan data dan nilai MAPE yang diperoleh. Pada pengujian ini dilakukan sebanyak 10 kali percobaan dengan masing-masing jumlah fitur yang telah ditetapkan. Parameter yang digunakan dari pengujian ini yaitu menggunakan perbandingan *Cr* dan *Mr* sebesar 0.4:0.6, jumlah *popsize* 200, jumlah generasi

sebesar 1000, dan jumlah data *training* dan data *testing* sebesar 80%:20%. Hasil dari pengujian jumlah fitur ditunjukkan pada Tabel 6.5.

Tabel 6.5 Pengujian jumlah fitur

Jumlah Fitur	Nilai MAPE							Rata-rata MAPE
	Percobaan ke-							
	1	2	3	4	5	…	10	
2	5.618	5.595	5.667	5.894	5.788	…	5.513	5.803
3	2.108	3.103	2.150	4.713	2.105	…	3.395	2.811
4	3.839	1.648	3.350	4.112	3.470	…	1.031	2.763
5	1.611	2.562	2.478	2.277	2.953	…	2.155	2.155

Pengujian pada jumlah fitur bertujuan untuk melakukan pembelajaran dalam mengenali pola data pemakaian air. Hasil pengujian jumlah fitur yang ditunjukkan pada Tabel 6.5 mendapatkan nilai rata-rata MAPE tertinggi sebesar 5.803 terdapat pada jumlah fitur 2. Untuk nilai rata-rata MAPE terendah sebesar 2.155 yang terdapat pada jumlah fitur 5. Hal tersebut dikarenakan pada ELM merupakan metode yang mengandalkan pelatihan pada data dan keterkaitan data satu dengan data lain sehingga membentu sebuah pola yang akan dijadikan sebagai pacuan dalam peramalan. Sehingga semakin banyak fitur yang digunakan maka semakin bagus untuk dijadikan acuan pembelajaran untuk mendapatkan nilai MAPE yang kecil. Gambar 6.5 berikut ini merupakan grafik hasil pengujian jumlah fitur.



Gambar 6.5 Pengujian jumlah fitur

1.6 Analisis Global Dari Hasil Pengujian

Pada penelitian ini dilakukan 4 macam pengujian terkait dengan metode ELM dan algoritme genetika. Pengujian yang dilakukan diantaranya pengujian perbandingan *Cr* dan *Mr*, pengujian jumlah *popsize*, pengujian jumlah generasi,

pengujian perbandingan data *training* dan data *testing* untuk mengetahui kinerja pada metode ELM, serta pengujian jumlah fitur pada ELM. Langkah awal pada pengujian ini yaitu melakukan pengujian pada perbandingan *Cr* dan *Mr* yang menujukkan bahwa perbandingan yang tepat dapat mempengaruhi nilai *fitness* yang diperoleh. Dari pengujian tersebut didapatkan perbandingan yang optimal serta mendapatkan nilai rata-rata *fitness* tertinggi sebesar 0.606 dengan *Cr* sebesar 0.4 dan *Mr* sebesar 0.6. Pada pengujian kedua yaitu pengujian jumlah *popsize* yang digunakan untuk mengetahui seberapa pengaruhnya jumlah *popsize* terhadap solusi yang diberikan. Pengujian jumlah *popsize* mendapatkan hasil yang optimal pada *popsize* 200 dengan nilai rata-rata *fitness* sebesar 0.865. Hasil dari pengujian tersebut menunjukkan bahwa semakin banyak *popsize* maka semakin beragam individu pada populasi yang diberikan sehingga mendapatkan solusi yang terbaik. Kemudian pada pengujian ketiga yaitu pengujian jumlah generasi yang bertujuan untuk mengetahui jumlah generasi untuk mendapatkan solusi yang optimal. Dari hasil pengujian jumlah generasi menunjukkan bahwa semakin banyak jumlah generasi maka nilai *fitness* yang diberikan semakin meningkat. Pada pengujian jumlah generasi memberikan hasil bahwa pada generasi ke-1000 mendapatkan nilai rata-rata *fitness* tertinggi sebesar 0.861. Sedangkan pengujian perbandingan data *training* dan data *testing* pada metode ELM bertujuan untuk mendapatkan data pelatihan yang tepat untuk membentuk sebuah pola data kemudian dilakukan pengujian terhadap data. Hasil dari pengujian tersebut didapatkan bahwa data *training* dan data *testing* yang tepat pada penelitian ini adalah 80%:20% dengan nilai rata-rata *fitness* sebesar 0.880. Dan pengujian jumlah fitur digunakan untuk mengetahui pengaruh jumlah fitur terhadap hasil peramalan dan tingkat MAPE yang diberikan. Dari hasil tersebut didapatkan bahwa semakin banyak jumlah fitur maka semakin kecil nilai MAPE yang diberikan.

Berdasarkan hasil pengujian parameter yang menghasilkan parameter optimal, maka dilakukan pengujian pada metode ELM dan ELM&Algoritme Genetika untuk membandingkan hasil evaluasi serta waktu eksekusi pada metode tersebut. Berikut ini hasil perbandingan metode ELM dan ELM&Algoritme Genetika ditunjukkan oleh Tabel 6.6.

Tabel 6.6 Pengujian perbandingan MAPE pada ELM dan ELM&Algoritme Genetika

No	ELM		ELM-Algoritme Genetika	
	MAPE	Waktu (detik)	MAPE	Waktu (detik)
1	3.633	0.078	0.268	155.033
2	5.301	0.078	0.170	145.642
3	4.232	0.078	0.129	143.666
4	4.401	0.078	0.330	144.471
5	3.558	0.078	0.814	142.571
6	4.094	0.078	0.453	141.625

7	5.506	0.109	1.022	146.241
8	4.978	0.093	0.340	145.837
9	4.144	0.094	0.577	147.940
10	5.331	0.094	0.175	146.268

Pada 10 kali percobaan didapatkan rata-rata hasil MAPE pada metode ELM sebesar 4.512 atau simpangan sebesar 193364.800 liter dengan rata-rata waktu eksekusi sebesar 0.085 detik, sedangkan pada metode ELM&Algoritme Genetika didapatkan rata-rata hasil MAPE sebesar 0.428 atau sebesar 85546.020 liter dengan rata-rata waktu eksekusi simpangan sebesar 145.929 detik. Selisih dari hasil MAPE pada ELM dan ELM&Algoritme Genetika sebesar 4.089. Sehingga dapat disimpulkan bahwa parameter-parameter yang digunakan pada setiap metode sangat mempengaruhi hasil yang diberikan, serta optimasi pada *input weight* menggunakan Algoritme Genetika mampu memberikan nilai MAPE lebih kecil pada permasalahan peramalan.

BAB 7 PENUTUP

Pada Bab ini menjelaskan tentang kesimpulan yang didapatkan dari tahap perancangan, implementasi, dan hasil dari pengujian sistem peramalan pemakaian air menggunakan *Extreme Learning Machine* dengan optimasi Algoritme Genetika.

1.1 Kesimpulan

Berikut ini kesimpulan yang diperoleh dari penelitian yang telah dilakukan:

1. Pada penelitian ini menggunakan metode *Extreme Learning Machine* (ELM) dengan optimasi Algoritme Genetika dapat diimplementasikan pada permasalahan peramalan pemakaian air pada PLTGU. Proses peramalan pemakaian air dilakukan dengan cara memasukkan jumlah produksi pemakaian air seminggu sekali. Langkah pertama yaitu data yang digunakan akan dibagi menjadi data *training* untuk membentuk pengenalan pola data dan dilakukan pelatihan sedangkan pada data *testing* untuk menguji data *training*, kemudian dilakukan perhitungan menggunakan metode ELM. Pada Algoritme Genetika digunakan untuk mengoptimasi *input weight* pada metode ELM. Nilai *fitness* pada proses Algoritme Genetika didapatkan dari hasil MAPE pada metode ELM. Sehingga hasil *fitness* yang didapatkan dengan nilai tertinggi akan dijadikan sebagai *input weight* yang akan digunakan pada proses peramalan. Algoritme genetika dapat digunakan untuk mendapatkan *input weight* yang optimal dengan menggunakan *real code*. Pada penelitian ini menggunakan jumlah gen sebesar 9 gen dan setiap pembentukan individu merepresentasikan solusi yang akan dipilih sebagai *input weight* pada ELM. Setiap individu akan memiliki nilai *fitness* yang didapatkan dari nilai MAPE pada ELM, sehingga nilai *fitness* tertinggi akan dipilih sebagai solusi yang optimal untuk dijadikan sebagai *input weight*.
2. Berdasarkan hasil pengujian pada metode ELM dengan optimasi Algoritme Genetika telah didapatkan parameter-parameter terbaik yaitu perbandingan nilai *Cr* sebesar 0.4 dan *Mr* sebear 0.6, jumlah *popsize* sebesar 200, jumlah generasi sebesar 1000, perbandingan data *training* dan data *testing* sebesar 80%:20%, serta jumlah fitur pada ELM sebanyak 5 fitur. Dari parameter hasil pengujian mendapatkan nilai rata-rata *error* sebesar 0.428 atau simpangan sebesar 85546.370 liter dengan rata-rata waktu eksekusi sebesar 145.929 detik. Sedangkan dengan menggunakan metode ELM mendapatkan rata-rata *eror* sebesar 4.517 atau simpangan sebesar 193364.800 liter dengan rata-rata waktu eksekusi sebesar 0.085 detik. Hal tersebut dapat disimpulkan bahwa metode ELM&Algoritme Genetika mampu memberikan nilai *error* lebih kecil dari hanya menggunakan metode ELM.

1.2 Saran

Pada penelitian ini masih banyak kekurangan sehingga diharapkan dengan saran ini dapat membantu dalam pengembangan lebih lanjut terkait dengan penelitian ini, antara lain:

1. Pada penelitian ini masih menggunakan pengolahan data yang bersifat statis yaitu pada sistem ini bisa berjalan dengan data yang sudah tetap. Diharapkan pada penelitian selanjutnya, pengolahan data dibuat dinamis yaitu bisa di kurangi atau ditambah.
2. Pengujian pada penelitian ini hanya dilakukan 4 tahap yaitu pengujian Cr dan Mr, pengujian jumlah popsize, pengujian jumlah maksimum generasi, dan pengujian perbandingan data *training* dan data *testing*. Oleh karena itu, untuk penelitian mendatang dapat menambahkan parameter-parameter yang akan diuji.

DAFTAR PUSTAKA

Arifudin, R., 2012. Optimasi Penjadwalan Proyek Dengan Penyeimbangan Biaya. *Jurnal Masyarakat Informatika,* 2(4), pp. 1-14.

Armanda, R. S. & Mahmudy, W. F., 2016. Penerapan Algoritma Genetika Untuk Penentuan Batasan Fungsi Kenggotaan Fuzzy Tsukamoto Pada Kasus Peramalan Permintaan Barang. *Jurnal Teknologi Informasi dan Ilmu Komputer (JTIIK),* 3(3), pp. 169-173.

Armstrong, J. S. & Collopy, F., 1992. Error Measures For Generalizing About Forecasting Methods: Empirical Comparisons. *International Journal of Forecasting,* Volume 8, pp. 69-80.

Choudhury, M. R., Hsieh, M.-K., Vidic, R. D. & Dzombak, D. A., 2012. Corrosion Management In Power Plant Cooling Systems Using Tertiary-Treated Municipal Wastewater As Makeup Water. *Corrosion Science,* Volume 61, pp. 231-241.

Fardani, D. P., Wuryanto, E. & Werdiningsih, I., 2015. Sistem Pendukung Keputusan Peramalan Jumlah Kunjungan Pasien Menggunakan Metode Extreme Learning Machine (ELM). *Information System Engineering and Business Inteligence,* 1(1), pp. 33-40.

Hansun, S., 2012. Peramalan Data IHSG Menggunakan Fuzzy Time Series. *IJCCS,* 6(2), pp. 79-88.

Hermawanto, D., 2013. Algoritma Genetika dan Contoh Aplikasinya. *Imu Komputer,* pp. 1-10.

Hidayat, S. S., Kencana, I. P. E. N. & Jayanegara, K., 2013. Prediksi Pengguna Bus Trans Sarbagita Dengan Metode Adaptive Neuro Fuzzy Inference System. *E-Jurnal Matematika,* 2(3), pp. 46-52.

Huang, G.-B., Zhu, Q.-Y. & Siew, C.-K., 2006. Extreme learning machine: Theory and applications. *Neurocomputing,* Volume 70, p. 489–501.

Indroprasto & Suryani, E., 2012. Analisis Pengendalian Persediaan Produk Dengan Metode EOQ Menggunakan Algoritma Genetika Untuk Mengefisiensikan Biaya Persediaan. *Teknik ITS,* Volume 1, pp. 305-309.

Istiqara, K., Furqon, M. T. & I., 2018. Prediksi Kebutuhan Air PDAM Kota Malang Menggunakan Metode Fuzzy Time Series Dengan Algoritme Genetika. *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer,* 2(1), pp. 133-142.

Khotimah, B. K., Sari R, E. M. & Yulianarta, H., 2010. Kinerja Metode Extreme Learning Machine. *SimanteC,* 1(3), pp. 180-191.

Malhotra, R., Singh, N. & Singh, Y., 2011. Genetic Algorithms: Concepts, Design for Optimization of Process Controller. *Computer and Information Science,* 4(2), pp. 39-54.

Mishra, A. K. & Ramarabhu, S., 2011. Functionalized graphene sheets for arsenic removal and desalination of sea water. *Desalination,* Volume 282, pp. 39-45.

Purnomo, D. M. et al., 2015. Genetika Algorithm Optimization for Extreme Learning Machine based Mircoalgal Growth Forcasting of Chlamydomonas sp. *ICACSIS,* pp. 242-248.

Rozi, F. & Sukmana, F., 2016. METODE SIKLIS DAN ADAPTIVE NEURO FUZZY INFERENCE SISTEM UNTUK PERAMALAN CUACA. *JIPI (Jurnal Ilmiah Pendidikan Informatika),* 1(1), pp. 7-13.

Saparuddin, 2010. PEMANFAATAN AIR TANAH DANGKAL SEBAGAI SUMBER AIR BERSIH DI KAMPUS. *SMARTek,* 8(2), pp. 143-152.

Soman, S. S. & Zareipour, H., 2010. A Review of Wind Power and Wind Speed Forecasting Methods With Different Time Horizons. *IEEE,* pp. 1-8.

Sun, Z.-L., Choi, T.-M., Au, K. F. & Yu, Y., 2008. Sales Forecasting Using Extreme Learning Machine With Applications In Fashion Retailing. *Decision Support Systems,* Volume 46, pp. 411-419.

Suryadmaja, I. B., Noken, I. N. & Dharma, S. I. G. B., 2015. Karakteristik Pola Pemakaian Dan Pelayanan Air Bersih. *Spektran,* 3(1), pp. 20-29.

Varnamkhasti, M. J. & Lee, L. S., 2012. A Fuzzy Genetic Algorithm Based on Binary Encoding For Solving Multidimensional Kn. *journal of applied mathematics,* pp. 1-23.

Wan, C., Xu, Z., Dong, Z. Y. & Wong, K. P., 2014. Probabilistic Forecasting of Wind Power Generation Using Extreme Learning Machine. *IEEE TRANSACTIONS ON POWER SYSTEMS,* 29(3), pp. 1034-1044.

Widodo, A. W. & Mahmudy, W. F., 2010. Penerapan Algoritma Genetika Pada Sistem Rekomendasi Wisata Kuliner. *KURSOR,* 5(4), pp. 205-211.

Yang, H., Yi, J., Zhao, J. & Dong, Z., 2013. Extreme Learning Machine Based Genetic Algorithm And Its Application. *Neurocomputing,* Volume 102, pp. 154-162.

Yuningsih, A. & Masduki, A., 2011. Potensi Energi Arus Laut Untuk Pembangkit Tenaga Listrik Dikawasan Pesisir Flores Timur, NTT. *Ilmu dan Teknologi Kelautan Tropis,* 3(1), pp. 13-25.