repository.ub.ac.id

PREDIKSI JUMLAH KRIMINALITAS MENGGUNAKAN METODE *EXTREME LEARNING MACHINE* (STUDI KASUS DI KABUPATEN PROBOLINGGO)

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:

Sema Nabillah Dewi

NIM. 145150207111074

PROGRAM STUDI TEKNIK INFORMATIKA

JURUSAN TEKNIK INFORMATIKA

FAKULTAS ILMU KOMPUTER

UNIVERSITAS BRAWIJAYA

MALANG

2018

UNIVERSITAS BRAWIJAYA

PENGESAHAN

PREDIKSI JUMLAH KRIMINALITAS MENGGUNAKAN METODE *EXTREME LEARNING MACHINE* (STUDI KASUS DI KABUPATEN PROBOLINGGO)

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh :
Sema Nabillah Dewi
NIM: 145150207111074

Skripsi ini telah diuji dan dinyatakan lulus pada
04 Mei 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

repository.ub.ac.id

PERNYATAAN ORISINALITAS

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

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

Malang, 18 April 2018

Sema Nabillah Dewi

NIM: 145150207111074

UNIVERSITAS BRAWIJAYA

KATA PENGANTAR

Puji dan syukur penulis panjatkan kehadirat Allah SWT atas limpahan rahmat dan karunia-Nya, sehingga penulis dapat menyelesaikan skripsi yang berjudul **"Prediksi Jumlah Kriminalitas Menggunakan Metode *Extreme Learning Machine* (Studi Kasus Di Kabupaten Probolinggo)"** ini dengan baik dan benar. Shalawat serta salam semoga senantiasa tercurahkan kepada junjungan besar kita Nabi Muhammad S.A.W beserta keluarga dan para sahabat sekalian.

Dalam proses penyelesaian skripsi ini penulis mendapatkan banyak bantuan dan dukungan dari berbagai pihak. Oleh karena itu, penulis ingin menyampaikan rasa hormat dan terimakasih kepada:

1. Bapak Imam Cholissodin, S.Si, M.Kom dan Bapak Edy Santoso, S.Si, M.Kom selaku dosen pembimbing yang telah dengan sabar membimbing, memberi masukan, dan mengarahan penulis sehingga dapat menyelesaikan skripsi ini.
2. Seluruh pihak kepolisian Kabupaten Probolinggo terutama bagian reskrim yang telah membantu menyediakan data bagi penulis untuk menyelesaikan skripsi ini.
3. Seluruh dosen Informatika/Ilmu Komputer Universitas Brawijaya atas ketersediaannya mendidik dan memberikan ilmu kepada penulis selama masa perkuliahan.
4. Seluruh civitas akademika Informatika/Ilmu Komputer Universitas Brawijaya yang telah banyak membantu penulis selama menempuh studi di Informatika/Ilmu Komputer Universitas Brawijaya dan memberikan dukungan semangat bagi penulis selama penyelesaian skripsi ini.
5. Kedua Orang Tua penulis yaitu Bapak Ade Maman S.H dan Ibu Wiwik Himatul Adiniyah, adik penulis Sela Fitria Dewi beserta keluarga besar yang telah mendukung penulis dengan segala usahanya, mulai dari doa, nasehat, perhatian, kasih sayang, materi, semangat hidup, dan tauladan yang semata-mata untuk keberhasilan penulis.
6. Teman-teman Program Studi Informatika/Ilmu Komputer angkatan 2014 yang selalu memberikan dukungan, masukan dan kebersamaan selama Penulis menempuh pendidikan di Fakultas Ilmu Komputer Universitas Brawijaya.
7. Teman-teman Seitse Valgus yang memberi semangat dan dukungan kepada penulis selama pengerjaan skripsi di luar kampus.
8. Teman-teman Kopma Squad yang memberi dukungan kepada penulis dan seluruh teman-teman yang tidak disebutkan satu persatu telah memberikan semangat kepada penulis.
9. Semua pihak yang tidak dapat penulis sebutkan satu per satu, yang telah membantu dan terlibat baik secara langsung maupun tidak langsung dalam penulisan skripsi ini.

Penulis menyadari bahwa skripsi ini tidak lepas dari kesalahan dan kekurangan. Oleh karena itu, kritik dan saran dari berbagai pihak sangat penulis harapkan demi

penyempurnaan skripsi ini. Semoga skripsi ini dapat memberikan manfaat dan dapat menambah wawasan keilmuan khususnya bidang ilmu komputer.

Malang, 18 April 2018

Penulis

semanabillah888@gmail.com

repository.ub.ac.id

ABSTRAK

Tingkat kejahatan di Indonesia semakin merajalela. Ambisi masyarakat akan memiliki harta kekayaan dari jalan yang tidak wajar dengan melakukan tindak kriminalitas. Kriminalitas merupakan tindakan yang melanggar aturan undang-undang yang dapat meresahkan masyarakat. Setiap masyarakat memiliki resiko menjadi korban tindak kriminalitas. Semakin besar resiko yang dimiliki masyarakat menandakan semakin tidak amannya suatu daerah. Namun, tidak bisa dipastikan jumlah tindak kriminalitas dari waktu ke waktu karena jumlahnya yang tidak menentu. Hal ini menyebabkan pihak kepolisian mengalami kesulitan untuk mengatasi masalah tindak kriminalitas. Prediksi yang tepat dan akurat dapat membantu meminimalisir tidak kriminalitas yang akan terjadi. Penelitian ini bertujuan untuk memperoleh prediksi jumlah kriminalitas menggunakan metode *Extreme Learning Machine* (ELM). Berdasarkan implementasi dan pengujian yang dilakukan menggunakan data kriminalitas Polres Kabupaten Probolinggo tahun 2012 hingga 2017 diperoleh arsitektur jaringan yang maksimum yaitu jumlah fitur sebanyak 7, perbandingan rasio data yaitu 80%:20%, dan jumlah *neuron* pada *hidden layer* sebanyak 7 serta fungsi aktivasi sigmoid biner. Nilai *error* yang rendah dihitung menggunakan tingkat kesalahan *Mean Square Error* (MSE) yaitu sebesar 0,037662.

Kata Kunci: *Extreme Learning Machine*, kriminalitas, *Mean Square Error* (MSE)

UNIVERSITAS BRAWIJAYA

repository.ub.ac.id

ABSTRACT

The crime rate in Indonesia is highly increased. A lot of people want to become wealthy in a wrong way by commiting a crime. Criminality is an act that violates the rules of the law that can disturb the public. Every society has a risk of becoming a victim of crime. The greater the risk that the community has, the more unsafe their area is. However, the number of criminal acts cant't be ensured from time to time due to the uncertain number. This causes the police will having a trouble in resolving the criminal acts. A proper and accurate prediction can help minimizing criminal acts that will be happened. This research is intended to get predicted numbers of criminality using Extreme Learning Machine method (ELM). Based on the implementation and testing done by using crime data of Probolinggo District Police in 2012 until 2017, obtained the maximum network architecture that is the number of features as much as 7, the comparison of data ratio is 80%: 20%, and the number of neurons in the hidden layer as much as 7 and the binary sigmoid activation function. The low error value is calculated using the Mean Square Error (MSE) error rate and the result is 0.037662.

Keywords: *Extreme Learning Machine, Criminality, Mean Square Error (MSE)*

UNIVERSITAS BRAWIJAYA

DAFTAR ISI

repository.ub.ac.id
UNIVERSITAS BRAWIJAYA

repository.ub.ac.id
UNIVERSITAS BRAWIJAYA

DAFTAR GAMBAR

repository.ub.ac.id
UNIVERSITAS BRAWIJAYA

DAFTAR TABEL

repository.ub.ac.id
UNIVERSITAS BRAWIJAYA

BAB 1 PENDAHULUAN

1.1 Latar Belakang

Dewasa ini kriminalitas di Indonesia semakin merajalela. Kompleksitas masyarakat modern membutuhkan berbagai macam kebutuhan materil yang melimpah dan disertai ambisi-ambisi sosial yang tidak konstruktif. Keinginan dalam memiliki harta kekayaan yang diperoleh dari jalan yang tidak wajar, mendorong seseorang maupun kelompok melakukan tindak kejahatan. Kriminalitas atau kejahatan adalah suatu bentuk tindakan yang melanggar aturan undang-undang yang telah ditetapkan oleh pemerintahan dan menyimpang dari norma-norma sosial serta meresahkan masyarakat (Kartono, 1999). Beberapa bentuk tindak kriminal yaitu pencurian, penganiayaan, tindak asusila, pembunuhan, penipuan, korupsi dan lain sebagainya. Dilansir dari catatan Badan Pusat Statistik (BPS) tercatat bahwa jumlah tindak kriminalitas di Jawa Timur mengalami peningkatan yang sangat drastis selama periode 2014-2015. Probolinggo merupakan salah satu dari 38 Kabupaten/Kota di Jawa Timur yang memiliki jumlah kriminalitas cukup tinggi. Di Kabupaten Probolinggo pada tahun 2016 terdapat 982 kasus kejahatan yang dilaporkan dan mengalami kenaikan hingga 24,84% dari tahun sebelumnya (Kepolisian Resort Kabupaten Probolinggo, 2017).

Selama ini jumlah kriminalitas pada Kabupaten Probolinggo jumlahnya berbeda setiap bulannya. Hal tersebut membuat pihak kepolisian mengalami kesulitan dalam memprediksi tingkat kriminalitas dari waktu ke waktu, karena datanya cenderung *fluktuatif* (dinamis). Menurut Kasubaghumas Polres Kabupaten Probolinggo, AKP Ida Bagus, faktor yang mendorong seseorang atau kelompok melakukan tindak pidana yaitu disebabkan oleh kondisi lingkungan sekitar yang mengalami perubahan yang sangat cepat, menurunnya norma-norma dan sanksi sosial yang memberikan pengaruh buruk yang mengacu terjadinya disorganisasi dalam masyarakat. Untuk meminimalisir tingkat kriminalitas, pihak kepolisian membutuhkan metode khusus yang disertai perhitungan-perhitungan dalam memprediksi jumlah kriminalitas. Sehingga pihak kepolisian memperoleh prediksi yang akurat dan efektif dalam menentukan jumlah kriminalitas. Maka dari itu diperlukan suatu metode komputasi untuk memprediksi jumlah kriminalitas.

Terdapat beberapa penelitian menggunakan topik pembelajaran yang sama sebelumnya seperti penelitian yang dilakukan oleh Novi Indah Pradasari dan kawan-kawan (2013) tentang prediksi penyakit saluran pernafasan menggunakan Jaringan Syaraf Tiruan *Backpropagation*. Pada penelitian tersebut, menggunakan dua tahap yaitu tahap penelitian yang digunakan untuk melatih jaringan syaraf tiruan, dan tahap kedua yaitu tahap pengujian dengan tujuan untuk menguji jaringan yang telah dilatih. Pada aplikasi tersebut, memiliki 3 *neuron* keluaran dengan memperoleh ketepatan hasil pada iterasi 30000. Serta target *error* sebesar 0,0001 dan *learning rate* 0,1.

Raudlatul Munawarah, Oni Soesanto, dan M. Reza Faizal (2016) dengan menerapkan metode *Support Vector Machine* untuk diagnosa hepatitis, menyatakan bahwa hasil pembelajaran berasal dari pelatihan data tes fungsi hati. Pada penelitian ini hasil pelatihan data *training* dalam menyelesaikan masalah *Quadratic Programing* (QP) *Problem* mengalami kesulitan dalam pencarian nilai α.

Penelitian lainnya yaitu, Subhan Panji Cipta (2016) tentang penerapan algoritma E*volving Neural Network* untuk prediksi curah hujan, menyatakan bahwa tingkat akurasi yang dimiliki oleh algoritma E*volving Neural Network* (ENN) lebih baik dibandingkan algoritma lainnya. Pada penelitian ini menggunakan 50 populasi percobaan dengan memperoleh kondisi cukup stabil pada generasi ke 300 dari 2000 generasi, serta tingkat keakuratannya sebesar 85%.

Penelitian tentang prediksi yang sudah sering dilakukan biasanya jumlah iterasinya terlalu banyak sehingga membutuhkan waktu yang cukup lama. Oleh karena itu, penelitian ini akan mengaplikasikan sebuah metode baru dari jaringan syaraf tiruan (JST) yaitu *Extreme Learning Machine* (ELM). ELM merupakan salah satu algoritme pembelajaran baru pada jaringan syaraf tiruan *feedforward* menggunakan satu *hidden layer* atau dikenal dengan istilah *single hidden layer feedforward neural network* (SLFNs). Adapun kelebihan yang dimiliki ELM diantaranya kemampuan belajar yang sangat cepat dan memiliki nilai error yang rendah serta tingkat akurasi yang lebih baik dibandingkan dengan metode-metode lainnya (Mahdiyah, et al., 2015). Untuk itu, kemampuan ELM diharapkan dapat menjadi solusi dari permasalahan di atas dan dapat membantu penulis untuk mengetahui jumlah kriminalitas di suatu daerah serta menghasilkan ramalan atau prediksi yang lebih efektif, khususnya di Kabupaten Probolinggo.

1.2 Rumusan Masalah

Permasalahan yang terjadi berdasarkan latar belakang di atas dapat dirumuskan sebagai berikut:

1. Bagaimana memprediksi jumlah kriminalitas di Kabupaten Probolinggo dengan menerapkan metode *Extreme Learning Machine* (ELM).
2. Bagaimana tingkat *error* yang diperoleh dari hasil prediksi jumlah kriminalitas di Kabupaten Probolinggo menggunakan metode *Extreme Learning Machine* (ELM) yang diukur dari perhitungan *Mean Square Error* (MSE).

1.3 Tujuan

Tujuan yang ingin dicapai dari penelitian ini adalah sebagai berikut:

1. Mengimplementasikan metode *Extreme Learning Machine* (ELM) dalam memprediksi jumlah kriminalitas di Kabupaten Probolinggo.
2. Menguji tingkat *error* yang diperoleh dari hasil prediksi jumlah kriminalitas di Kabupaten Probolinggo menggunakan metode *Extreme Learning Machine* (ELM) yang diukur dari perhitungan *Mean Square Error* (MSE).

1.4 Manfaat

Manfaat dari penelitian ini adalah sebagai berikut:

1. Model struktur jaringan ELM yang diperoleh dari penelitian ini dapat digunakan untuk memprediksi jumlah kriminalitas di Kabupaten Probolinggo.
2. Memberikan informasi kepada pihak Kepolisian Kabupaten Probolinggo dalam mengambil kebijakan untuk meminimalisir jumlah kriminalitas di Kabupaten Probolinggo.

1.5 Batasan Masalah

Batasan masalah diberikan untuk memperoleh kejelasan dalam penelitian ini. Batasan masalah pada penelitian ini adalah sebagai berikut:

1. Aplikasi ini dibuat hanya untuk memprediksi jumlah kriminalitas di Kabupaten Probolinggo.
2. Data yang digunakan berasal dari POLRES Kabupaten Probolinggo menggunakan data jumlah kriminalitas bulanan selama enam tahun (tahun 2012 hingga 2017).

1.6 Sistematika Pembahasan

Sistematika pembahasan yang digunakan penulis untuk mempermudah pemahaman dari penelitian skripsi ini adalah:

BAB I: PENDAHULUAN

Pada bab ini memberikan gambaran umum mengenai isi penelitian seperti latar belakang, rumusan masalah, tujuan, manfaat, batasan masalah dan sistematika pembahasan.

BAB II: LANDASAN KEPUSTAKAAN

Pada bab ini berisi tentang penjelasan beberapa teori yang relevan terhadap pokok bahasan dan referensi yang mendasari penelitian ini.

BAB III: METODOLOGI PENELITIAN

Pada bab ini penulis menjabarkan tentang metode yang dilakukan penulis dalam memilih data dan menghitungnya. Pada bab ini penulis juga mendeskripsikan definisi operasional variabel dalam penelitian ini.

BAB IV: PERANCANGAN

Pada bab ini penulis menguraikan beberapa rancangan aplikasi prediksi jumlah kriminalitas berdasarkan analisis kebutuhan.

BAB V: IMPLEMENTASI

Pada bab ini berisi tentang implementasi sistem prediksi jumlah kriminalitas berdasarkan perancangan yang telah dibuat sebelumnya.

BAB VI: PENGUJIAN DAN ANALISIS

Pada bab ini penulis melakukan pengujian sistem untuk mengetahui fungsi-fungsi berjalan dengan baik atau tidak.

BAB VII: PENUTUP

Pada bab ini penulis menjelaskan serta memberikan kesimpulan dari keseluruhan hasil penelitian serta penulis juga mengharapkan saran demi kesempurnaan penelitian ini.

repository.ub.ac.id
UNIVERSITAS BRAWIJAYA

BAB 2 LANDASAN KEPUSTAKAAN

Pada bab ini akan dibahas mengenai implementasi dari *Extreme Learning Machine* (ELM) pada penelitian-penelitian sebelumnya. Beberapa teori terkait metode *Extreme Learning Machine* (ELM), kriminalitas, dan *Mean Square Error* (MSE) dalam menghitung tingkat *error* dari hasil prediksi jumlah kriminalitas yang juga akan dijelaskan dalam kajian pustaka.

2.1 Kajian Pustaka

Cukup banyak penelitian dalam memecahkan permasalahan prediksi dengan menerapkan metode *Extreme Learning Machine* (ELM). Salah satu penelitian sebelumnya yaitu mengenai peramalan jumlah kunjungan pasien yang dilakukan pada bagian Poli Gigi di Rumah Sakit DR. Wahidin Sudiro Husodo Mojokerto. Penelitian ini menggunakan 116 data testing dengan dievaluasi beberapa kali, kemudian diuji coba berdasarkan fungsi aktivasi sigmoid. Sehingga diperoleh nilai akurasi optimal dari hasil peramalan dan tingkat kesalahan yang rendah yaitu sebesar 0,027 dan *Epoch* sebanyak 500 serta 7 unit *hidden layer*. Penelitian tersebut menekankan nilai *Mean Squared Error* (MSE) terendah (Fardani, et al., 2015).

Penelitian selanjutnya dilakukan oleh Singh, R dan Balasundaram, S (2007) mengenai *time series prediction*, menyimpulkan bahwa ELM adalah metode prospektif dalam permasalahan *time series prediction.*

Kemudian penelitian dari Pangaribuan, J.J. (2016) yang membahas mengenai diagnosis penyakit diabetes melitus. Penelitian ini menggunakan metode ELM yang dilakukan untuk membantu dalam menentukan apakah seseorang terjangkit penyakit diabetes atau tidak. Penggunaan ELM pada penelitian ini cukup efektif dari segi kecepatan dan memiliki tingkat akurasi yang baik dengan tingkat kesalahan *Mean Squared Error* (MSE) sebesar 0,4036. Hasil peramalan terbaik dilihat dari tingkat kesalahan yang mendekati 0.

Penelitian lainnya yaitu mengenai kinerja metode *Extreme Learning Machine* (ELM) pada sistem peramalan. Penelitian ini menyimpulkan bahwa dibandingkan dengan metode konvensional lain salah satunya metode *Backpropagation*, hasil dari pembelajaran metode ELM lebih baik karena memiliki nilai *error* yang rendah. Nilai MSE yang dihasilkan dari metode ELM yaitu sebesar 1,100% sedangkan nilai MSE yang dihasilkan dari metode *Backpropagation* sebesar 3,1933% (Khotimah, et al., 2010)

Berdasarkan penelitian-penelitian terkait peramalan atau prediksi diatas, penerapan metode *Extreme Machine Learning* (ELM) sangat efektif dan efisien dalam penentuan prediksi dengan *learning speed* yang lebih cepat dan pemilihan *input weight* (bobot) secara acak. Maka penulis melakukan penelitian serupa yang membahas tentang prediksi jumlah kriminalitas dengan metode *Extreme Machine Learning* (ELM), sehingga dapat memudahkan pihak kepolisian dalam

menganalisis tingkat kriminalitas dalam satu tahun kedepan serta membantu dalam meminimalisir kriminalitas atau tindak kejahatan.

Tabel 2.1 Kajian Pustaka

No.	Pustaka	Objek	Metode	Hasil
1	(Fardani, et al., 2015)	Jumlah Kunjungan Pasien	*Extreme Learning Machine* (ELM)	Menghasilkan nilai akurasi yang optimal dan tingkat kesalahan yang rendah pada fungsi aktivasi sigmoid yaitu sebesar 0,027 dan *Epoch* sebanyak 500 serta 7 unit *hidden layer*.
2	(Singh & Balasundaram, 2007)	Prediksi *time series*	*Extreme Learning Machine* (ELM)	Analisa metode ELM yang prospektif dalam permasalahan prediksi *time series.*
3	(Pangaribuan, 2016)	Penyakit Diabetes Melitus	*Extreme Learning Machine* (ELM)	Diagnosis penyakit diabetes melitus dengan kecepatan perhitungan yang efektif dan nilai MSE sebesar 0,4036.
4	(Khotimah, et al., 2010)	Data *time series* dengan atribut yang terikat	*Extreme Learning Machine* (ELM)	Memberikan hasil peramalan yang lebih baik dengan perolehan rata-rata nilai MSE sebesar 1,100% menggunakan metode ELM. Sedangkan metode *Backpropagation* rata-rata nilai MSE yang dihasilkan sebesar 3,1933%.

2.2 *Forecasting*

Forecasting atau prediksi adalah suatu perkiraan sistematis mengenai sebuah proses tentang apa yang mungkin terjadi di periode selanjutnya berdasarkan sumber informasi periode sebelumnya dan saat ini yang diperoleh untuk memperkecil suatu kesalahan (Mendome, et al., 2016). Secara umum hasil prediksi berkaitan erat dengan ketidakpastian, sehingga harus memperhitungkan

faktor akurasi yang tidak akan selalu memperoleh hasil prediksi dengan akurasi 100%. Dengan adanya perbedaan waktu maka *forecasting* mempunyai peran penting untuk menentukan kapan akan terjadi suatu peristiwa sehingga dapat mempersiapkan segala tindakan yang dibutuhkan.

2.3 Kriminalitas

Kriminalitas merupakan suatu bentuk tindakan yang melanggar aturan undang-undang yang telah ditetapkan oleh pemerintahan dan menyimpang dari norma-norma sosial serta meresahkan masyarakat (Kartono, 1999). Secara kriminologi berbasis sosiologis, kriminalitas diartikan sebagai pola tingkah laku yang dapat meresahkan maupun merugikan masyarakat sebagai korban kejahatan.

Tindakan kriminal atau kejahatan biasanya dipandang dari perilaku manusia yang bertentangan dengan norma sosial, norma hukum dan norma agama yang berlaku di lingkungan tempat mereka tinggal. Beberapa bentuk tindak kriminal seperti:

a. Pencurian
   Istilah pencurian dapat diartikan diam-diam atau sembunyi-sembunyi. Pencuri merupakan orang yang melakukan pencurian. Sedangkan pencurian adalah suatu tindakan mengambil hak milik orang lain secara diam-diam.
b. Penganiayaan
   Penganiayaan adalah tindakan menyakiti atau melukai orang lain dengan unsur kesengajaan.
c. Pembunuhan
   Pembunuhan merupakan perbuatan keji yang bersangkutan dengan hilangnya nyawa seseorang. Sanksi hukuman yang diberikan kepada sang pelaku diantaranya hukuman mati atau pidana penjara seumur hidup atau pidana dengan jangka waktu tertentu, paling lama 20 (dua puluh) tahun.

2.4 Jaringan Syaraf Tiruan

Jaringan Syaraf Tiruan atau yang biasa dikenal dengan istilah *Artificial Neural Network* (ANN) merupakan salah satu bagian dari *machine learning.* Jaringan syaraf tiruan memiliki kemiripan dengan kondisi otak manusia yang terbentuk dari sebagian besar *neuron* yang memiliki hubungan sangat erat antara *neuron* satu dengan yang lain. (Siang, 2009). *Neuron-neuron* tersebut memiliki peranan penting dalam mengolah dan meneruskan informasi kepada *neuron* lainnya. Hubungan antar *neuron* ini disebut bobot (*weight*). Dalam jaringan syaraf tiruan terdapat tiga lapisan antara lain:

1. Lapisan Masukan (*Input Layer*)
   Unit-unit *input* diinisialisasikan sebagai node-node dalam *input layer* yang akan menerima masukan dari luar. Masukan atau *input* dari lapisan ini menggambarkan suatu masalah.
2. Lapisan Tersembunyi (*Hidden Layer*)

Node-node di dalam *hidden layer* biasa disebut unit-unit tersembunyi. *Hidden Layer* merupakan lapisan yang menghubungkan antara *input layer* dan *output layer*.

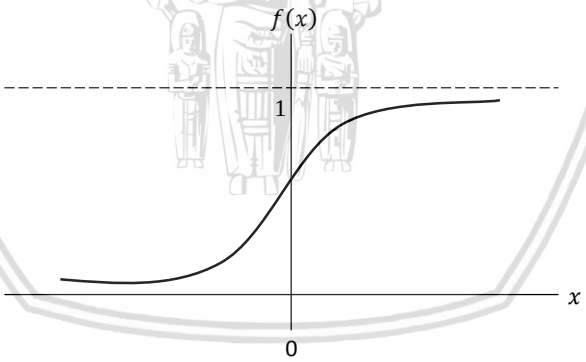
3. Lapisan Keluaran (*Output Layer*)
   Unit-unit *output* diinisialisasikan sebagai node-node dalam *output layer*, dimana keluaran yang dihasilkan merupakan *output* JST dari suatu permasalahan.

Dalam jaringan syaraf tiruan terdapat fungsi aktivasi yang dapat digunakan untuk menentukan *output* suatu *neuron* dengan berargumen *net input*. *Net input* terdiri dari kombinasi linier *input* beserta bobotnya (Siang, 2009). Fungsi ini memiliki tujuan untuk memodifikasi *output* kedalam rentang nilai tertentu. Berikut ini adalah fungsi aktivasi yang sering digunakan dalam jaringan syaraf (Kusumadewi, 2003) yaitu:

1. Fungsi *sigmoid biner*
   Fungsi *sigmoid biner* memiliki interval *output* 0 sampai 1 dengan membentuk kurva S yang dapat menghasilkan *output* lebih cepat. Fungsi *sigmoid biner* dapat dirumuskan seperti Persamaan 2.1 dan bentuk dari fungsi *sigmoid biner* digambarkan pada Gambar 2.1.

   $$f(x) = \frac{1}{1+e^{-x}} \quad (2.1)$$

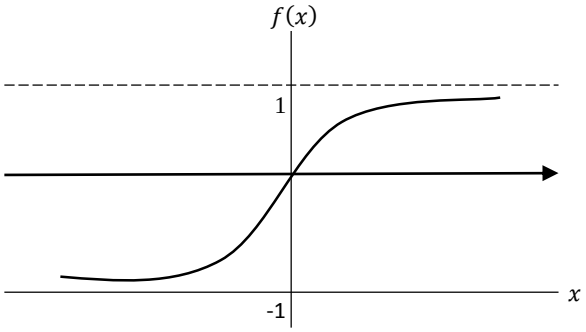
   Keterangan:
   $f(x)$ = Fungsi aktivasi sigmoid biner.
   e^{-x} = Eksponensial pangkat minus data ke- x.



Gambar 2.1 Fungsi Aktivasi *Sigmoid Biner*

2. Fungsi *sigmoid bipolar*
   Sama halnya dengan fungsi *sigmoid biner,* hanya saja fungsi *sigmoid bipolar* Memiliki rentang nilai antara -1 dan 1. Fungsi *sigmoid bipolar* dapat dirumuskan seperti Persamaan 2.2 dan bentuk dari fungsi *sigmoid bipolar* darpat dilihat pada Gambar 2.2.

   $$f(x) = \frac{1-e^{-x}}{1+e^{-x}} \quad (2.2)$$



Gambar 2.2 Fungsi Aktivasi *Sigmoid Bipolar*

2.5 Machine Learning

Machine Learning merupakan suatu cabang ilmu kecerdasan buatan pada sebuah sistem untuk mempelajari data-data yang telah diperoleh. Data yang diperoleh nantinya akan diaplikasikan dengan teknik-teknik *machine learning.* Tanpa adanya data, algoritma ELM tidak dapat bekerja. Terdapat dua data yaitu data pelatihan (*training*) dan data uji coba (*testing*). Data *training* digunakan untuk melatih sebuah algoritma, sedangkan data *testing* untuk mengetahui performa algoritma ketika menemukan atau memperoleh data baru yang sebelmnya tidak ada.

2.6 Metode Extreme Learning Machine

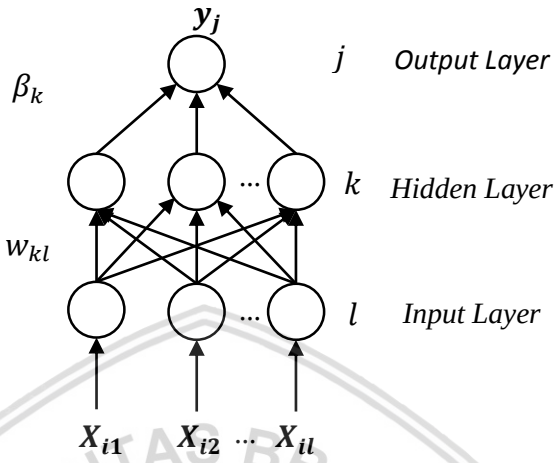
Extreme Learning Machine (ELM) merupakan sekumpulan metode pembelajaran yang diawasi. Metode ini adalah metode baru dari Jaringan Syaraf Tiruan atau *Artificial Neural Network* (ANN) diperkenalkan pertama kali oleh Huang (2004). ELM merupakan bagian dari jaringan syaraf tiruan *feedforward* dengan *Single Hidden Layer Feedforward neural Networks* (SLFNs) (Mahdiyah, et al., 2015).

Metode pembelajaran ini digunakan untuk mengatasi kelemahan pada jaringan syaraf tiruan (JST) *feedforward* yaitu *learning speed.* Alasan yang diutarakan Huang terhadap JST *feedforward* memiliki *learning speed* yang rendah yaitu:

1. Dalam melakukan training, menggunakan *slow gradient based learning algorithm.*
2. Parameter jaringan ditetapkan secara *iterative* menggunakan metode pembelajaran tersebut.
   Sedangkan pada pembelajaran JST seperti algoritma *backpropagation*, parameter ditentukan secara manual (Huang, et al., 2005).

Metode ini memiliki kelebihan dibanding metode yang sudah ada seperti *Support Vector Machine* (SVM) dan *Backpropagation* (BP) terutama dalam hal konsumsi atau penggunaan waktu dan performa. Pada metode ELM parameter *input weight* dipilih secara acak atau random. Dengan begitu ELM dapat menghasilkan *good generalization performance* dan memiliki *learning speed* yang

cepat. Metode ELM menggunakan Proses Pelatihan (*training*) dan Proses Uji coba (*testing*) sehingga dapat memperoleh hasil peramalan atau prediksi yang diinginkan. Konfigurasi sederhana algoritma ELM dapat dijelaskan pada Gambar 2.3.



Gambar 2.3 Struktur *Extreme Learning Machine*

Model ELM secara matematis lebih efektif dan sederhana dari jaringan syaraf tiruan *feedforward.* Model matematis dari ELM untuk N jumlah sampel yang berbeda (x_i, t_i) adalah sebagai berikut.

$$x_i = [x_{i1}, x_{i2}, \dots, x_{il}]^T \in \boldsymbol{R^l}$$
$$T = [t_{i1}, t_{i2}, \dots, t_{iN}]^T \in \boldsymbol{R^N}$$
$$X = [x_1, x_2, \dots, x_N]^T \in \boldsymbol{R^N}$$

Keterangan:

x_i = data ke-*i*

t_i = target data ke-*i*

X = keseluruhan data yang telah dinormalisasi

N = jumlah data

Langkah-langkah perhitungan metode ELM dibagi menjadi dua proses yaitu proses *training* dan proses *testing.*

2.6.1 Proses *Training*

Hal pertama yang harus dilakukan yaitu proses *training* pada metode ELM. Proses *training* dilakukan guna memperoleh *output weight* optimal atau memiliki tingkat kesalahan yang rendah. Tujuan dari proses *training* adalah untuk mengembangkan model ELM. Langkah-langkah *training* yang akan diproses adalah sebagai berikut:

1. Inisialisasi seluruh bobot (*input weight*) diatur dengan bilangan *random* yang kecil dari -1 hingga 1.

2. Keluaran di *hidden layer* dihitung dengan cara mengalikan masukan berupa data sejumlah N yang merupakan normalisasi data dalam bentuk matriks (X) dengan matriks *input weight* $k \times l$ yang diperoleh secara *random*, dimana k adalah jumlah *hidden* neuron dan l adalah jumlah *input* node. Perhitungan keluaran di *hidden layer* dapat dilihat pada Persamaan 2.3. Selanjutnya hitung keluaran *hidden layer* (H_{init}) dengan menggunakan fungsi aktivasi yang ditunjukkan pada Persamaan 2.1 untuk fungsi *sigmoid biner* dan Persamaan 2.2 untuk fungsi *sigmoid bipolar.*

   $$H_{init} = X.w^T \quad (2.3)$$

   Keterangan:

   H_{init} = Matriks *keluaran hidden layer*

   X = Matriks normalisasi data tanpa target

   w^T = Matriks *transpose input weight*
3. Setelah memperoleh keluaran *hidden layer* dengan fungsi aktivasi ($H(x)$). Kemudian menghitung H^+ yang merupakan *Moore Penrose Generalized Invers* yaitu untuk menghitung nilai invers dari matriks $H(x)$ dengan fungsi aktivasi (Mahdiyah, et al., 2015). Berikut ini Persamaan 2.4 dalam menghitung nilai *output weight* dari *hidden layer* ke *output layer.*

   $$\beta = H^+T \quad (2.4)$$

   Keterangan:

   β = Matriks *output weight* dari *hidden layer* ke *output layer*

   H^+= Matriks *Moore Penrose Generalized Invers* dari matriks H

   T = Matriks target

2.6.2 Proses *Testing*

Setelah proses *training,* diperoleh *output weight* yang optimal dari JST dengan metode ELM. Tujuan dari proses *testing* adalah mengevaluasi potensi metode ELM dalam hal prediksi. Langkah-langkah pada proses *testing* adalah sebagai berikut:

1. Inisialisasi *input weight* (diperoleh dari proses *training*)
2. Semua keluaran di *hidden layer* dihitung dengan fungsi aktivasi ($H_{(x)}$) menggunakan Persamaan 2.3.
3. Menghitung keluaran hasil prediksi (hasil keluaran pada *output layer*) yang dihitung dengan menggunakan Persamaan 2.5.

   $$y = H(x).\beta \quad (2.5)$$

   Keterangan:

   y = *Output layer* yang merupakan hasil prediksi

   β = *Output weight* yang diperoleh dari proses *training*

   $H(x)$ = Keluaran di *hidden layer* dihitung dengan fungsi aktivasi
4. Menghitung nilai *error* pada *output layer* yang dapat dilihat pada Persamaan 2.8.

2.7 Normalisasi Data

Normalisasi data merupakan metode *preprocessing* yang bertujuan untuk standarisasi seluruh data yang digunakan agar berada pada jarak tertentu (Patro & Sahu, 2015). Normalisasi data dilakukan karena memiliki nilai *input* dengan range tak terbatas, yaitu puluhan, ratusan bahkan ribuan. *Input* tersebut akan diubah ke nilai *output* dengan range terbatas, yaitu antara 0 dan 1, sehingga didapatkan nilai normalisasi yang kecil. Berikut adalah metode *Min-Max Normalization* yang digunakan dalam proses normalisasi data yang ditunjukkan pada Persamaan 2.6 (Jain & Bhandare, 2011).

$$x' = \frac{x - \min}{\max - \min} \tag{2.6}$$

Keterangan:

x' = Nilai data dari hasil normalisasi

x = Nilai asli data

min = Nilai minimum pada data set

max= Nilai maksimum pada data set

2.8 Denormalisasi Data

Agar dapat melihat hasil prediksi dengan jaringan syaraf ELM maka denormalisasi data perlu dilakukan untuk mendapatkan nilai asal dari hasil normalisasi (Siang, 2009). Berikut adalah penjabaran proses denormalisasi data pada Persamaan 2.7.

$$x = x'(\max - \min) + \min \tag{2.7}$$

Keterangan:

x' = Nilai data sebelum didenormalisasi

x = Nilai asli data setelah didenormalisasi

min = Nilai minimum pada data set

max= Nilai maksimum pada data set

2.9 *Mean Square Error* (MSE)

Mean Square Error (MSE) merupakan metode yang digunakan untuk mengevaluasi hasil prediksi. Masing-masing kesalahan dikuadratkan guna mengontrol kesalahan peramalan yang besar. Nilai MSE pada hasil prediksi ditunjukkan pada persamaan berikut (Hyndman & Koehler, 2005). Berikut adalah proses perhitungan nilai MSE pada Persamaan 2.8.

$$MSE = \frac{\sum_{i=1}^{n} {e_i}^2}{n} = \frac{\sum_{i=1}^{n} (y_i - t_i)^2}{n} \tag{2.8}$$

Keterangan:

n = Jumlah data

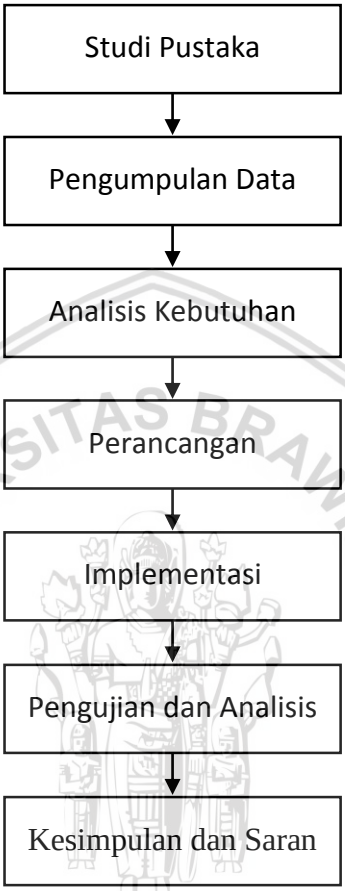
e_i = *Error*

y_i = Nilai *output* (prediksi)

t_i = Nilai aktual

BAB 3 METODOLOGI PENELITIAN

Pada bab ini akan diterapkan mengenai tahapan-tahapan dalam pembuatan sebuah perancangan, implementasi serta pengujian sistem. Tahapan-tahapan tersebut dapat dilihat dari diagram blok metode penelitian pada Gambar 3.1.



Gambar 3.1 Metode Penelitian

3.1 Studi Pustaka

Pada tahap studi pustaka dilakukan pengumpulan informasi atau data sebagai rujukan penelitian yang diperoleh dari berbagai macam sumber berupa buku, jurnal, *e-book*, paper penelitian dan internet. Kumpulan informasi tersebut berupa sistem prediksi jumlah kriminalitas di Kabupaten Probolinggo, pengenalan pola serta algoritma *Extreme Learning Machine.*

3.2 Pengumpulan Data

Tahap pengumpulan data menjelaskan mengenai kegiatan dalam pencarian data yang bertujuan memperoleh jawaban dari semua persoalan penelitian. Kumpulan data atau informasi pada tahap ini berupa kebutuhan dalam prediksi jumlah kriminalitas berupa data primer (data yang didapat langsung dari narasumber). Penelitian ini dilaksanakan di POLRES Kabupaten Probolinggo

dengan melakukan wawancara serta dokumentasi data yang diperoleh berupa pengetahuan mengenai jumlah kriminalitas dari tahun 2013 hingga tahun 2017. Data tersebut dapat diubah menjadi pola data pada Tabel 3.1.

Tabel 3.1 Pola Data yang Digunakan

X_1	X_2	X_3	target

X_1 : data jumlah kriminalitas 3bulan sebelumnya

X_2 : data jumlah kriminalitas 2bulan sebelumnya

X_3 : data jumlah kriminalitas 1bulan sebelumnya

3.3 Analisis Kebutuhan

Pada tahap analisis kebutuhan dilakukan analisa berbagai macam kebutuhan yang diperlukan sistem prediksi jumlah kriminalitas menggunakan metode *Extreme Learning Machine* (ELM) sesuai dengan permintaan *user* atau pengguna. Analisis kebutuhan memiliki sebutan lain yaitu kebutuhan proses. Keluaran yang diharapkan dari penelitian ini yaitu sistem dapat menampilkan data kriminalitas yang dihitung menggunakan metode ELM untuk memprediksi jumlah kriminalitas.

3.4 Perancangan

Tahap ini digunakan untuk mengimplementasikan metode *Extreme Learning Machine* dengan melakukan proses pelatihan dalam penentuan pola jaringan syaraf. Hal pertama yang harus dilakukan yaitu normalisasi pada data training, dilanjutkan dengan inisialisasi bobot yang dilakukan hingga diperoleh nilai keluaran. Nilai tersebut nantinya akan dibandingkan dengan target yang akan dituju. Selisih dari kedua nilai tersebut akan dimasukan pada nilai *error* dengan tujuan mengupdate bobot berikutnya yang dilakukan berulang kali sampai memperoleh nilai *Mean Square Error* (MSE) paling kecil.

3.5 Implementasi

Pada tahap implementasi menggunakan bahasa pemrograman java dalam pembuatan program serta didukung perangkat lunak lain yang didasarkan pada perancangan sistem sebelumnya. Pada implementasi dilakukan perhitungan *Extreme Learning Machine* (ELM) dengan bobot yang telah ditentukan.

3.6 Pengujian dan Analisis

Pada tahap ini dilakukan pengujian yang diharapkan dapat menghasilkan sistem yang berjalan dengan baik serta menganalisa hasil perolehan nilai MSE terendah menggunakan metode *Extreme Learning Machine.*

3.7 Kesimpulan dan Saran

Pada tahap penarikan kesimpulan merupakan tahapan terakhir berupa hasil dari pembuatan analisis dan pengujian sistem. Sedangkan saran merupakan

masukan-masukan yang dibuat untuk memperkecil nilai kesalahan sebagai acuan untuk pengembangan penelitian berikutnya.

BAB 4 PERANCANGAN

Bab perancangan menjelaskan mengenai uraian masalah, siklus penyelesaian prediksi jumlah kriminalitas menggunakan metode *Extreme Learning Machine* (ELM), perhitungan manual dan perancangan antar muka.

4.1 Deskripsi Masalah

Permasalahan yang diselesaikan adalah memprediksi jumlah kriminalitas di Kabupaten Probolinggo dengan menggunakan variabel unit masukan yaitu jumlah kejahatan. Kemudian dilakukan evaluasi nilai *error* berdasarkan hasil prediksi untuk memperoleh kualitas prediksi yang baik. Data yang akan dideskripsikan berasal dari instansi POLRES Kabupaten Probolinggo dengan data bulanan dari tahun 2012 hingga tahun 2017.

Untuk mengatasi masalah tersebut dilakukan perhitungan menggunakan metode ELM. Selain data jumlah kriminalitas, perhitungan ini juga menggunakan masukan berupa parameter jumlah *neuron* pada *hidden layer,* fungsi aktivasi, serta perbandingan antara jumlah data *training* dan data *tersting*. Sebelum memasuki proses ELM, hal yang harus dilakukan adalah proses normalisasi menggunakan *Min-Max Normalization* dari inisialisasi data *training* dan data *testing*. Selanjutnya dilakukan proses perhitungan prediksi dengan metode ELM. Kemudian masuk pada proses perhitungan nilai *error* yang dievaluasi menggunakan metode *Mean Square Error* (MSE). Perhitungan ini memperoleh keluaran berupa hasil prediksi yang telah didenormalisasi dan nilai *Mean Square Error* (MSE).

Dalam perhitungan manual, penelitian ini menggunakan 10 sampel data. Data kriminalitas yang digunakan adalah jumlah kriminalitas yang terjadi dalam setahun sebanyak 10 bulan. Data yang digunakan dapat dilihat pada Tabel 4.1.

Tabel 4.1 Data Jumlah Kriminalitas

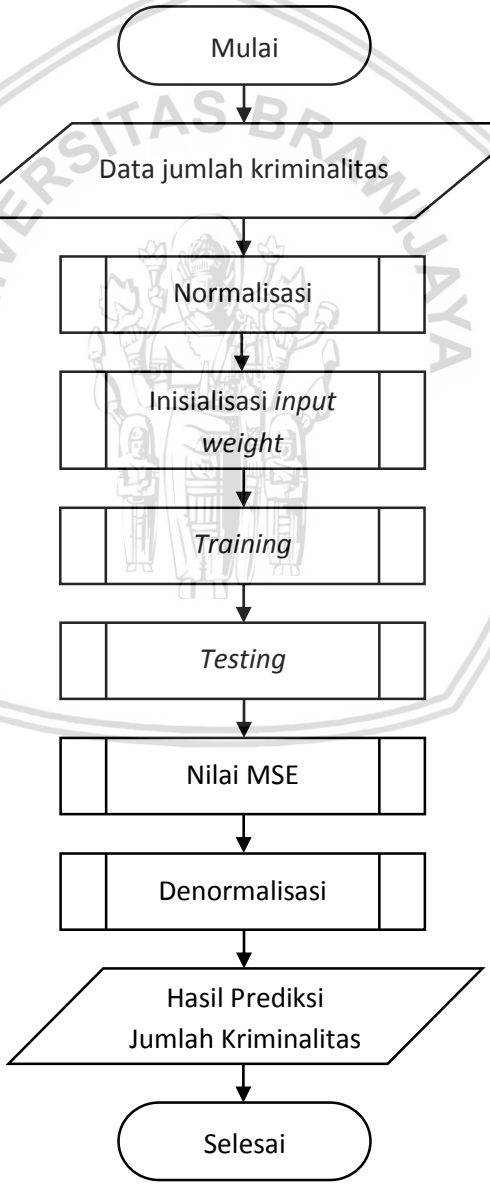
Data ke-	Bulan	Jumlah Kriminalitas
1	Januari 2012	60
2	Februari 2012	71
3	Maret 2012	71
4	April 2012	69
5	Mei 2012	68
6	Juni 2012	44
7	Juli 2012	57
8	Agustus 2012	62
9	September 2012	59
10	Oktober 2012	67

4.2 Siklus Metode *Extreme Learning Machine* (ELM)

Siklus Metode *Extreme Learning Machine* (ELM) sangat dibutuhkan untuk menjelaskan alur kerja algoritma ELM. Alur kerja yang dijelaskan mengenai proses normalisasi data menggunakan *Min-Max Normalization,* proses *Extreme Learning Machine*, proses perhitungan nilai *Mean Square Error* (MSE) dan proses denormalisasi data.

4.2.1 Proses *Extreme Learning Machine*

Proses *Extreme Learning Machine* (ELM) digunakan sistem untuk memprediksi jumlah kriminalitas dengan perolehan hasil prediksi mendekati target dan perolehan nilai *error* seminimal mungkin. Gambar 4.1 menunjukkan diagram alir proses ELM.



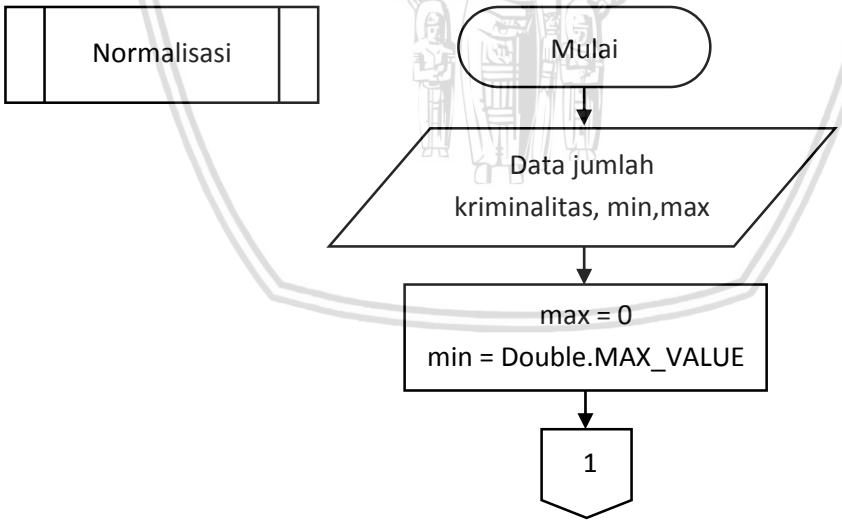
Gambar 4.1 Diagram Alir Proses *Extreme Learning Machine*

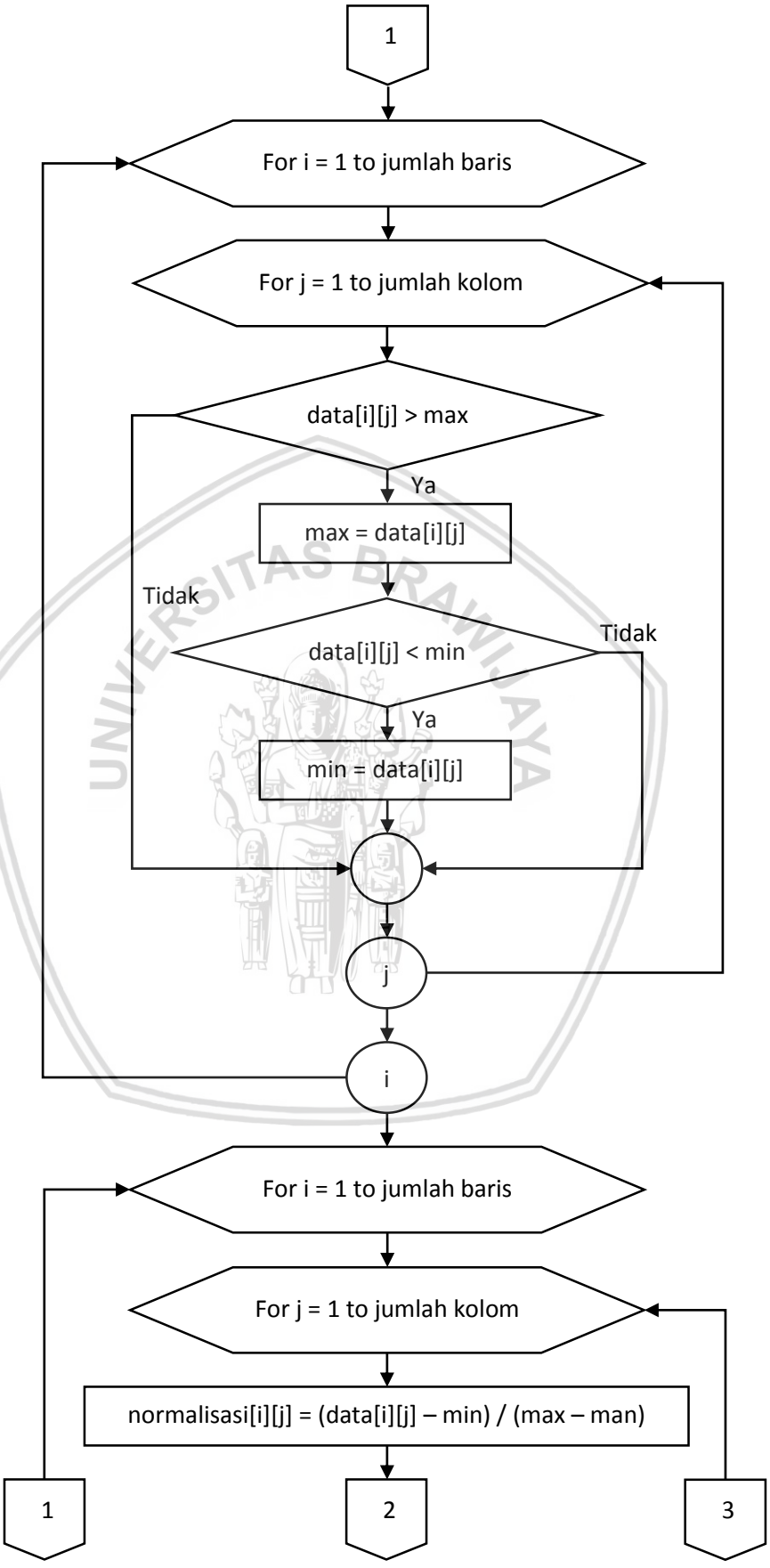
Proses *Extreme Learning Machine* pada Gambar 4.1 memiliki beberapa tahapan sebagai berikut:

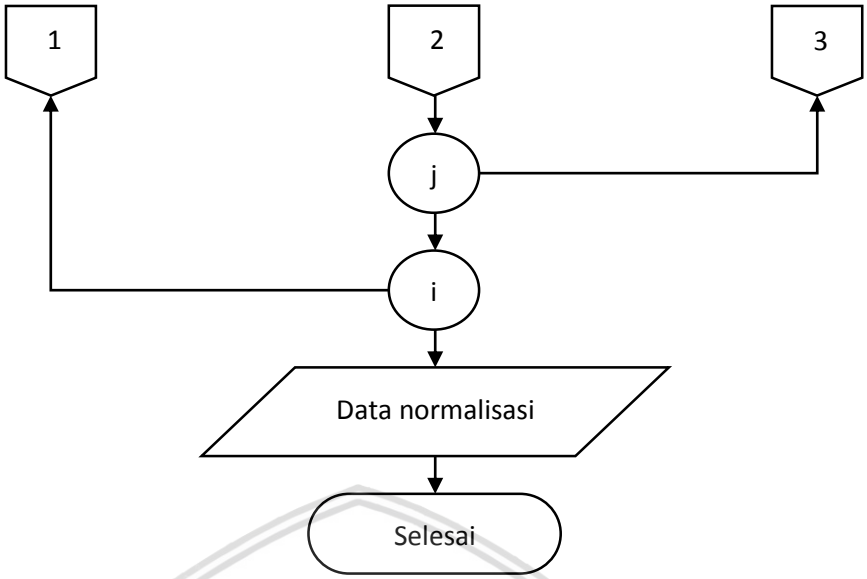
1. Masukan pada sistem berupa data jumlah kriminalitas
2. Melakukan perhitungan normalisasi data dengan Persamaan 2.6. Diagram alir proses normalisasi dapat dilihat pada Gambar 4.2
3. Inisialisasi *input weight* dapat dilihat pada Gambar 4.3
4. Melakukan proses *training* dengan data yang telah dinormalisasi. Diagram alir proses *training* dapat dilihat pada Gambar 4.4
5. Melakukan proses *testing* untuk menentukan hasil prediksi. Diagram alir proses *testing* dapat dilihat pada Gambar 4.14
6. Melakukan perhitungan nilai *Mean Square Error* (MSE) dengan Persamaan 2.8. Diagram alir proses perhitungan nilai MSE dapat dilihat pada Gambar 4.16
7. Melakukan perhitungan denormalisasi data dengan Persamaan 2.7. Diagram alir proses denormalisasi dapat dilihat pada Gambar 4.17
8. Keluaran berupa hasil prediksi jumlah kriminalitas yang telah didenormalisasi

4.2.2 Proses Normalisasi

Normalisasi merupakan proses dimana nilai data diubah ke skala tertentu. Pada penelitian ini dilakukan normalisasi data menggunakan metode *Min-Max Normalization* dengan merubah data menjadi lebih kecil yaitu range antara [0-1]. Gambar 4.2 menunjukkan diagram alir proses normalisasi data.



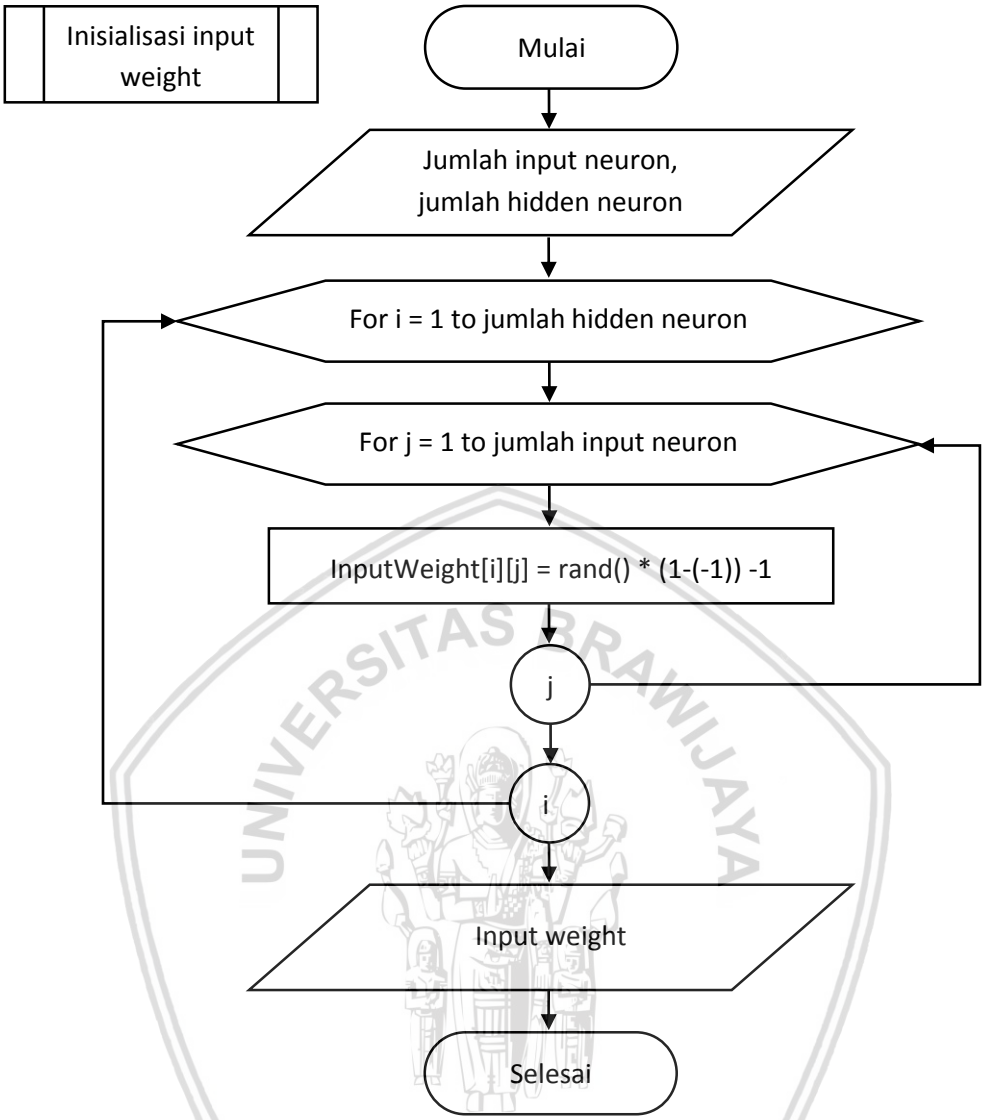

1
For i = 1 to jumlah baris
For j = 1 to jumlah kolom
data[i][j] > max
Ya
max = data[i][j]
Tidak
data[i][j] < min
Tidak
Ya
min = data[i][j]
j
i
For i = 1 to jumlah baris
For j = 1 to jumlah kolom
normalisasi[i][j] = (data[i][j] – min) / (max – man)
1
2
3



Gambar 4.2 Diagram Alir Proses Normalisasi

Berdasarkan Gambar 4.2, langkah-langkah proses normalisasi jumlah kriminalitas menggunakan metode *Min-Max Normalization* diuraikan sebagai berikut:

1. Masukan pada sistem berupa data jumlah kriminalitas yang akan dinormalisasi, nilai maksimum (max) dan nilai minimum (min)
2. Mencari nilai maksimum dan nilai minimum dari semua data yang telah dimasukkan
3. Melakukan perulangan sebanyak jumlah baris
4. Melakukan perulangan sebanyak jumlah kolom
5. Menghitung nilai normalisasi untuk setiap data masukan dengan rumus pada Persamaan 2.6
6. Keluaran berupa data jumlah kriminalitas yang telah dinormalisasi



Gambar 4.3 Diagram Alir Proses Inisialisasi *Input Weight*

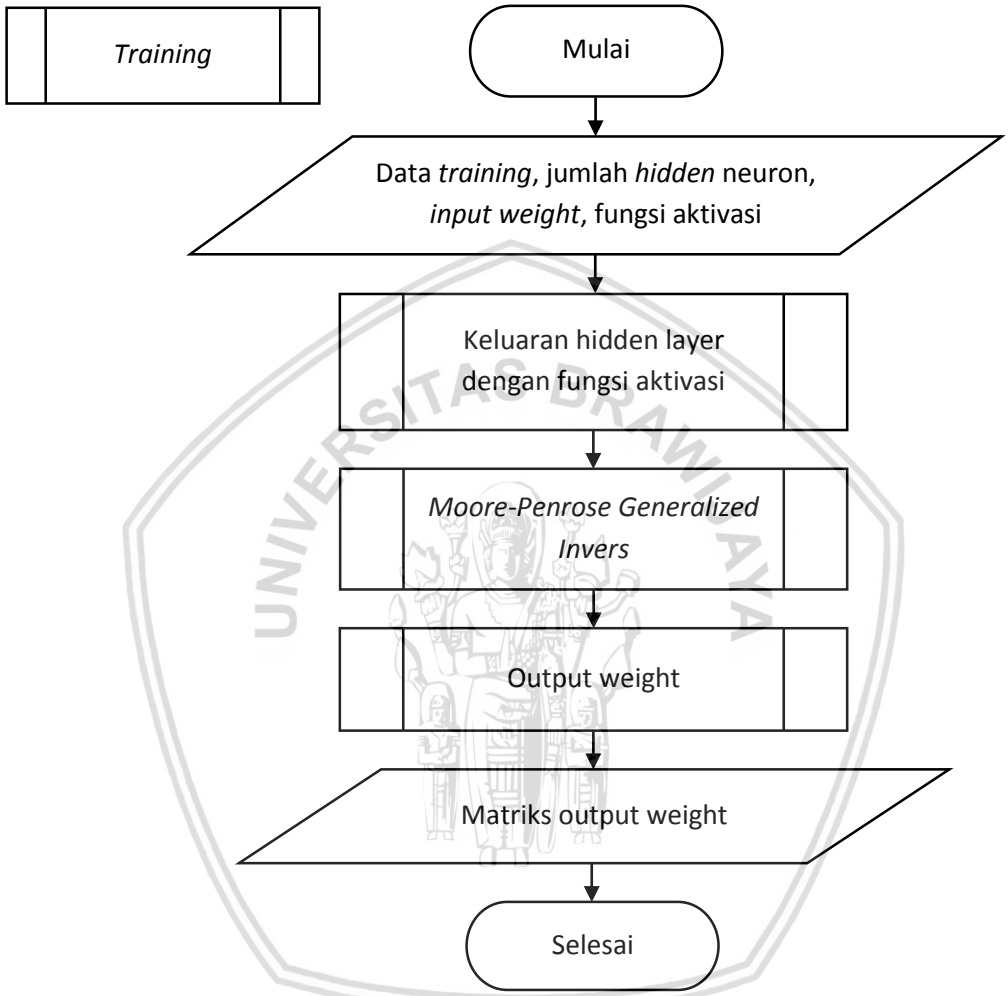
Proses inisialisasi *input weight* pada Gambar 4.3 memiliki beberapa tahapan sebagai berikut:

1. Melakukan perulangan sebanyak jumlah *hidden neuron*
2. Melakukan perulangan sebanyak jumlah *input neuron*
3. Menghitung nilai *input weight* menggunakan rumus random dengan range antara -1 dan 1

4.2.3 Proses *Training*

Proses *Training* merupakan proses pelatihan yang dilakukan guna memperoleh nilai *input weight,* dan *output weight* yang memiliki tingkat kesalahan yang rendah. Perolehan nilai tersebut nantinya akan digunakan pada proses *testing*. Langkah pertama yang dilakukan yaitu memasukkan nilai *input weight* yang telah diinisialisasi secara random. Kemudian menghitung keluaran

hidden layer dengan fungsi aktivasi (H_{init}) yang dilanjutkan dengan menghitung matriks *Moore-Penrose Generalized Invers* (H^{+}) dari hasil keluaran pada *hidden layer* menggunakan fungsi aktivasi. Hasil perkalian antara H^{+} dan target digunakan untuk mencari *output weight*. Setelah itu nilai *output weight* akan dibawa menuju proses *testing* untuk dilakukan perhitungan kembali. Gambar 4.4 menunjukkan diagram alir proses *training* menggunakan metode ELM.

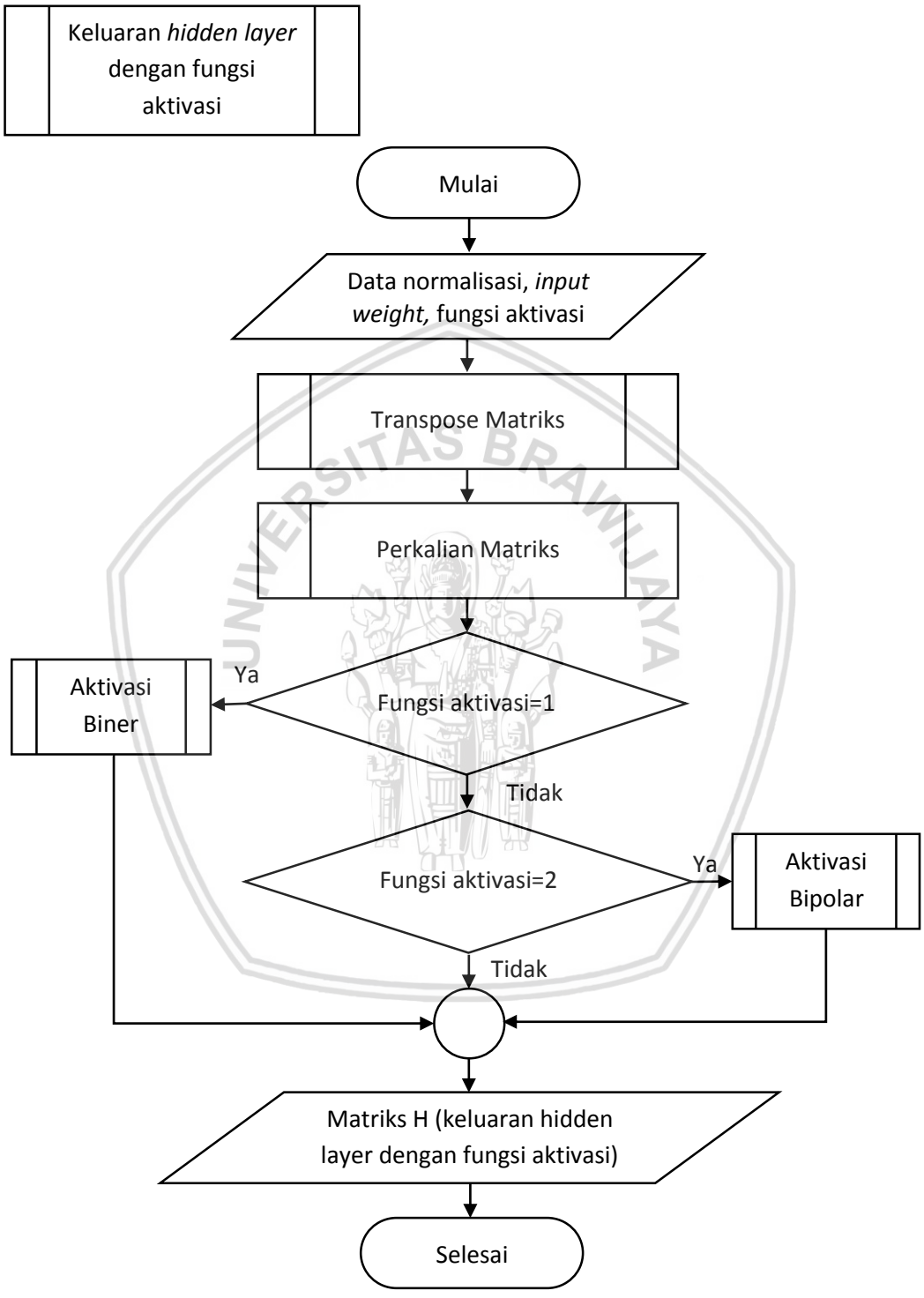


Gambar 4.4 Diagram Alir Proses *Training*

Proses *training* pada Gambar 4.4 memiliki beberapa tahapan sebagai berikut:

1. Masukan yang diterima sistem berupa data *training*, jumlah *hidden layer*, *input weigth* dan fungsi aktivasi
2. Melakukan perhitungan keluaran *hidden layer* menggunakan fungsi aktivasi ($H(x)$) sesuai dengan Persamaan 2.3. Diagram alir untuk menghitung ($H(x)$) dapat dilihat pada Gambar 4.5
3. Melakukan perhitungan matriks *Moore-Penrose Generalized Invers* dari hasil keluaran pada *hidden layer* menggunakan fungsi aktivasi. Gambar 4.10 merupakan diagram alir untuk menghitung matriks *Moore-Penrose Generalized Invers.*

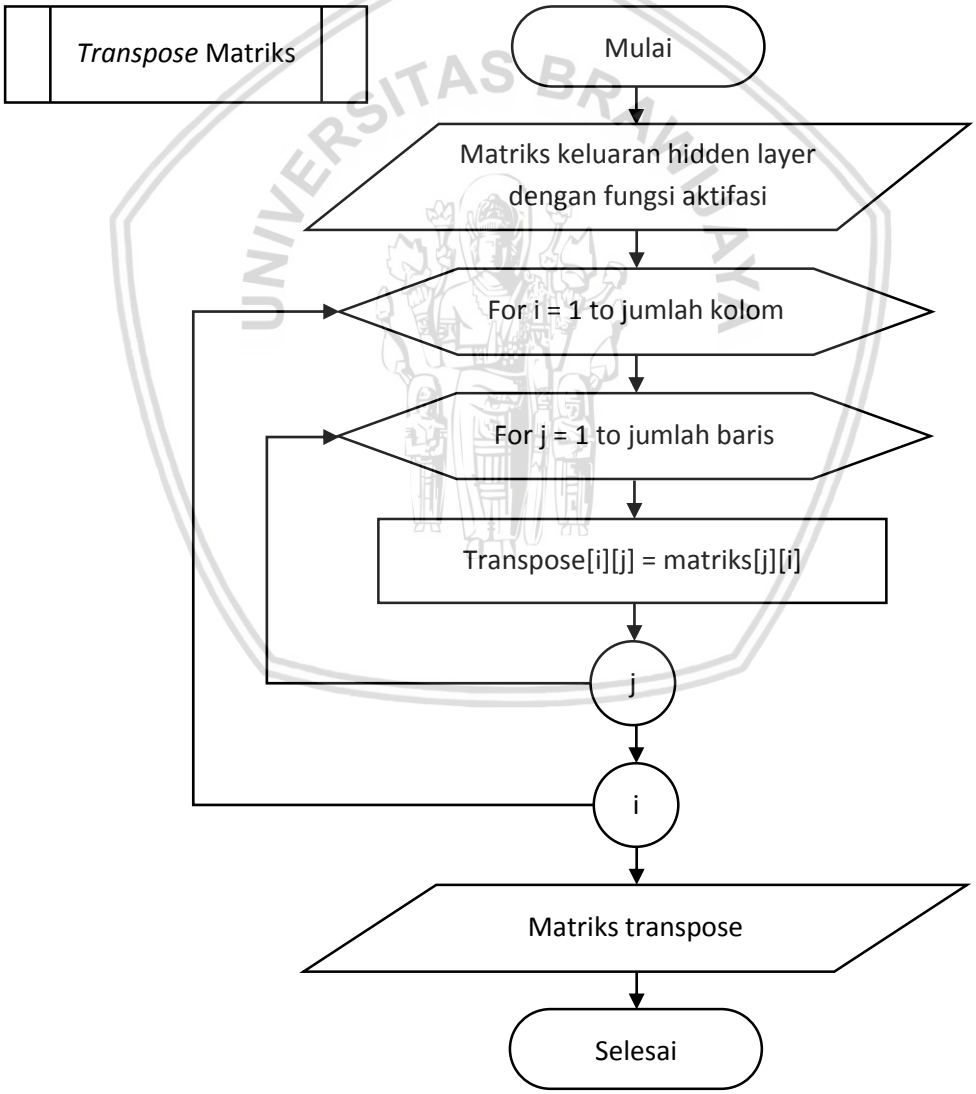
4. Melakukan perhitungan *output weight* sesuai dengan Persamaan 2.4. Diagram alir untuk menghitung *output weight* dapat dilihat pada Gambar 4.13
5. Hasil keluaran sistem berupa matriks *output weight*



Gambar 4.5 Diagram Alir keluaran *hidden layer* dengan fungsi aktivasi

Berdasarkan Gambar 4.5, langkah-langkah dalam menghitung keluaran *Hidden Layer* dengan fungsi aktivasi diuraikan sebagai berikut:

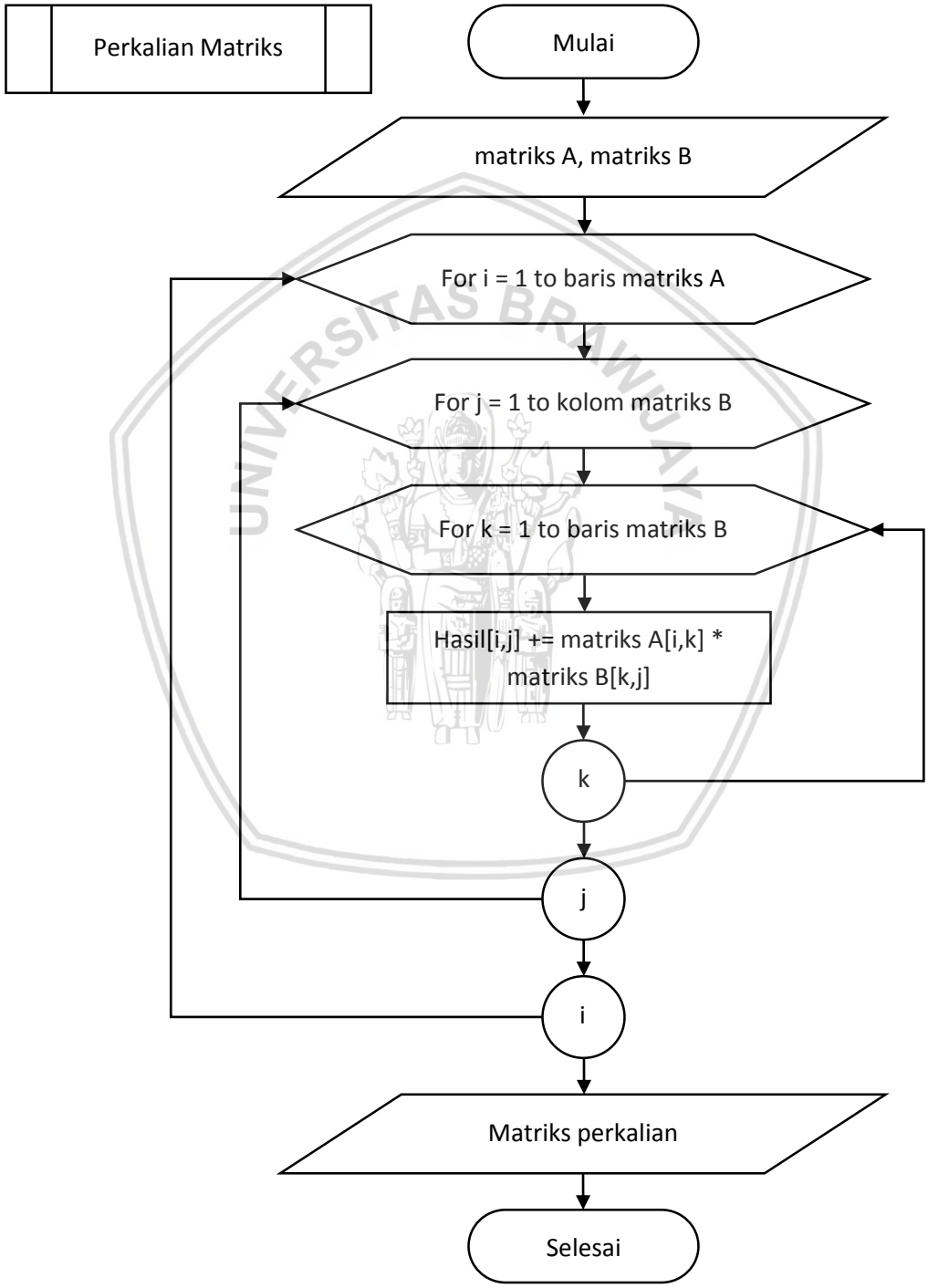
1. Masukan yang diterima sistem berupa data yang telah dinormalisasi berdasarkan jumlah data yang telah dibagi pada proses *training* dan proses *testing*
2. *Transpose* matriks dengan mengubah baris menjadi kolom begitupun sebaliknya kolom menjadi baris Diagram alir untuk *transpose* matriks dapat dilihat pada Gambar 4.6
3. Melakukan perkalian matriks dari hasil *transpose* dengan keluaran *hidden layer*
4. Melakukan perhitungan fungsi aktivasi sigmoid biner menggunakan rumus pada Persamaan 2.1
5. Melakukan perhitungan fungsi aktivasi sigmoid bipolar menggunakan rumus pada Persamaan 2.2
6. Keluaran pada sistem berupa matriks keluaran *hidden layer* yang telah diaktivasi ($H(x)$)



Gambar 4.6 Diagram Alir *Transpose* Matriks

Berdasarkan Gambar 4.6, langkah-langkah dalam menghitung *transpose* matriks diuraikan sebagai berikut:

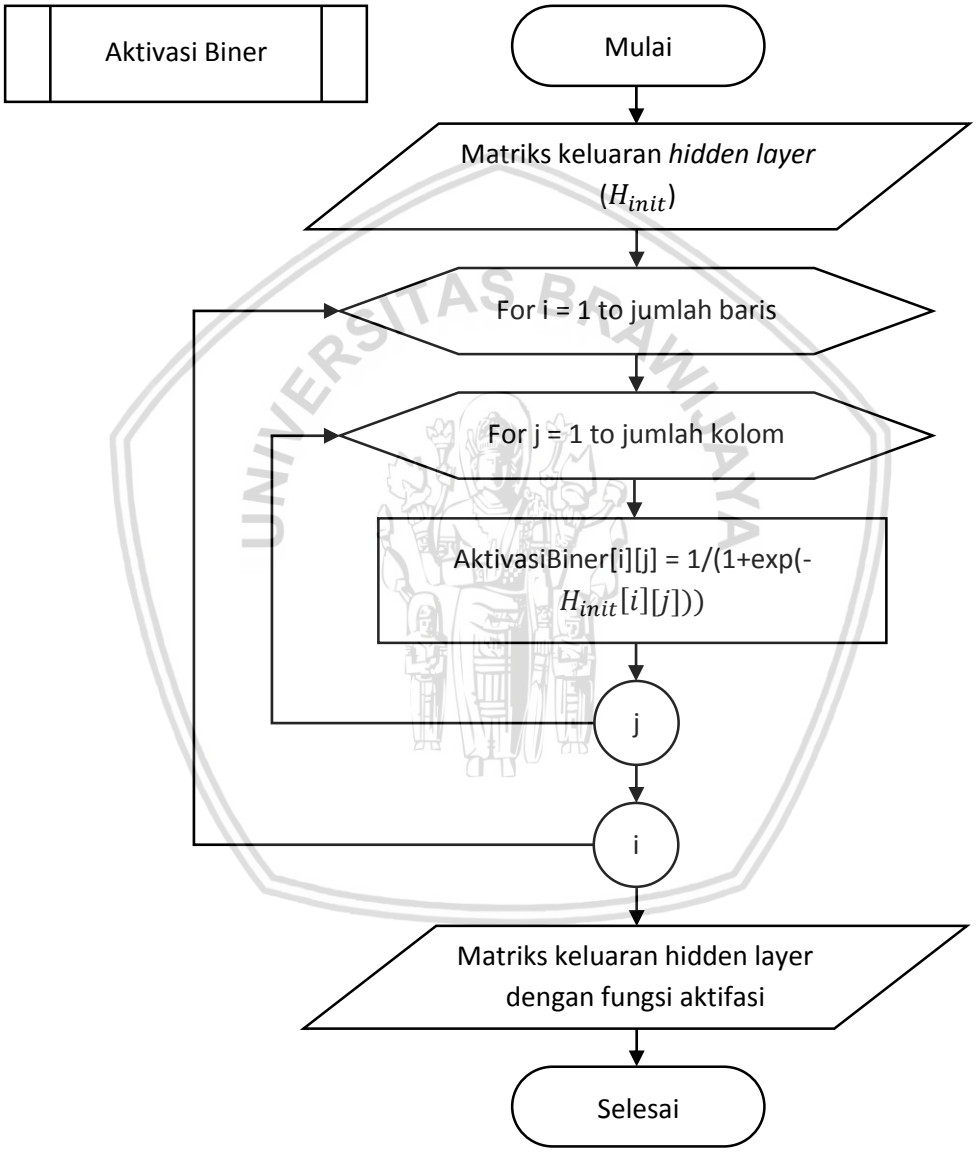
1. Masukan yang diterima sistem berupa matriks keluaran *hidden layer* dengan fungsi aktivasi ($H(x)$)
2. Melakukan *transpose* matriks dengan mengubah baris menjadi kolom dan sebaliknya kolom menjadi baris
3. Keluaran sistem berupa matriks hasil perolehan dari *transpose*



Gambar 4.7 Diagram Alir Perkalian Matriks

Berdasarkan Gambar 4.7, langkah-langkah dalam menghitung perkalian matriks diuraikan sebagai berikut:

1. Masukan yang diterima sistem berupa 2 matriks, misal: matriks A dan matriks B
2. Melakukan perkalian kedua matriks dengan cara mengalikan bilangan-bilangan yang terdapat pada kolom matriks A dengan bilangan-bilangan yang terdapat pada baris matriks B
3. Keluaran sistem berupa matriks hasil perkalian

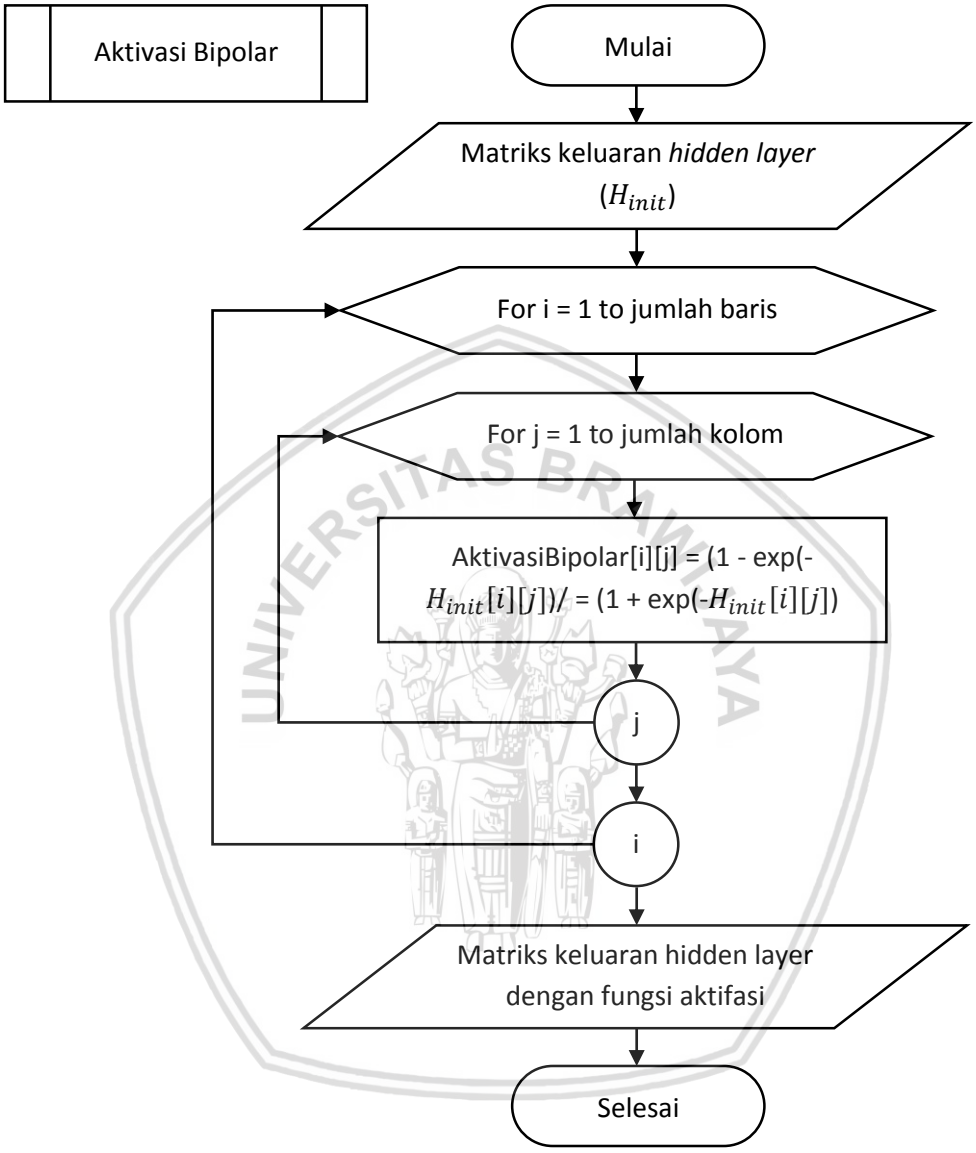


Gambar 4.8 Diagram Alir Fungsi Aktivasi Sigmoid Biner

Berdasarkan Gambar 4.8, langkah-langkah dalam menghitung matriks keluaran *Hidden* Layer menggunakan fungsi aktivasi sigmoid biner diuraikan sebagai berikut:

1. Masukan yang diterima sistem berupa matriks keluaran *hidden layer* (H_{init})

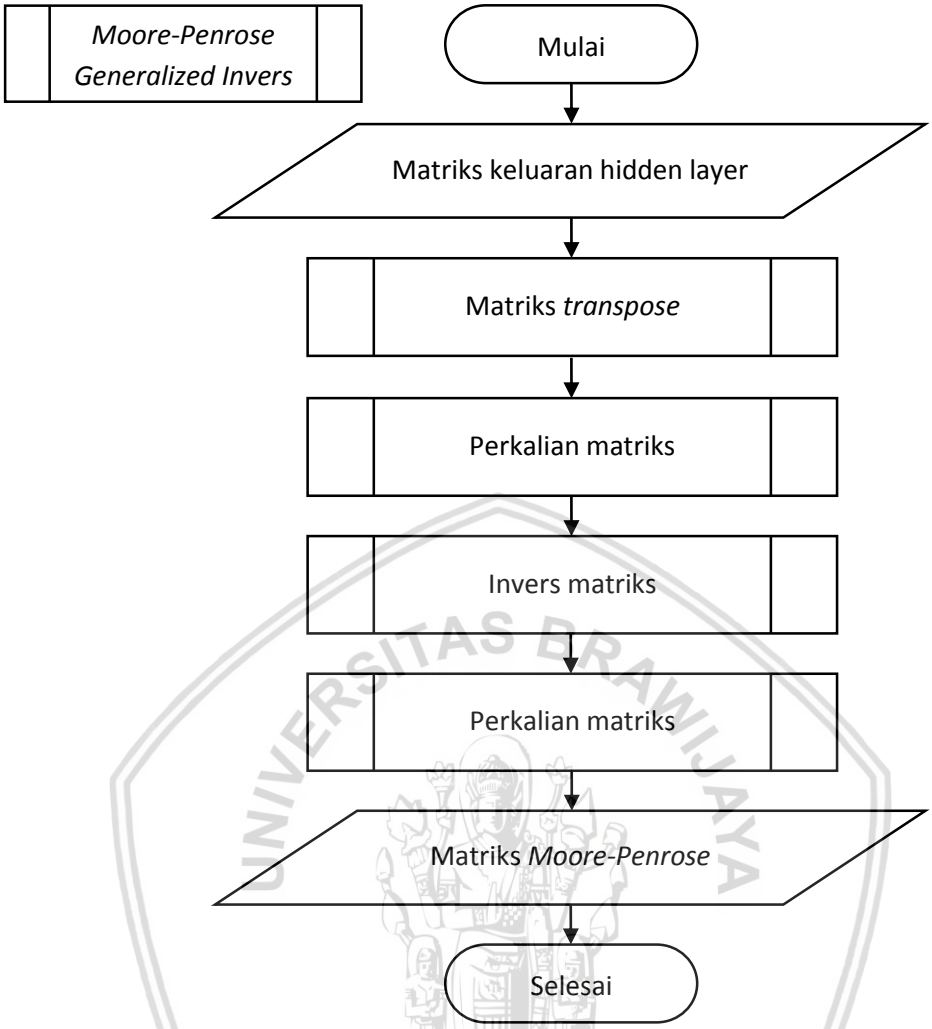
2. Melakukan perhitungan fungsi aktivasi sigmoid biner menggunakan rumus pada Persamaan 2.1
4. Keluaran sistem berupa matriks keluaran *hidden layer* yang telah diaktivasi ($H(x)$)



Gambar 4.9 Diagram Alir Fungsi Aktivasi Sigmoid Bipolar

Berdasarkan Gambar 4.9, langkah-langkah dalam menghitung matriks keluaran *Hidden* Layer menggunakan fungsi aktivasi sigmoid bipolar diuraikan sebagai berikut:

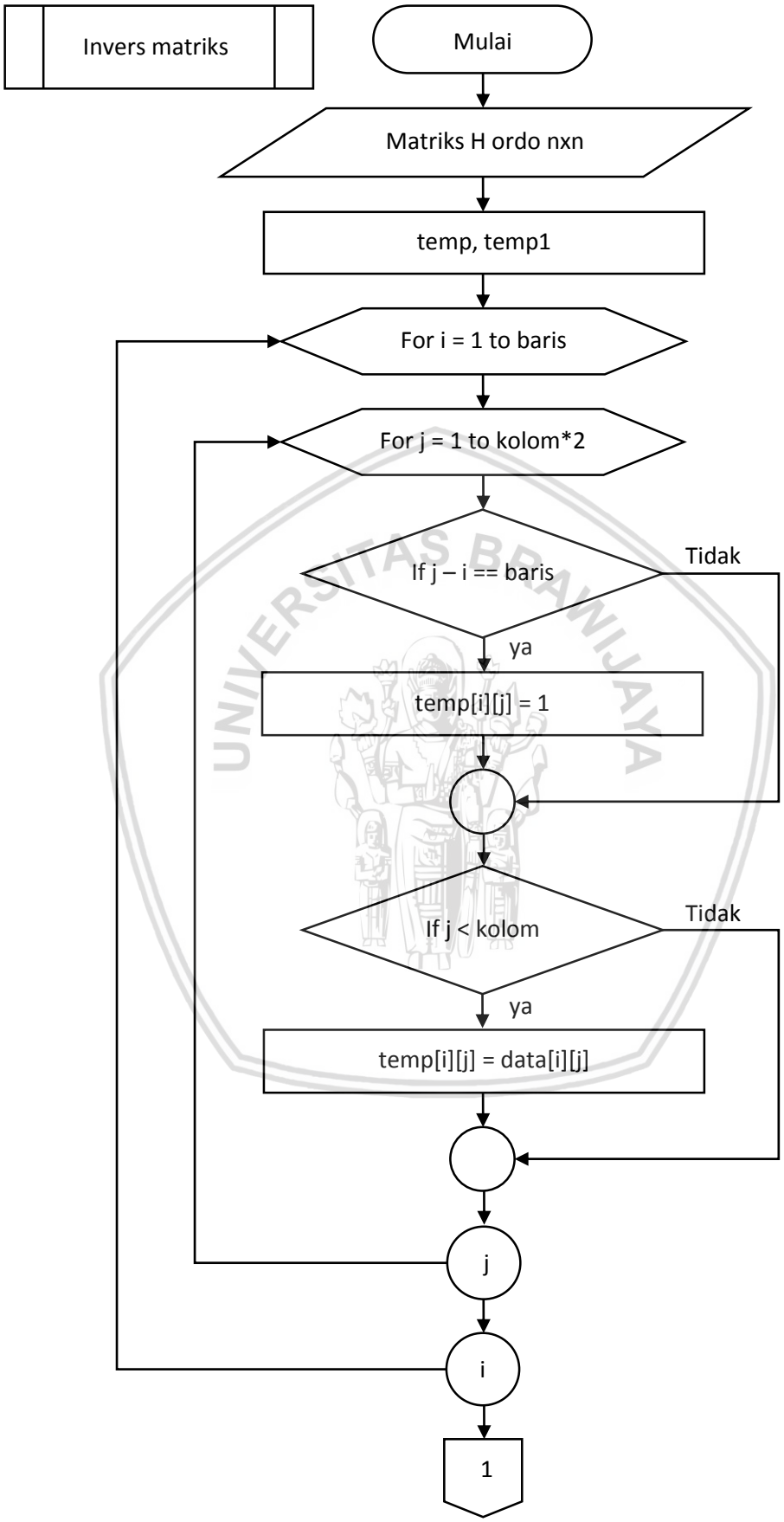
1. Masukan yang diterima sistem berupa matriks keluaran *hidden layer* (H_{init})
2. Melakukan perhitungan fungsi aktivasi sigmoid bipolar menggunakan rumus pada Persamaan 2.2
3. Keluaran sistem berupa matriks keluaran *hidden layer* yang telah diaktivasi ($H(x)$

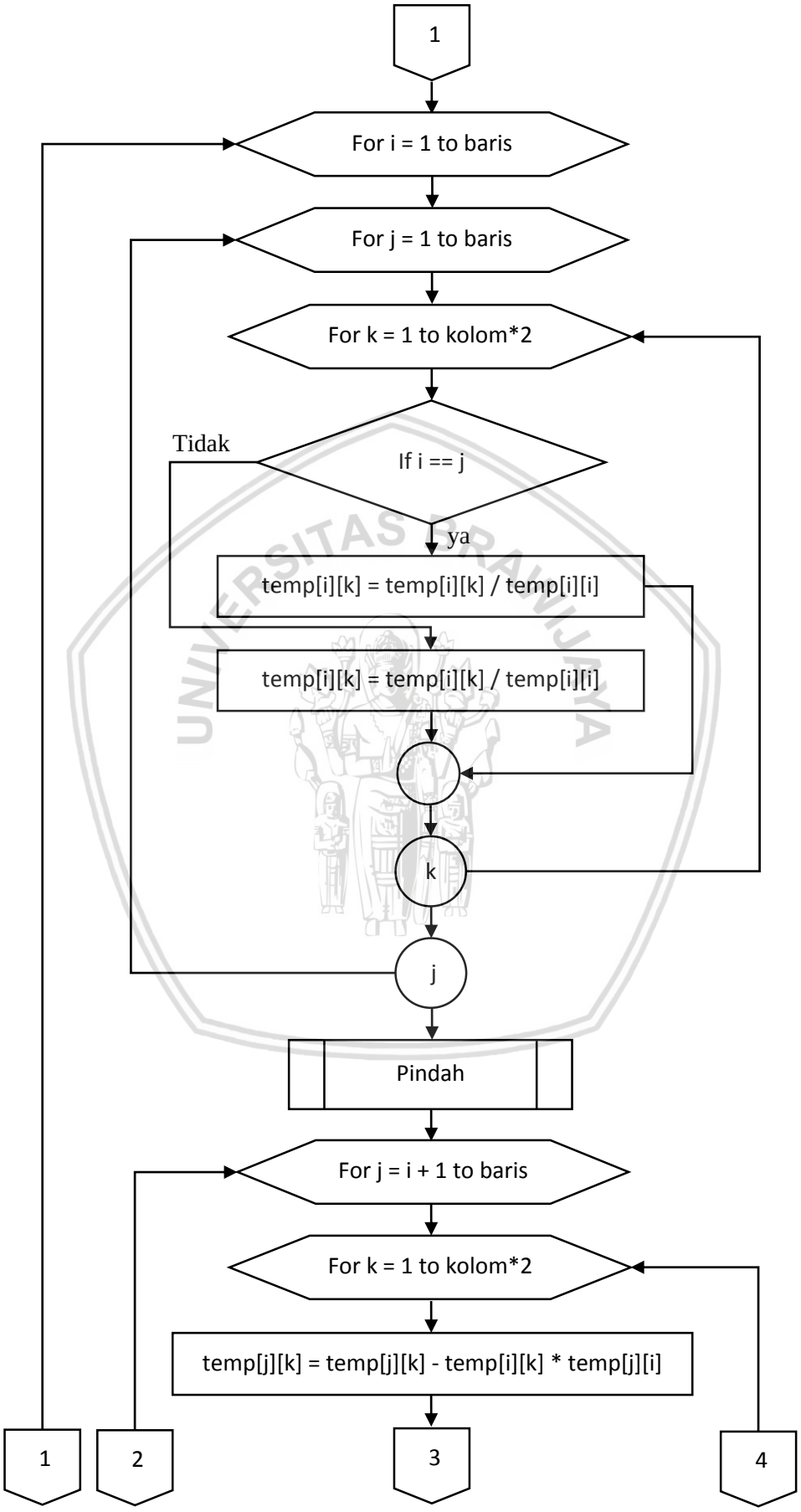


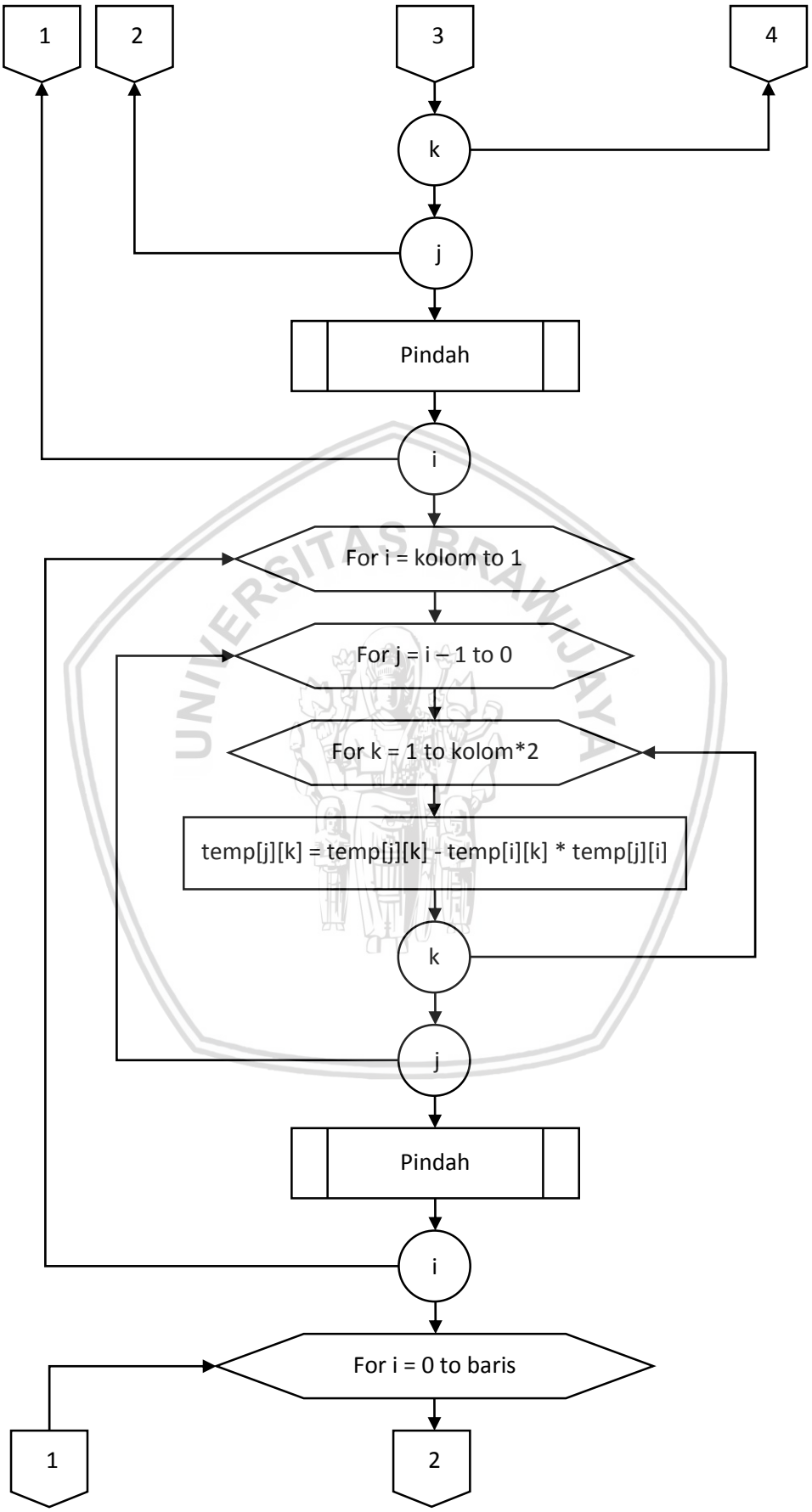
Gambar 4.10 Diagram Alir *Moore-Penrose Generalized Invers*

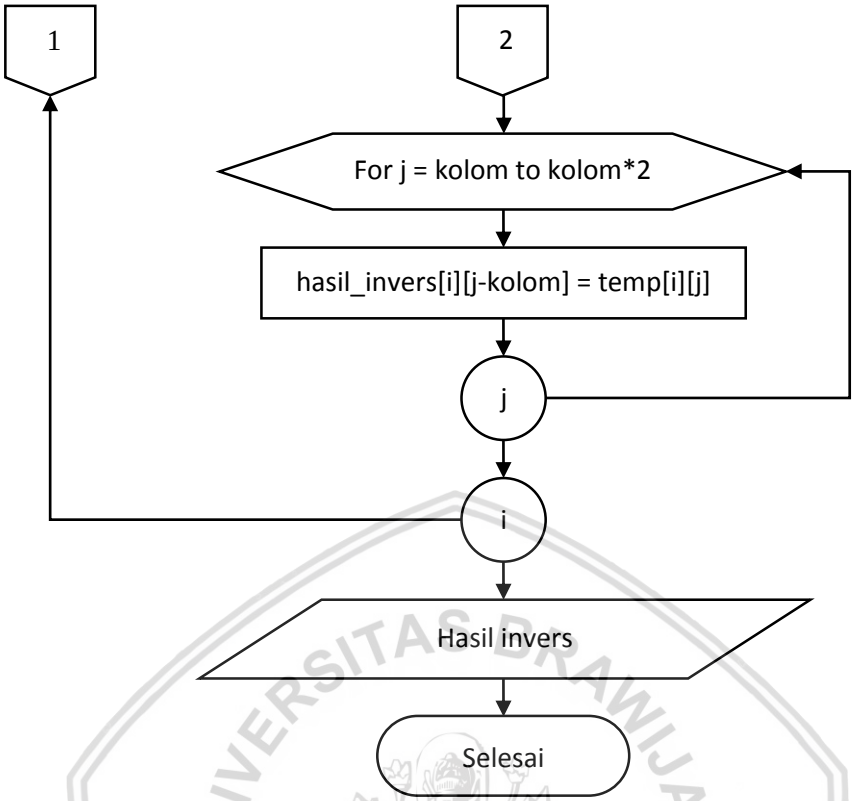
Proses menghitung matriks *Moore-Penrose Generalized Invers* pada Gambar 4.10 memiliki beberapa tahapan sebagai berikut:

1. Masukan yang diterima sistem berupa matriks keluaran *hidden layer*
2. Melakukan *transpose* matriks pada keluaran *hidden layer*
3. Melakukan perkalian matriks antara hasil *transpose* dan keluaran *hidden layer*
4. Melakukan *invers* dari hasil perkalian matriks tersebut menggunakan operasi baris elementer (OBE)
5. Melakukan perkalian matriks dari hasil *invers* dan hasil *transpose* keluaran *hidden layer*
6. Keluaran sistem berupa matriks *Moore-Penrose Generalized Invers* (H^+)


Invers matriks
Mulai
Matriks H ordo nxn
temp, temp1
For i = 1 to baris
For j = 1 to kolom*2
If j – i == baris
Tidak
ya
temp[i][j] = 1
If j < kolom
Tidak
ya
temp[i][j] = data[i][j]
j
i
1


1
For i = 1 to baris
For j = 1 to baris
For k = 1 to kolom*2
Tidak
If i == j
ya
temp[i][k] = temp[i][k] / temp[i][i]
temp[i][k] = temp[i][k] / temp[i][i]
k
j
Pindah
For j = i + 1 to baris
For k = 1 to kolom*2
temp[j][k] = temp[j][k] - temp[i][k] * temp[j][i]
1
2
3
4

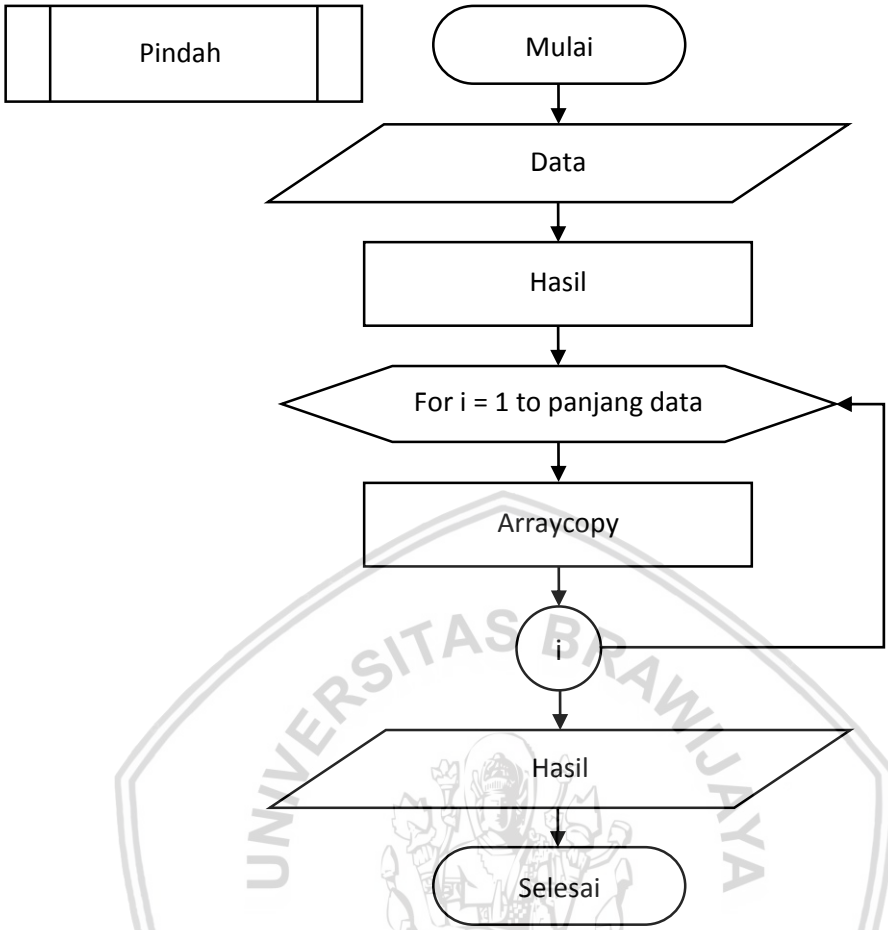

1
2
3
4
k
j
Pindah
i
For i = kolom to 1
For j = i – 1 to 0
For k = 1 to kolom*2
temp[j][k] = temp[j][k] - temp[i][k] * temp[j][i]
k
j
Pindah
i
For i = 0 to baris
1
2



Gambar 4.11 Diagram Alir *Invers* Matriks Perkalian

Berdasarkan Gambar 4.11, langkah-langkah dalam menghitung invers dari perkalian matriks diuraikan sebagai berikut:

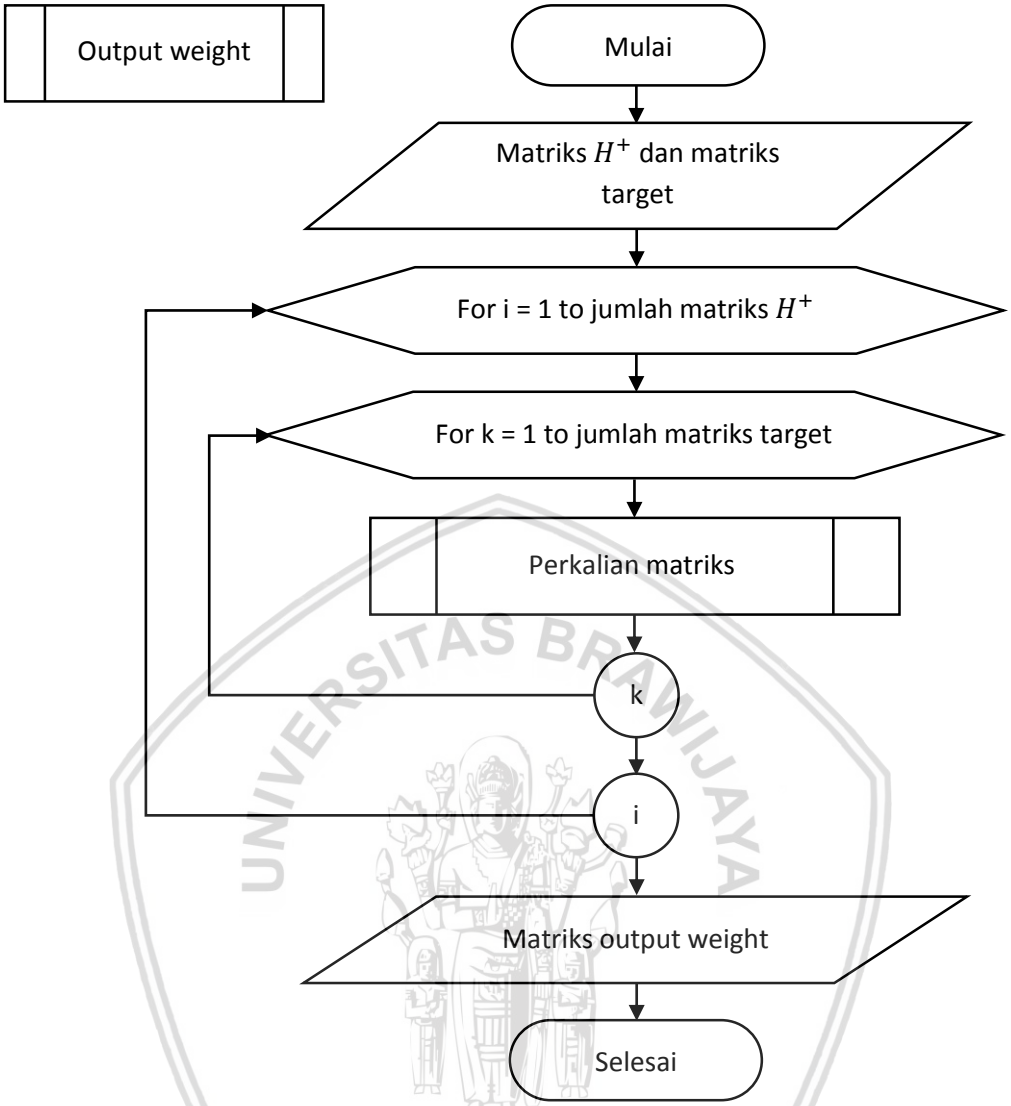
1. Masukan yang diterima sistem berupa matriks hasil perkalian antara matriks *transpose* keluaran *hidden layer* dan matriks keluaran *hidden layer* dengan ordo nxn
2. Inisialisasi matriks identitas yang digunakan dalam Operasi Baris Elementer (OBE) dengan ordo yang sama dengan matriks hasil perkalian
3. Melakukan proses mengubah baris dibawah diagonal menjadi nilai 0
4. Melakukan proses mengubah baris diatas diagonal menjadi nilai 0
5. Melakukan proses *invers* matriks dengan mengambil matriks sesuai ordo awal, yang kemudian menjadi matriks baru
6. Keluaran sistem berupa matriks hasil *invers*



Gambar 4.12 Diagram Alir Pindah

Berdasarkan Gambar 4.12, langkah-langkah dalam memindahkan matriks invers diuraikan sebagai berikut:

1. Masukan yang diterima sistem berupa data yang akan dipindahkan
2. Melakukan perulangan sebanyak panjang data
3. Melakukan arraycopy yang digunakan untuk menyalin semua elemen di dalam array ke dalam array yang baru
4. Keluaran sistem berupa matriks hasil yang telah dipindahkan



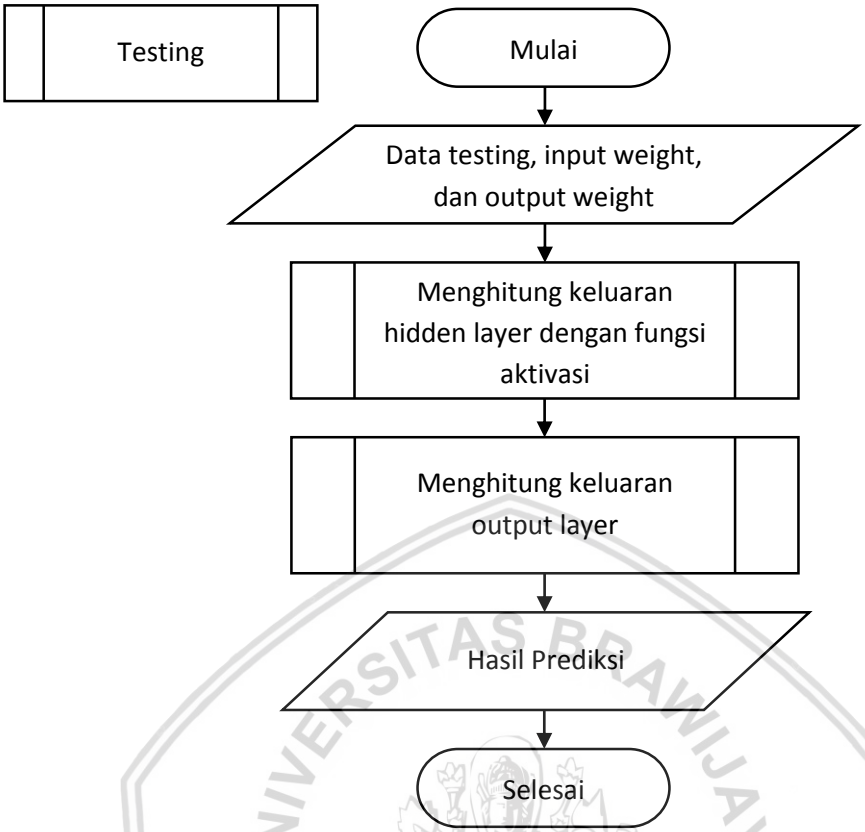
Gambar 4.13 Diagram Alir *Output Weight*

Berdasarkan Gambar 4.13, langkah-langkah dalam menghitung *output weight* diuraikan sebagai berikut:

1. Masukan yang diterima sistem berupa matriks *Moore-Penrose Generalized Invers* (H^+) dengan matriks target
2. Melakukan perkalian matriks dari hasil matriks H^+ dan hasil matriks target menggunakan rumus pada Persamaan 2.4
3. Keluaran sistem berupa matriks *output weight*

4.2.4 Proses *Testing*

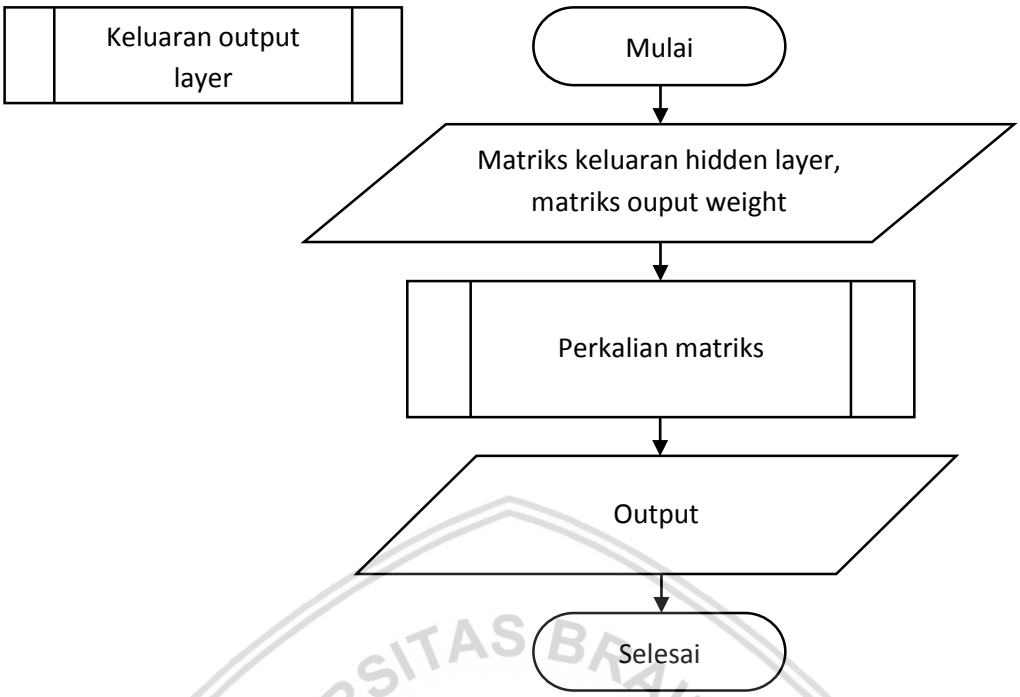
Proses *Testing* merupakan proses uji coba yang diperoleh dari proses *training* berupa nilai *input weight,* dan *output weight*. Proses *testing* dilakukan sama seperti langkah-langkah pada proses training. Gambar 4.14 merupakan diagram alir proses *testing* menggunakan metode ELM.



Gambar 4.14 Diagram Alir Proses *Testing*

Proses *testing* pada Gambar 4.14 memiliki beberapa tahapan sebagai berikut:

1. Masukan yang diterima sistem berupa data *testing.* Nilai *input weight* dan *output weight* yang digunakan dari proses *training*
2. Melakukan perhitungan keluaran *hidden layer* dengan fungsi aktivasi menggunakan rumus pada Persamaan 2.3. Diagram alir untuk menghitung keluaran *hidden layer* dapat dilihat pada Gambar 4.5
3. Melakukan perhitungan keluaran *output layer* menggunakan rumus pada Persamaan 2.5. Diagram alir untuk menghitung keluaran *output layer* dapat dilihat pada Gambar 4.15
4. Melakukan perhitungan nilai *error* (MSE) menggunakan rumus pada Persamaan 2.6. Diagram alir untuk menghitung nilai MSE dapat dilihat pada Gambar 4.16
5. Hasil keluaran sistem berupa hasil prediksi dan nilai *error*



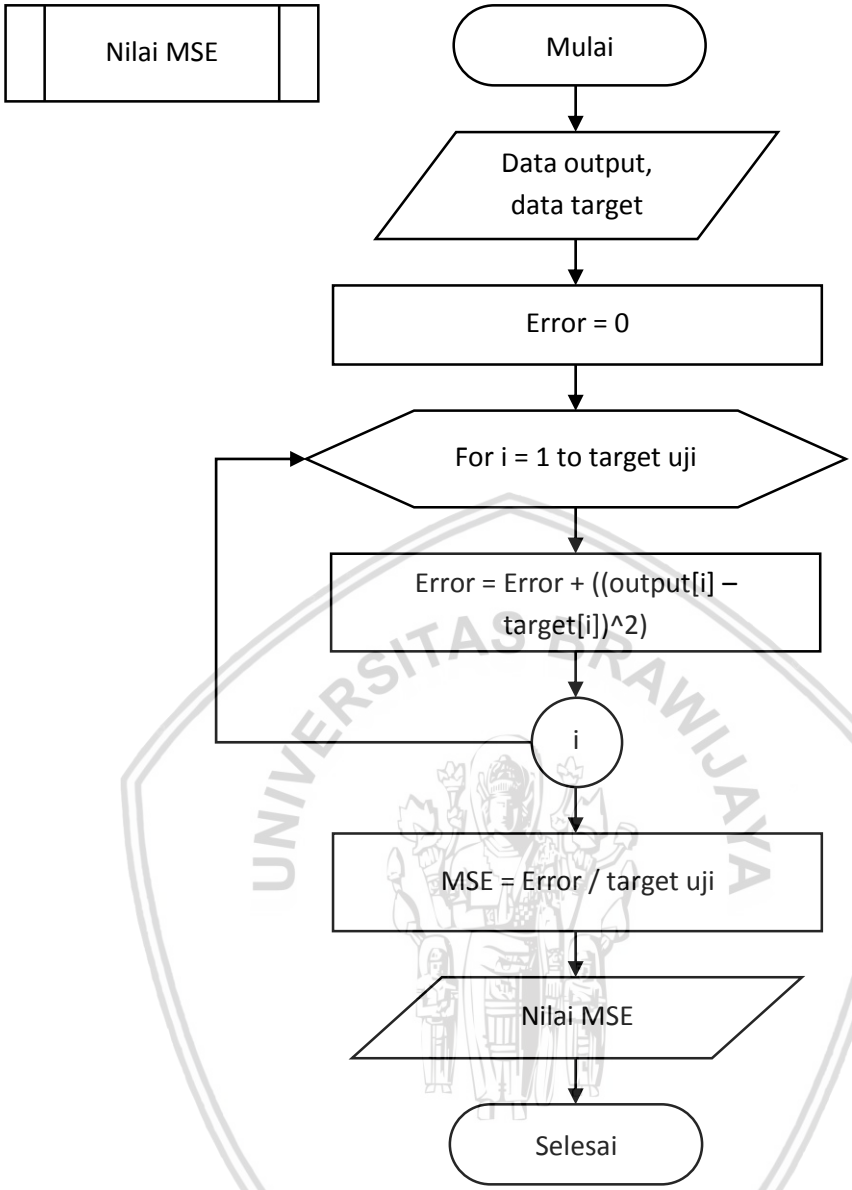
Gambar 4.15 Diagram Alir Proses Menghitung Keluaran *Output Layer*

Proses menghitung keluaran *output layer* pada Gambar 4.15 memiliki beberapa tahapan sebagai berikut:

1. Masukan yang diterima sistem berupa matriks keluaran *hidden layer* dan matriks *output weight* yang diperoleh dari proses *training*
2. Melakukan perhitungan keluaran *output layer* berupa perkalian matriks *hidden layer* dengan matriks *output layer* sesuai rumus pada Persamaan 2.5
3. Hasil keluaran sistem berupa keluaran pada *output layer* (*y*). Keluaran tersebut merupakan *output* hasil proses *testing*

4.2.5 Proses Perhitungan Nilai *Mean Square Error* (MSE)

Perhitungan tingkat *error* menggunakan *Mean Square Error* (MSE) merupakan proses untuk mengevaluasi metode ELM dengan mengetahui tingkat *error* yang dihasilkan. Gambar 4.16 menunjukkan diagram alir proses perhitungan nilai *Mean Square Error* (MSE).



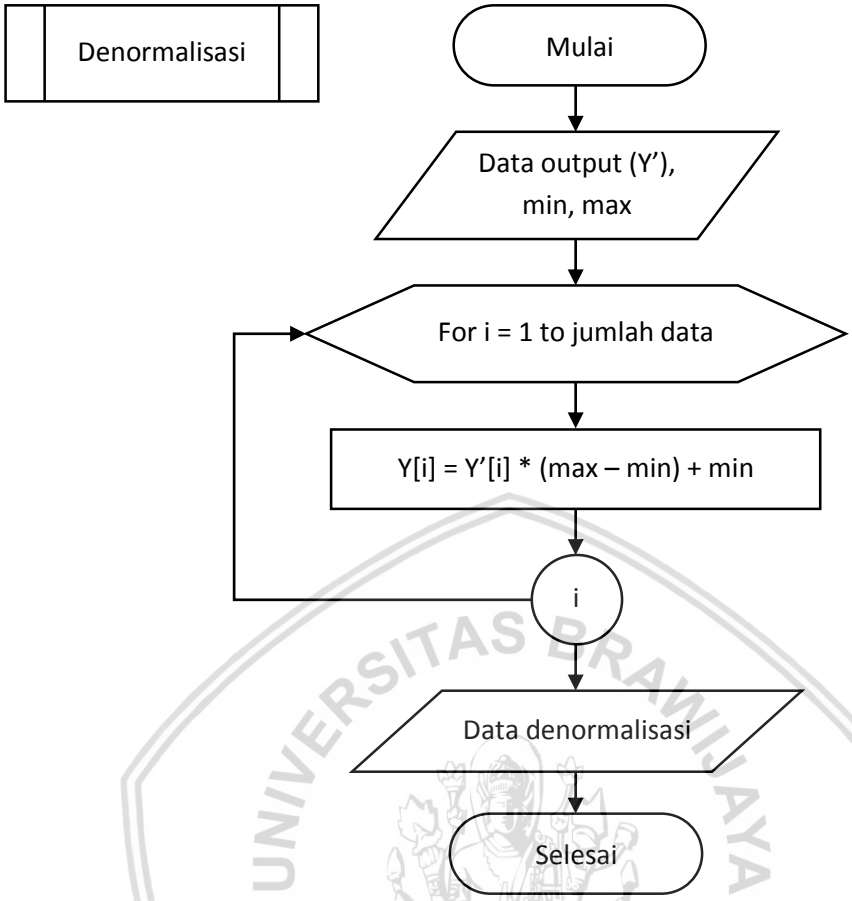
Gambar 4.16 Diagram Alir Proses Menghitung Tingkat *Error* Menggunakan MSE

Proses menghitung tingkat *error* menggunakan MSE pada Gambar 4.16 memiliki beberapa tahapan sebagai berikut:

1. Masukan yang diterima sistem berupa data *output* dan data target
2. Melakukan perhitungan hasil akhir dari nilai MSE menggunakan rumus pada Persamaan 2.8
3. Hasil keluaran sistem yaitu tingkat *error* dari perhitungan ELM berupa nilai MSE.

4.2.6 Proses Denormalisasi

Denormalisasi merupakan proses untuk memperoleh nilai sebenarnya dari keluaran hasil prediksi. Gambar 4.17 menunjukkan diagram alir proses denormalisasi data.



Gambar 4.17 Diagram Alir Proses Denormalisasi

Berdasarkan Gambar 4.17, langkah-langkah proses denormalisasi diuraikan sebagai berikut:

1. Masukan pada sistem berupa matriks keluaran pada *output layer*, nilai maksimun (max), dan nilai minimun (min)
2. Melakukan perhitungan denormalisasi menggunakan rumus pada Persamaan 2.7
3. Hasil keluaran sistem berupa data yang telah didenormalisasi yang merupakan data asli hasil prediksi

4.3 Perhitungan Manual

Perhitungan manual pada penelitian ini menggunakan 10 sampel data dengan pembagian data *training* sebanyak 80% dan data *testing* sebanyak 20%. Sehingga diperoleh 8 data *training* dan 2 data *testing*. Jumlah *neuron* pada *hidden layer* yang digunakan dalam perhitungan manual ini sebanyak 2 dengan fungsi aktivasi *sigmoid* serta jumlah fitur sebanyak 3, sehingga diperoleh X1 untuk data 3 bulan sebelum, X2 untuk data 2 bulan sebelum, dan X3 untuk data 1 bulan sebelum target. Berikut adalah data yang digunakan dapat dilihat pada Tabel 4.2.

Tabel 4.2 Data *Training* dan Data *Testing*

Data ke-	Data Input			Target
	X1	X2	X3	
1	60	71	71	69
2	71	71	69	68
3	71	69	68	44
4	69	68	44	57
5	68	44	57	62
6	44	57	62	59
7	57	62	59	67
8	62	59	67	55
9	59	67	55	54
10	67	55	54	58

4.3.1 Perhitungan Normalisasi Data

Data sekunder pada Tabel 4.2 dinormalisasi terlebih dahulu menggunakan *Min-Max Normalization* untuk keperluan pengolahan data. Berikut adalah langkah-langkah manualisasi data yang dihitung berdasarkan Persamaan 2.6.

Langkah 1: menentukan nilai maksimal dan minimal dari keseluruhan data yang ditunjukkan pada Tabel 4.3.

Tabel 4.3 Nilai Maksimal dan Minimal

Max	71
Min	44

Langkah 2: melakukan perhitungan normalisasi data dengan menggunakan metode *Min-Max Normalization.* Berikut merupakan contoh perhitungan nilai normalisasi data.

$$x'_{1,1} = \frac{x_{1,1} - \min}{\max - \min} = \frac{60 - 44}{71 - 44} = 0{,}592593$$

Data yang telah dinormalisasi dimasukan sebagai data inputan yang terdiri dari X1, X2, dan X3 dengan membentuk suatu pola data. Berikut adalah pola hasil perhitungan normalisasi data secara keseluruhan dapat dilihat pada Tabel 4.4.

Tabel 4.4 Normalisasi Data

Data ke-*i*	Data Input			Target
	X1	X2	X3	
1	0,592593	1	1	0,925926
2	1	1	0,925926	0,888889
3	1	0,925926	0,888889	0
4	0,925926	0,888889	0	0,481481
5	0,888889	0	0,481481	0,666667
6	0	0,481481	0,666667	0,555556

Tabel 4.4 Normalisasi Data (Lanjutan)

Data ke-*i*	Data Input			Target
	X1	X2	X3	
7	0,481481	0,666667	0,555556	0,851852
8	0,666667	0,555556	0,851852	0,407407
9	0,555556	0,851852	0,407407	0,37037
10	0,851852	0,407407	0,37037	0,518519

4.3.2 Inisialisasi *Input Weight*

Langkah yang harus dilakukan setelah dilakukan *input* data dan normalisasi yaitu inisialisasi *input weight* yang ditentukan secara acak (*random*) dengan range [-1,1]. Inisialisasi *input weight* ditunjukkan pada Tabel 4.5. Sesuai dengan apa yang telah dijelaskan sebelumnya, penelitian ini menggunakan 2 *neuron pada hidden layer* dan 3 *node* pada *input layer*, sehingga terbentuk *input weight* dengan ordo 2x3 sebanyak 6 nilai.

Tabel 4.5 Inisialisasi Input *Weight*

Input weight ($w_{j,k}$)	
$w_{1,1}$	-0,63
$w_{1,2}$	0,17
$w_{1,3}$	0,25
$w_{2,1}$	0,74
$w_{2,2}$	-0,28
$w_{2,3}$	0,88

4.3.3 Perhitungan Proses *Training*

Perhitungan proses *training* dilakukan guna memperoleh bobot terbaik untuk digunakan pada proses *testing*. Terdapat 8 data yang digunakan pada proses *training* dengan 2 *neuron* pada *hidden layer* dan fungsi aktivasi *sigmoid.* Perhitungan proses *training* menggunakan metode ELM memiliki langkah-langkah sebagai berikut.

Langkah 1: melakukan perhitungan nilai keluaran pada *hidden layer* menggunakan fungsi aktivasi. Keluaran pada *hidden layer* dapat dihitung dari perkalian antara data hasil normalisasi dan *input weight* yang didapat secara *random* berdasarkan rumus pada Persamaan 2.3, kemudian dilakukan perhitungan fungsi aktivasi menggunakan fungsi aktivasi *sigmoid.* Hasil dari keluaran *hidden neuron* mempunyai ordo 12x2. Contoh perhitungan keluaran pada *hidden layer* adalah sebagai berikut.

$$H_{init\ 1,1} = X.w^T$$

$$H_{init\ 1,1} = \big((-0{,}63 * 0{,}592593) + (\ 0{,}17 * 1) + (\ 0{,}25 * 1)\big)$$
$$= 0{,}0467$$

Berikut adalah hasil dari keluaran *hidden layer* secara keseluruhan dapat dilihat pada Tabel 4.6.

Tabel 4.6 Matriks Keluaran *Hidden Layer*

H_{init}	1	2
1	0,0467	1,0385
2	-0,229	1,2748
3	-0,25	1,263
4	-0,432	0,4363
5	-0,44	1,0815
6	0,2485	0,4519
7	-0,051	0,6585
8	-0,113	1,0874

Contoh perhitungan keluaran *hidden layer* menggunakan fungsi aktivasi adalah sebagai berikut.

$$H(x)_{1,1} = \frac{1}{1+e^{-x}} = \frac{1}{1+e^{-0{,}0467}} = 0{,}5117$$

Berikut adalah hasil keluaran *hidden layer* menggunakan fungsi aktivasi *sigmoid* secara keseluruhan dapat dilihat pada Tabel 4.6.

Tabel 4.7 Matriks Keluaran *Hidden Layer* Menggunakan fungsi aktivasi

$H(x)$	1	2
1	0,5117	0,7385
2	0,4431	0,7815
3	0,4377	0,7795
4	0,3936	0,6074
5	0,3918	0,7468
6	0,5618	0,6111
7	0,4872	0,6589
8	0,4719	0,7479

Langkah 2: melakukan perhitungan *output weight* dari *hidden layer* ke *output layer.* Hal yang harus dilakukan adalah menghitung matriks *Moore-Penrose Generalized Invers* dari hasil keluaran pada *hidden layer* menggunakan fungsi aktivasi. Matriks *Moore-Penrose Generalized Invers* dapat dihitung dengan rumus $H^+ = (H^T H)^{-1} H^T$. Pertama melakukan *transpose* pada matriks $H(x)$. Hasil

transpose mempunyai ordo 2x8 yang ditunjukkan pada Tabel 4.8 (nilai *transpose* matriks keluaran *hidden layer* disertakan pada lampiran C.1).

Tabel 4.8 *Transpose* Matriks Keluaran *Hidden Layer* dengan Fungsi Aktivasi

H^T	1	2	3	...	8
1	0,5117	0,4431	0,4377	...	0,4719
2	0,7385	0,7815	0,7795	...	0,7479

Setelah melakukan *transpose* pada matriks $H(x)$, dilanjutkan dengan menghitung nilai matriks dari perkalian hasil *transpose* dengan keluaran *hidden layer.* Pada Tabel 4.9 merupakan hasil perkalian matriks secara keseluruhan yang mempunyai ordo 2x2. Contoh perhitungan perkalian matriks adalah sebagai berikut.

$$(H^T H)_{1,1} = (0{,}5117 * 0{,}5117) + (0{,}4431 * 0{,}4431) + (0{,}4377 * 0{,}4377) + .. + (0{,}7479 * 0{,}7479)$$
$$= 1{,}7339$$

Tabel 4.9 Perkalian Hasil *Transpose* dengan Keluaran *Hidden Layer* Menggunakan Fungsi Aktivasi

$H^T H$	1	2
1	1,7339	2,6143
2	2,6144	4,0573

Selanjutnya menentukan *invers* matriks $H^T H$. Operasi Baris Elementari (OBE) adalah metode yang akan digunakan dalam perhitungan *invers matriks.* Perhitungan *invers matriks* dijabarkan sebagai berikut.

1. Bentuk matriks I sebagai *Identitas,* $R1$ sebagai Baris 1, dan $R2$ sebagai Baris 2
   $$[H|I] = \begin{bmatrix} 1{,}7339 & 2{,}6144 \\ 2{,}6144 & 4{,}0573 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$
2. $R1 : 1{,}7339 \rightarrow R1$
   $$[H|I] = \begin{bmatrix} 1 & 1{,}5077 \\ 2{,}6143 & 4{,}0573 \end{bmatrix} \begin{bmatrix} 0{,}5767 & 0 \\ 0 & 1 \end{bmatrix}$$
3. $R2 - 2{,}6143 * R1 \rightarrow R2$
   $$[H|I] = \begin{bmatrix} 1 & 1{,}5077 \\ 0 & 0{,}1156 \end{bmatrix} \begin{bmatrix} 0{,}5767 & 0 \\ -1{,}5077 & 1 \end{bmatrix}$$
4. $R2 : 0{,}1156 \rightarrow R2$
   $$[H|I] = \begin{bmatrix} 1 & 1{,}5077 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 0{,}5767 & 0 \\ -13{,}0455 & 8{,}6524 \end{bmatrix}$$
5. $R1 - 1{,}5077 * R2 \rightarrow R1$
   $$[H|I] = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 20{,}2454 & -13{,}0455 \\ -13{,}0455 & 8{,}6524 \end{bmatrix}$$

Berikut adalah hasil *invers* matriks secara keseluruhan dapat dilihat pada Tabel 4.10.

Tabel 4.10 *Invers* Matriks

$H^T H$	1	2
1	20,2454	-13,0455
2	-13,0455	8,6524

Setelah mendapatkan *invers* dari matriks $H^T H$, maka hasil persamaan matriks *Moore-Penrose Generalized Invers* (H^+) adalah sebagai berikut.

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

$$H^+{}_{1,1} = (20{,}2454 * 0{,}5117) + (-13{,}0455 * 0{,}7385) = 0{,}7244$$

Berikut adalah hasil matriks *Moore-Penrose Generalized Invers* secara keseluruhan dapat dilihat pada Tabel 4.11 (nilai matriks *Moore-Penrose Generalized Invers* disertakan pada lampiran C.2).

Tabel 4.11 Matriks *Moore-Penrose Generalized Invers*

H^+	1	2	3	...	8
1	0,7244	-1,2242	-1,3067	...	-0,2027
2	-0,2847	0,9815	1,0341	...	0,3149

Langkah 3: melakukan perhitungan *output weight* (bobot antara *hidden layer* dan output layer) dengan perkalian antara invers matriks dan matriks target. Contoh perhitungan *output weight* sebagai berikut.

$$\beta = H^+ T$$

$$\beta_{1,1} = (0{,}7244 * 0{,}925926) + (-1{,}2242 * 0{,}888889) + (-1{,}3067 * 0) + (0{,}0455 * 0{,}481481) + \cdots + (-0{,}2027 * 0{,}407407)$$
$$= 1{,}28694$$

Tabel 4.12 merupakan hasil *output weight* (bobot antara *hidden layer* dan output layer) secara keseluruhan.

Tabel 4.12 Nilai *Output Weight*

β
1,28694
0,00242

4.3.4 Perhitungan Proses *Testing*

Perhitungan proses *testing* dilakukan setelah mendapat *output weight* dari perhitungan proses *training*. Data yang digunakan pada perhitungan proses *testing* sebanyak 2 dengan 2 *neuron* pada *hidden layer* dan fungsi aktivasi *sigmoid.* Langkah-langkah yang harus dilakukan sama seperti proses *training* yaitu sebagai berikut.

Langkah 1: melakukan perhitungan nilai keluaran pada *hidden layer* menggunakan fungsi aktivasi. Keluaran pada *hidden layer* dapat dihitung dari perkalian antara

data hasil normalisasi dan *input weight* yang didapat secara *random* berdasarkan rumus pada Persamaan 2.3, kemudian dilakukan perhitungan fungsi aktivasi menggunakan fungsi aktivasi *sigmoid.* Hasil dari keluaran *hidden neuron* mempunyai ordo 2x2. Contoh perhitungan keluaran pada *hidden layer* adalah sebagai berikut.

$$H_{init\ 1,1} = X.w^T$$

$$H_{init\ 1,1} = \big((-0{,}63 * 0{,}555556) + (\ 0{,}17 * 0{,}851852) + (\ 0{,}25 * 0{,}407407)\big)$$
$$= -0{,}10333$$

Berikut adalah hasil dari keluaran *hidden layer* secara keseluruhan dapat dilihat pada Tabel 4.13.

Tabel 4.13 Matriks Keluaran *Hidden Layer*

H_{init}	1	2
1	-0,10333	0,53111
2	-0,37485	0,84222

Contoh perhitungan keluaran *hidden layer* menggunakan fungsi aktivasi adalah sebagai berikut.

$$H(x)_{1,1} = \frac{1}{1+e^{-x}} = \frac{1}{1+e^{-(-0{,}10333)}} = 0{,}474192$$

Berikut adalah hasil keluaran *hidden layer* menggunakan fungsi aktivasi *sigmoid* secara keseluruhan dapat dilihat pada Tabel 4.14.

Tabel 4.14 Matriks Keluaran *Hidden Layer* Menggunakan Fungsi Aktivasi

$H(x)$	1	2
1	0,474192	0629729
2	0,407388	0,698915

Langkah 2: Menghitung keluaran hasil prediksi (hasil keluaran pada *output layer*) dengan menggunakan Persamaan 2.5. Hasil dari matriks keluaran pada *output layer* mempunyai ordo 2x1. Contoh perhitungan keluaran hasil prediksi adalah sebagai berikut.

$$y = H\beta$$

$$y_1 = (0{,}474192 * 1{,}28694) + (0{,}629729 * 0{,}00242)$$
$$= 0{,}611783$$

Tabel 4.15 adalah hasil hasil keluaran pada *output layer* secara keseluruhan.

Tabel 4.15 Hasil Keluaran pada *Output Layer*

y
0,611783
0,525977

4.3.5 Perhitungan Nilai MSE

Perhitungan nilai *Mean Square Error* (MSE) menggunakan Persamaan 2.8. Perhitungan nilai MSE bertujuan untuk mengevaluasi hasil prediksi dengan mengukur performa dari jaringan, yakni seberapa baik kemampuan jaringan dalam mengenali suatu pola. Berikut adalah hasil perhitungan nilai MSE.

$$MSE = \frac{\sum_{i=1}^{n}(y_i - t_i)^2}{n}$$

$$= \frac{(0{,}611783 - 0{,}37037)^2 - (0{,}525977 - 0{,}518519)^2}{2} = 0{,}0582246$$

4.3.6 Perhitungan Denormalisasi Data

Perhitungan denormalisasi data dilakukan untuk memperoleh data yang sebenarnya dari data yang telah dinormalisasi. Perhitungan denormalisasi data menggunakan Persamaan 2.7 dan dapat dilihat contoh perhitungannya sebagai berikut.

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

$$x_1 = 0{,}611783(71 - 44) + 44$$

$$= 60{,}518$$

Berikut adalah hasil perhitungan denormalisasi data dapat dilihat pada Tabel 4.16.

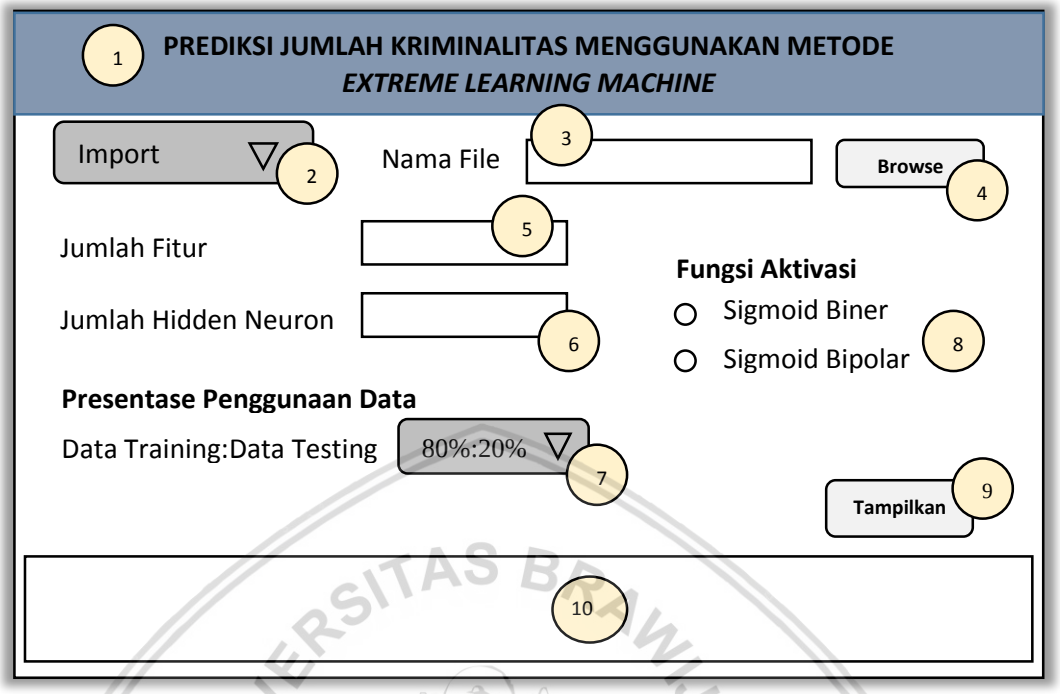
Tabel 4.16 Denormalisasi Data

denormalisasi
60,518
58,201

4.4 Perancangan Antar Muka

Perancangan antar muka sangat dibutuhkan untuk menggambarkan keadaan sebenarnya dari implementasi sistem prediksi jumlah kriminalitas dengan metode ELM yang akan dibangun. Rancangan antar muka dari aplikasi ini terdiri dari 5 halaman yaitu halaman untuk *import* data, normalisasi data, *weight*, *training*, serta *testing* dan evaluasi.

4.4.1 Perancangan Halaman *Import* Data

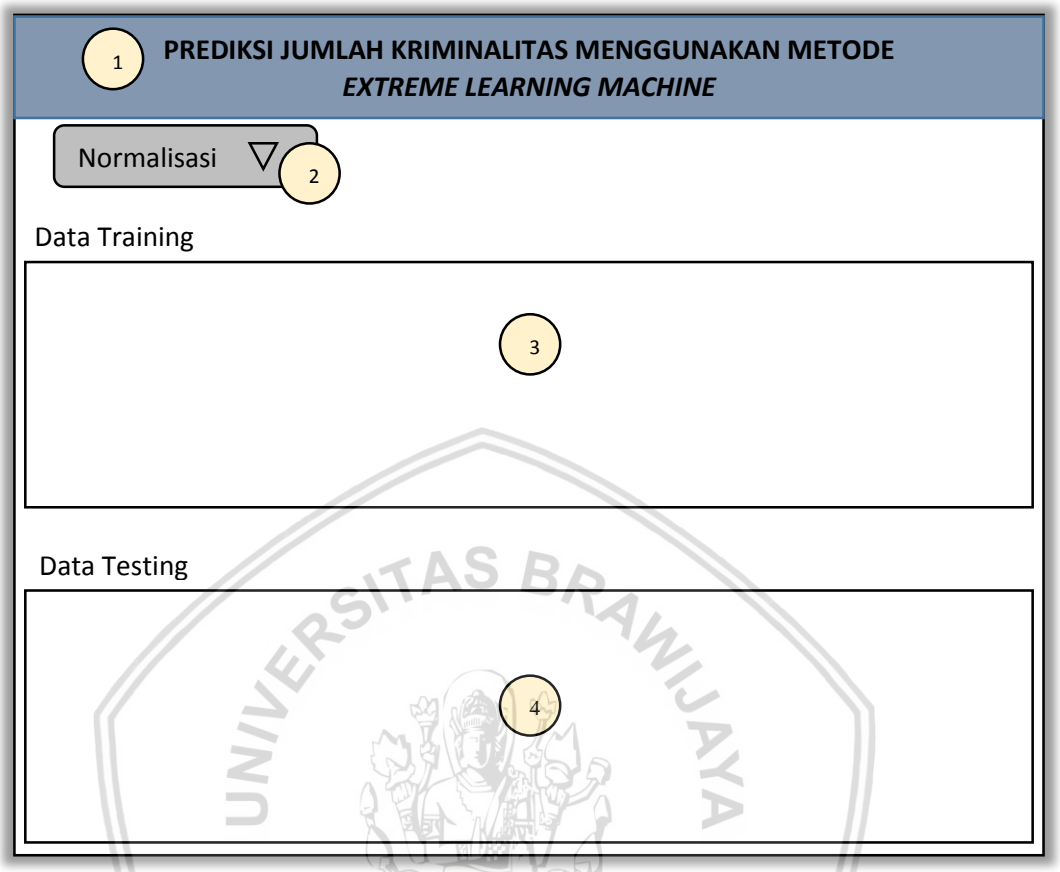


Gambar 4.18 Perancangan Halaman *Import* Data

Keterangan:

1. Header pada aplikasi
2. *Combo box* untuk memilih menu aplikasi. Menu yang dipilih akan tampil dengan warna abu-abu sebagai tanda bahwa menu *import* sedang aktif
3. *Text box* digunakan untuk menampilkan nama file yang dipilih
4. *Button Browse* digunakan untuk memilih data yang akan dilatih dan diuji dalam bentuk .xls
5. *Text box* digunakan untuk menentukan jumlah fitur berupa data masukan
6. *Text box* digunakan untuk menentukan jumlah *hidden neuron*
7. *Combo box* untuk menentukan presentase data *training* dan data *testing*
8. *Radio Button* digunakan untuk memilih salah satu opsi fungsi aktivasi
9. *Button* Tampilan digunakan untuk proses menampilkan data yang telah diimport sebelumnya
10. Data *view* berupa tabel digunakan untuk menampilkan data yang telah diimport sebelumnya

4.4.2 Perancangan Halaman Normalisasi Data

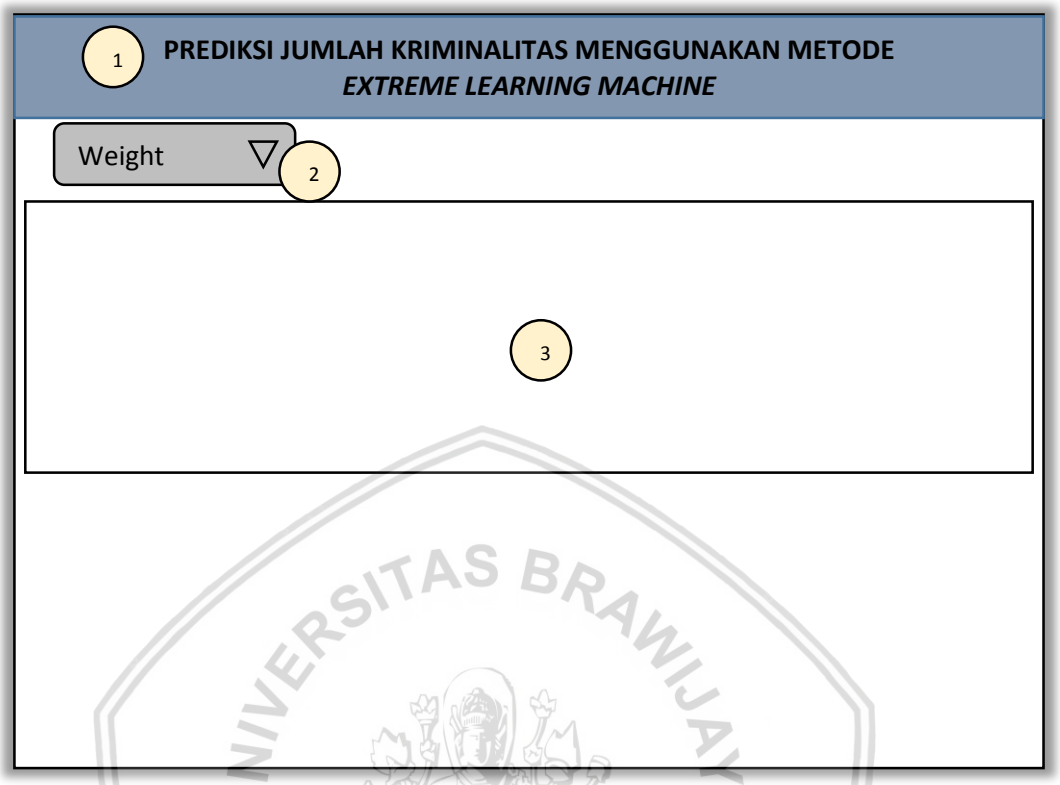


Gambar 4.19 Perancangan Halaman Normalisasi Data

Keterangan:

1. Header pada aplikasi
2. *Combo box* untuk memilih menu aplikasi. Menu yang dipilih akan tampil dengan warna abu-abu sebagai tanda bahwa menu normalisasi sedang aktif
3. Data *view* berupa tabel untuk menampilkan hasil normalisasi data *training*
4. Data *view* berupa tabel untuk menampilkan hasil normalisasi data *training*

4.4.3 Perancangan Halaman Weight

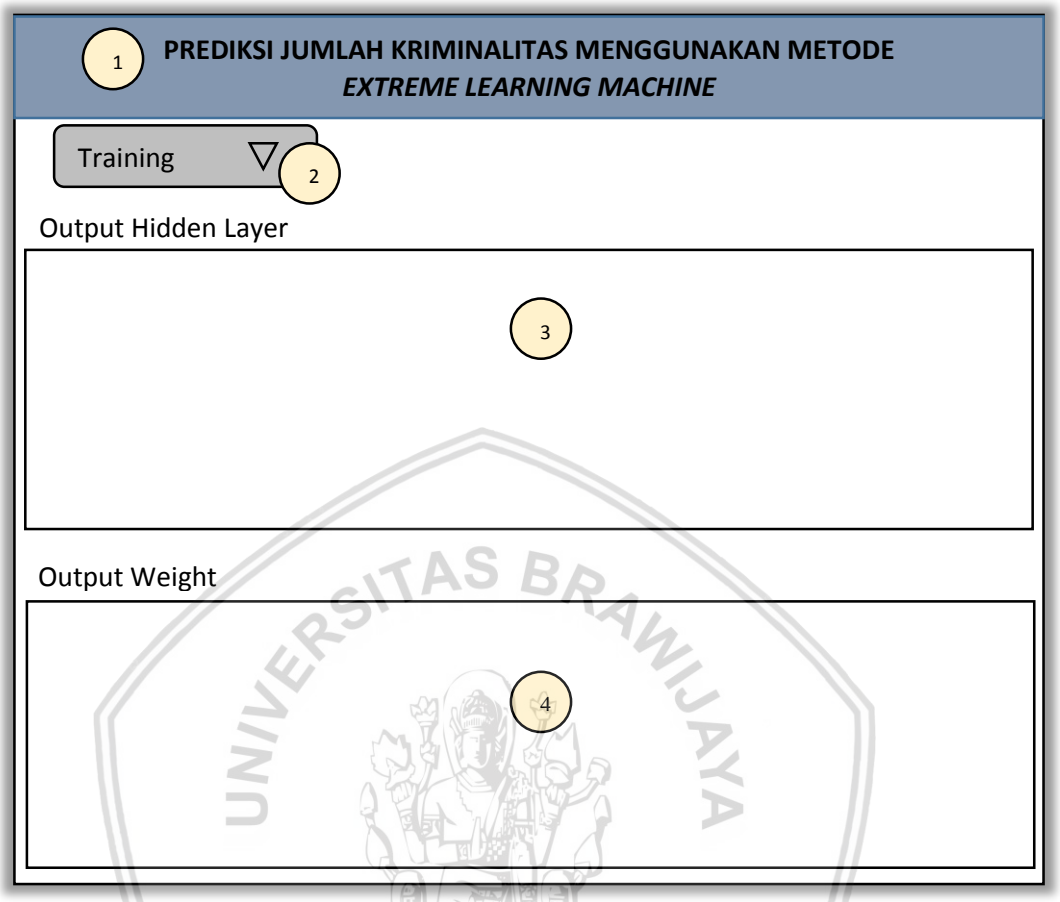


Gambar 4.20 Perancangan Halaman *Weight*

Keterangan:

1. Header pada aplikasi
2. *Combo box* untuk memilih menu aplikasi. Menu yang dipilih akan tampil dengan warna abu-abu. Hal ini sebagai tanda bahwa menu *weight* sedang aktif
3. Data *view* berupa tabel untuk menampilkan nilai *weight*

4.4.4 Perancangan Halaman *Training*

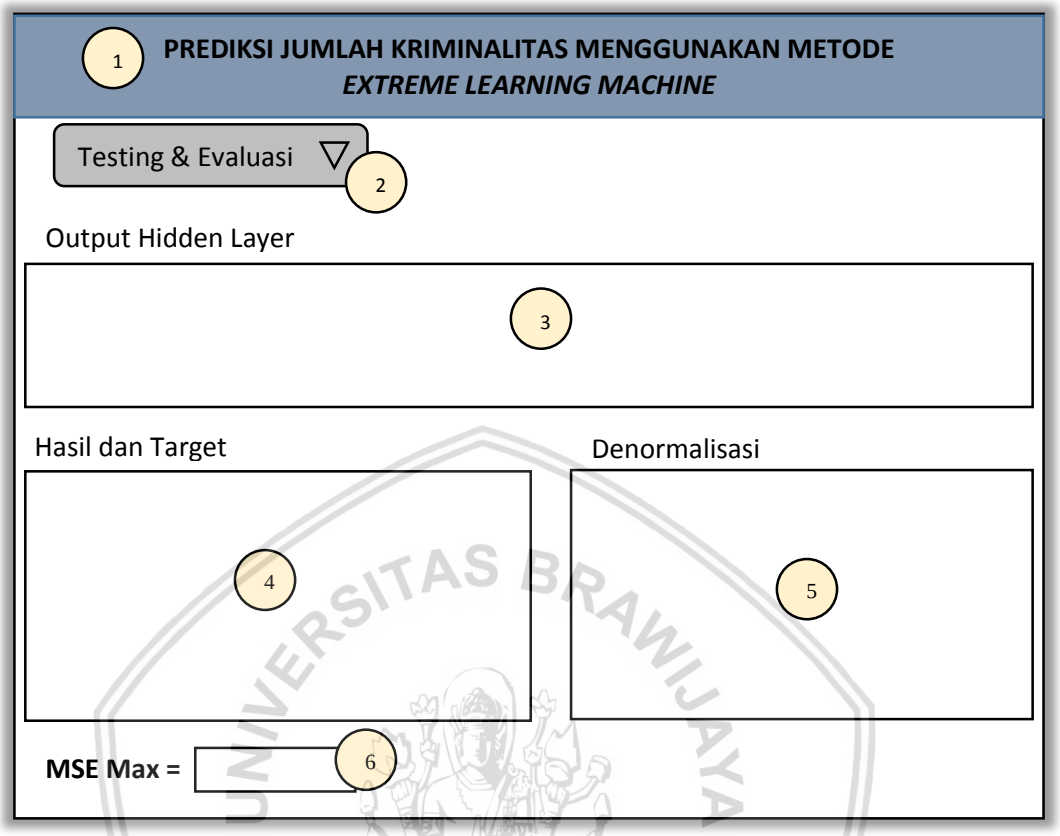


Gambar 4.21 Perancangan Halaman *Training*

Keterangan:

1. Header pada aplikasi
2. *Combo box* untuk memilih menu aplikasi. Menu yang dipilih akan tampil dengan warna abu-abu. Hal ini sebagai tanda bahwa menu *training* sedang aktif
3. Data *view* berupa tabel digunakan untuk menampilkan hasil perhitungan *output hidden layer*
4. Data *view* berupa tabel digunakan untuk menampilkan hasil dari perhitungan *output weight*

4.4.5 Perancangan Halaman Testing dan Evaluasi



Gambar 4.22 Perancangan Halaman *Testing* dan Evaluasi

Keterangan:

1. Header pada aplikasi
2. *Combo box* untuk memilih menu aplikasi. Menu yang dipilih akan tampil dengan warna abu-abu sebagai tanda bahwa menu *testing* dan evaluasi sedang aktif
3. Data *view* berupa tabel digunakan untuk menampilkan hasil perhitungan *output hidden layer*
4. Data *view* berupa tabel digunakan untuk menampilkan hasil dan target yang ternormaliasasi
5. Data *view* berupa tabel digunakan untuk menampilkan hasil dan target yang telah didenormalisasi
6. *Text box* digunakan untuk menampilkan nilai maksimal MSE

4.5 Perancangan Pengujian Sistem

Pada tahap perancangan pengujian sistem terkait algoritma *Extreme Learning Machine* (ELM) dilakukan guna menguji tingkat *error* dalam memprediksi jumlah kriminalitas. Skenario pengujian pada sistem ini adalah sebagai berikut:

a. Pengujian variasi fitur data

b. Pengujian rasio data
c. Pengujian fungsi aktivasi
d. Pengujian jumlah *neuron* pada *hidden layer*

4.5.1 Pengujian Variasi Fitur Data

Skenario pengujian variasi fitur data digunakan untuk mengetahui jumlah fitur terbaik yang sesuai dengan perolehan nilai MSE yang rendah pada prediksi jumlah kriminalitas. Pengujian ini menggunakan analisis teknikal dengan menggunakan data historis jumlah kriminalitas dari bulan-bulan sebelumnya untuk mengetahui hasil prediksi pada bulan berikutnya. Pada penelitian ini, jumlah fitur yang akan diuji coba adalah 2 sampai 7 fitur. Pengujian variasi fitur data ditunjukkan pada Tabel 4.17.

Tabel 4.17 Pengujian Variasi Fitur Data

Percobaan ke-*i*	**Nilai MSE Pada Jumlah Fitur Data**					
	2	**3**	**4**	**5**	**6**	**7**
1						
2						
3						
4						
5						
6						
7						
8						
9						
10						
Rata-Rata Nilai MSE						

4.5.2 Pengujian Rasio Data

Skenario pengujian rasio data digunakan untuk mengetahui seberapa besar pengaruh rasio data terhadap nilai MSE yang dihasilkan. Pengujian ini dilakukan dengan 6 jenis perbandingan data *training* dan data *testing* yaitu 80%:20%, 70%:30%, 60%:40%, 50%:50%, 40%:60%, dan 30%:70%. Sekenario pengujian rasio ditunjukkan pada Tabel 3.18.

Tabel 4.18 Perbandingan Pengujian Rasio Data

Percobaan ke-*i*	Nilai MSE Pada Perbandingan Rasio Data					
	80%:20%	70%:30%	60%:40%	50%:50%	40%:60%	30%:70%
1						
2						
3						
4						
5						
6						
7						
8						
9						
10						
Rata-Rata Nilai MSE						

4.5.3 Pengujian Fungsi Aktivasi

Skenario pengujian fungsi aktivasi bertujuan untuk melakukan transformasi suatu masukan menjadi suatu keluaran tertentu. Pada JST masukan akan menerima suatu informasi. Masukan ini akan diproses melalui suatu fungsi yang akan menjumlahkan beberapa masukan, kemudian hasil dari penjumlahan akan dilakukan perbandingan dengan nilai target (*threshold*) tertentu melalui fungsi aktivasi pada setiap *layer.*

Untuk menguji fungsi aktivasi yang akan diuji coba adalah fungsi aktivasi sigmoid biner dengan range 0 sampai 1 dan fungsi aktivasi sigmoid bipolar dengan range -1 sampai 1. Pengujian fungsi aktivasi ditunjukkan pada Tabel 4.19.

Tabel 4.19 Pengujian Fungsi Aktivasi

Percobaan ke-*i*	Nilai MSE Pada Fungsi Aktivasi	
	Sigmoid Biner	Sigmoid Bipolar
1		
2		
3		
4		
5		
6		

Tabel 4.19 Pengujian Fungsi Aktivasi (Lanjutan)

Percobaan ke-*i*	Nilai MSE Pada Fungsi Aktivasi	
	Sigmoid Biner	Sigmoid Bipolar
7		
8		
9		
10		
Rata-Rata Nilai MSE		

4.5.4 Pengujian Jumlah *Neuron* Pada *Hidden Layer*

Skenario pengujian jumlah *neuron* pada *hidden layer* memiliki tujuan untuk memperoleh hasil pengenalan yang lebih baik. Pada penelitian ini, jumlah *neuron* pada *hidden layer* yang diuji adalah 1 sampai 7. Pengujian jumlah *neuron* pada *hidden layer* ditunjukkan pada Tabel 4.20.

Tabel 4.20 Pengujian Jumlah *Neuron* Pada *Hidden Layer*

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

BAB 5 IMPLEMENTASI

Bab ini membahas mengenai implementasi sistem dan antarmuka berdasarkan analisis kebutuhan dan perancangan komputasi sistem yang telah dibuat. Implementasi sistem pada bab ini menggunakan metode *Extreme Learning Machine* (ELM) untuk memprediksi jumlah kriminalitas.

5.1 Spesifikasi Sistem

Pada sub bab ini akan dilakukan pembahasan mengenai spesifikasi dari sistem yang dibangun. Spesifikasi sistem terdiri dari spesifikasi perangkat keras dan spesifikasi perangkat lunak.

5.1.1 Spesifikasi Perangkat Keras

Berikut ini adalah spesifikasi perangkat keras yang digunakan dalam implementasi metode ELM untuk prediksi jumlah kriminalitas yang ditunjukkan pada Tabel 5.1.

Tabel 5.1 Spesifikasi Perangkat Keras

Nama Komponen	Spesifikasi
Prosesor	Intel(R) Core™ i3-4010U @ 1.70GHz
Memory RAM	2 GB
Hardisk	500 GB
Monitor	14 inch

5.1.2 Spesifikasi Perangkat Lunak

Berikut ini adalah spesifikasi perangkat lunak yang digunakan dalam implementasi metode ELM untuk prediksi jumlah kriminalitas yang ditunjukkan pada Tabel 5.2.

Tabel 5.2 Spesifikasi Perangkat Lunak

Nama	Spesifikasi
Sistem Operasi	Windows 7
Bahasa Pemrograman	Java
Tools Pemrograman	Netbeans IDE 8.0.1

5.2 Implementasi Algoritma

Pada implementasi untuk prediksi jumlah kriminalitas menggunakan metode ELM memiliki beberapa algoritma, diantaranya algoritma normalisasi data, algoritma inisialisasi *input weight*, algoritma keluaran *hidden layer* dengan fungsi aktivasi, algoritma *output weight*, algoritma keluaran keluaran *output layer*, algoritma hitung MSE, dan algoritma denormalisasi data.

5.2.1 Implementasi Algoritma Normalisasi Data

Langkah awal yang dilakukan oleh sistem yaitu menormalisasikan keseluruhan data ke dalam interval 0 hinggal 1 (sigmoid biner) atau -1 hingga 1 (sigmoid bipolar). Hal pertama yang harus dilakukan dalam normalisasi data adalah mencari nilai maksimum dan minimum data dengan menggunakan *Min-Max Normalization.* Prosedur normalisasi data dapat dilihat pada Kode Program 5.1.

Kode Program 5.1 Implementasi Algoritma Normalisasi Data

```
1  public double[][] Normalisasi(double[][] isi_Data) {
2          max = 0;
3          min = Double.MAX_VALUE;
4          double data_normal[][] = isi_Data;
5          for (int i = 0; i < isi_Data.length; i++) {
6              for (int j = 0; j < isi_Data[0].length; j++) {
7                  if (isi_Data[i][j] > max) {
8                      max = isi_Data[i][j];
9                  }
10                 if (isi_Data[i][j] < min) {
11                     min = isi_Data[i][j];
12                 }
13             }
14         }
15
16         for (int i = 0; i < isi_Data.length; i++) {
17             for (int j = 0; j < isi_Data[0].length; j++) {
18                 data_normal[i][j] = (isi_Data[i][j] - min) /
19                 (max - min);
20             }
21         }
22         return data_normal;
23     }
```

Berikut penjelasan Kode Program 5.1 mengenai Implementasi algoritma normalisasi data.

1. Baris 4 sampai 11 menjelaskan mengenai proses pencarian nilai minimal dan maksimal dari keseluruhan data.
2. Baris 16 sampai 19 menjelaskan mengenai proses perhitungan normalisasi sesuai Persamaan 2.6.

5.2.2 Implementasi Algoritma Inisialisasi *Input Weight*

Algoritma inisialisasi *input weight* merupakan proses untuk menghitung keluaran pada *hidden layer* dengan melakukan inisialisasi *input weight* terlebih dahulu. Inisialisasi *input weight* ditentukan menggunakan angka *random* dengan range antara -1 dan 1. *Input weight* yang diperoleh membentuk matrik dengan ordo sesuai masukan jumlah fitur dan jumlah *hidden neuron* oleh pengguna. Implementasi algoritma inisialisasi *input weight* dapat dilihat pada Kode Program 5.2.

Kode Program 5.2 Implementasi Algoritma Inisialisasi *Input Weight*

```
1   public double[][] Bobot(int jumlah_hidden, int jumlah_fitur)
2   {
3           double bobot[][] = new
4           double[jumlah_hidden][jumlah_fitur];
5           double max = 1;
6           double min = -1;
7           double range = (max - min);
8           for (int i = 0; i < jumlah_hidden; i++) {
9               for (int j = 0; j < jumlah_fitur; j++) {
10                  double weight = (Math.random() * range) + min;
11                  bobot[i][j] = weight;
12              }
13          }
14          return bobot;
15      }
```

Berikut penjelasan Kode Program 5.2 mengenai Implementasi algoritma inisialisasi *input weight*.

1. Baris 5 dan 6 menjelaskan mengenai proses penentuan batas nilai minimum dan maksimum.
2. Baris 7 sampai 11 menjelaskan mengenai proses inisialisasi *input weight* secara *random*.

5.2.3 Implementasi Algoritma Hitung Keluaran *Hidden Layer* dengan Fungsi Aktivasi

Algoritma hitung keluaran *hidden layer* dengan fungsi aktivasi merupakan proses perhitungan keluaran pada *hidden layer* menggunakan fungsi aktivasi. Kemudian dilakukan pencarian matriks *transpose* dari hasil keluaran *hidden layer* yang telah diaktivasi. Dilanjutkan perkalian matriks dan invers matriks. Implementasi hitung keluaran *hidden layer* dengan fungsi aktivasi dapat dilihat pada Kode Program 5.3.

Kode Program 5.3 Implementasi Algoritma Hitung Keluaran *Hidden Layer* dengan Fungsi Aktivasi

```
1   public double[][] outputHidden(double[][] weight, double[][]
2   temp_input, int aktivasi) {
3           double[][] wt = transpose(weight);
4           double[][] Hinit = perkalian(temp_input, wt);
5           double[][] hasilOutputHidden;
6           switch (aktivasi) {
7               case 1:
8                   hasilOutputHidden = aktivasi_biner(Hinit);
9                   break;
10              case 2:
11                  hasilOutputHidden = aktivasi_bipolar(Hinit);
12                  break;
13              default:
14                  hasilOutputHidden = aktivasi_biner(Hinit);
15                  break;
16          }
17          return hasilOutputHidden;
18      }
```

```
19
20  double[][] aktivasi_biner(double[][] Hinit) {
21          double[][] hasilOutputHidden = new
22          double[Hinit.length][Hinit[0].length];
23          for (int i = 0; i < Hinit.length; i++) {
24              for (int j = 0; j < Hinit[0].length; j++) {
25                  hasilOutputHidden[i][j] = 1 / 1 + Math.exp(-
26                  Hinit[i][j]);
27              }
28          }
29          return hasilOutputHidden;
30      }
31
32  double[][] aktivasi_bipolar(double[][] Hinit) {
33          double[][] hasilOutputHidden = new
34          double[Hinit.length][Hinit[0].length];
35          for (int i = 0; i < Hinit.length; i++) {
36              for (int j = 0; j < Hinit[0].length; j++) {
37                  hasilOutputHidden[i][j] = (1 - Math.exp(-
38                  Hinit[i][j])) / (1 + Math.exp(-Hinit[i][j]));
39              }
40          }
41          return hasilOutputHidden;
42      }
```

Berikut penjelasan Kode Program 5.3 mengenai Implementasi algoritma hitung keluaran *hidden layer* dengan fungsi aktivasi.

1. Baris 3 deklarasi array *transpose* matriks *weight*
2. Baris 4 deklarasi array perkalian matriks antara input dengan weight
3. Baris 5 deklarasi array hasil output pada *hidden layer*
4. Baris 6 sampai 18 menjelaskan mengenai proses perhitungan fungsi aktivasi untuk memilih jenis aktivasi dengan memanggil method perhitungan fungsi aktivasi.
5. Baris 20 sampai 30 menjelaskan mengenai proses perhitungan keluaran *hidden layer* menggunakan fungsi aktivasi sigmoid biner
6. Baris 32 sampai 42 menjelaskan mengenai proses perhitungan keluaran *hidden layer* menggunakan fungsi aktivasi sigmoid bipolar

5.2.4 Implementasi Algoritma *Transpose* Matriks

Algoritma *transpose* matriks merupakan hasil *transpose* dari matriks keluaran pada *hidden layer* dengan fungsi aktivasi. Implementasi algoritma *transpose* matriks dapat dilihat pada Kode Program 5.4.

Kode Program 5.4 Implementasi Algoritma *Transpose* Matriks

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

```
10          return transpose;
11      }
```

Penjelasan Kode Program 5.4 mengenai Implementasi algoritma *transpose* matriks adalah pada baris 4 sampai 6 dengan proses *transpose* pada matriks keluaran *hidden layer* dengan fungsi aktivasi.

5.2.5 Implementasi Algoritma Perkalian Matriks

Algoritma perkalian matriks merupakan proses perkalian kedua matriks dengan cara mengalikan bilangan-bilangan yang terdapat pada kolom matriks A dengan bilangan-bilangan yang terdapat pada baris matriks B. Implementasi algoritma perkalian matriks dapat dilihat pada Kode Program 5.5.

Kode Program 5.5 Implementasi Algoritma Perkalian Matriks

```
1   public double[][] perkalian(double[][] A, double[][] B) {
2           int barisA = A.length;
3           int barisB = B.length;
4           int kolomB = B[0].length;
5           double hasil[][] = new double[barisA][kolomB];
6           for (int i = 0; i < barisA; i++) {
7               for (int j = 0; j < kolomB; j++) {
8                   double x = 0;
9                   for (int k = 0; k < barisB; k++) {
10                      x += A[i][k] * B[k][j];
11                  }
12                  hasil[i][j] = x;
13              }
14          }
15          return hasil;
16      }
```

Penjelasan Kode Program 5.5 mengenai Implementasi algoritma perkalian matriks adalah pada baris 6 sampai 12 dengan proses perkalian antara kedua matriks.

5.2.6 Implementasi Algoritma *Invers* Matriks

Algoritma *invers* matriks merupakan hasil *transpose* dari matriks keluaran pada *hidden layer* dengan fungsi aktivasi. Implementasi algoritma *transpose* matriks dapat dilihat pada Kode Program 5.6.

Kode Program 5.6 Implementasi Algoritma *Invers* Matriks

```
1   public double[][] inverse_obe(double[][] data) {
2           int baris = data.length, kolom = data[0].length;
3           double[][] hasil_inverse = new double[baris][kolom];
4           double[][] temp = new double[baris][kolom * 2];
5           double[][] temp1 = new double[baris][kolom * 2];
6           for (int i = 0; i < baris; i++) {
7               for (int j = 0; j < kolom * 2; j++) {
8                   if (j - i == baris) {
9                       temp[i][j] = 1;
10                  }
11                  if (j < kolom) {
12                      temp[i][j] = data[i][j];
```

```
13                  }
14              }
15          }
16          for (int i = 0; i < baris; i++) {
17              for (int j = 0; j < baris; j++) {
18                  for (int k = 0; k < kolom * 2; k++) {
19                      if (i == j) {
20                          temp1[i][k] = temp[i][k]/temp[i][i];
21                      } else {
22                          temp1[j][k] = temp[j][k];
23                      }
24                  }
25              }
26              temp = pindah(temp1);
27              for (int j = i + 1; j < baris; j++) {
28                  for (int k = 0; k < kolom * 2; k++) {
29                      temp1[j][k] = temp[j][k] - temp[i][k] *
30                      temp[j][i];
31                  }
32              }
33              temp = pindah(temp1);
34          }
35          for (int i = kolom - 1; i > 0; i--) {
36              for (int j = i - 1; j >= 0; j--) {
37                  for (int k = 0; k < kolom * 2; k++) {
38                      temp1[j][k] = temp[j][k] - temp[i][k] *
39                      temp[j][i];
40                  }
41              }
42              temp = pindah(temp1);
43          }
44          for (int i = 0; i < baris; i++) {
45              for (int j = kolom; j < kolom * 2; j++) {
46                  hasil_inverse[i][j - kolom] = temp1[i][j];
47              }
48          }
49          return hasil_inverse;
50      }
51
52      public double[][] pindah(double[][] data) {
53          double[][] hasil = new
54          double[data.length][data[0].length];
55          for (int i = 0; i < data.length; i++) {
56              System.arraycopy(data[i], 0, hasil[i], 0,
57              data[0].length);
58          }
59          return hasil;
60      }
```

Berikut penjelasan Kode Program 5.6 mengenai Implementasi algoritma *invers* matriks.

1. Baris 6 sampai 15 merupakan deklarasi matriks identitas dan gabungan data
2. Baris 16 sampai 34 untuk mengubah baris dibawah diagonal menjadi nilai 0
3. Baris 35 sampai 43 untuk mengubah baris diatas diagonal menjadi nilai 0
4. Baris 44 sampai 50 merupakan dekralasi hasil invers
5. Baris 52 sampai 60 merupakan method untuk menyalin semua elemen di dalam array ke dalam array yang baru

5.2.7 Implementasi Algoritma Hitung *Output Weight*

Algoritma hitung *output weight* merupakan proses perhitungan keluaran pada *hidden layer* dengan melibatkan matriks *Moore-Penrose Generalized Invers* (H^+) yang telah dihitung sebelumnya dan matriks target. Implementasi algoritma hitung *output weight* dapat dilihat pada Kode Program 5.7.

Kode Program 5.7 Implementasi Algoritma Hitung *Output Weight*

```
1  double[][] hplus(double hinit[][]) {
2          double Htranspos[][] = transpose(hinit);
3          double temp_HtH[][] = perkalian(Htranspos, hinit);
4          double temp_Hinverse[][] = inverse_obe(temp_HtH);
5          double Hplus[][] = perkalian(temp_Hinverse,
6          Htranspos);
7          return Hplus;
8      }
9
10 double[] beta(double[][] Hplus, double[] target) {
11         double[] hasil_Beta = new double[Hplus.length];
12         for (int i = 0; i < Hplus.length; i++) {
13             for (int k = 0; k < target.length; k++) {
14                 hasil_Beta[i] += Hplus[i][k] * target[k];
15             }
16         }
17         return hasil_Beta;
18     }
```

Berikut penjelasan Kode Program 5.7 mengenai Implementasi algoritma hitung *output weight*.

1. Baris 2 dan 3 merupakan deklarasi array matriks *transpose* dan matriks perkalian
2. Baris 4 deklarasi array perhitungan *invers* matriks
3. Baris 5 sampai 6 merupakan deklarasi array matriks *Moore-Penrose Generalized Invers* (H^+)
4. Baris 12 sampai 20 menjelaskan mengenai proses perhitungan *output weight* (β) sesuai Persamaan 2.4.

5.2.8 Implementasi Algoritma Hitung Keluaran *Output Layer*

Algoritma hitung keluaran *output layer* merupakan proses perhitungan keluaran pada *output layer* dengan menggunakan nilai *output weight* yang telah diperoleh sebelumnya. Implementasi algoritma hitung keluaran *output layer* dapat dilihat pada Kode program 5.8.

Kode Program 5.8 Implementasi Algoritma Hitung Keluaran *Output Layer*

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

```
8           return hasil_y;
9       }
```

Penjelasan Kode Program 5.8 mengenai Implementasi algoritma hitung keluaran *output layer* adalah pada baris 3 sampai 5 dengan proses perhitungan keluaran pada *output layer* (*y*) sesuai Persamaan 2.5.

5.2.9 Implementasi Algoritma Hitung *Mean Square Error* (MSE)

Perhitungan mencari nilai *Mean Square Error* (MSE) dengan melibatkan nilai target dan nilai *output* (hasil prediksi). Implementasi algoritma hitung *Mean Square Error* (MSE) dapat dilihat pada Kode Program 5.9.

Kode Program 5.9 Implementasi Algoritma Hitung MSE

```
1  public double MSE(double[] target_uji, double[] y) {
2          double tot = 0;
3          for (int i = 0; i < target_uji.length; i++) {
4              double YminT = y[i] - target_uji[i];
5              double YminT_kuadrat = Math.pow(YminT, 2);
6              tot += YminT_kuadrat;
7          }
8          return tot / target_uji.length;
9      }
```

Berikut penjelasan Kode Program 5.9 mengenai Implementasi algoritma Hitung MSE.

1. Baris 3 menjelaskan mengenai proses perulangan untuk target uji ke-*i* sampai panjang target ujinya.
2. Baris 4 sampai 8 menjelaskan mengenai proses perhitungan nilai MSE sesuai Persamaan 2.8.

5.2.10 Implementasi Algoritma Denormalisasi Data

Algoritma denormalisasi data merupakan proses perhitungan yang melibatkan nilai minimal dan maksimal yang diperoleh dari proses normalisasi data. Keluaran yang dihasilkan berupa data asli dari hasil prediksi. Implementasi algoritma denormalisasi data dapat dilihat pada Kode Program 5.10.

Kode Program 5.10 Implementasi Algoritma Denormalisasi Data

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

Berikut penjelasan Kode Program 5.10 mengenai Implementasi algoritma denormalisasi data.

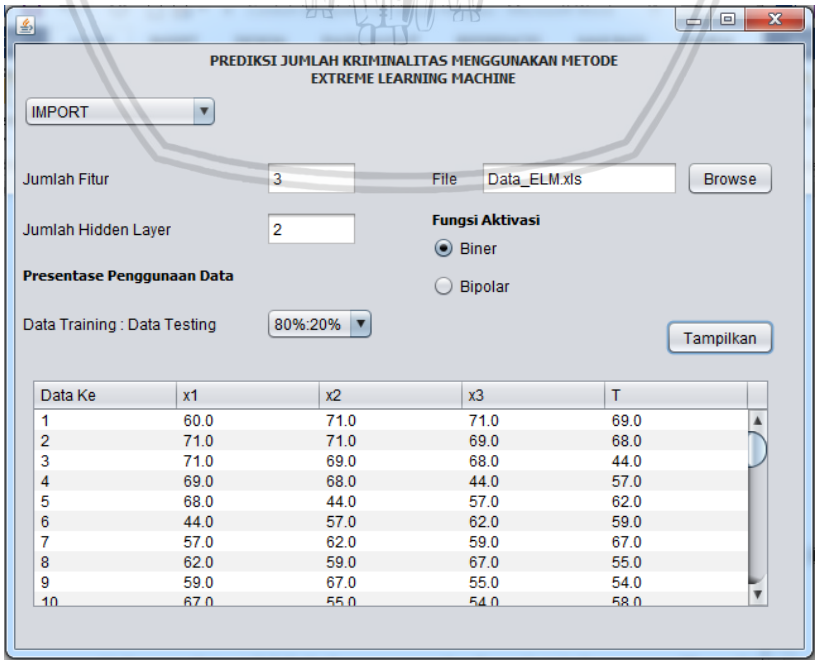
1. Baris 5 menjelaskan mengenai proses perulangan untuk data uji ke-*i* sampai panjang datanya.
2. Baris 6 sampai 7 menjelaskan mengenai proses perhitungan denormalisasi sesuai Persamaan 2.7.

5.3 Implementasi Antarmuka

Implementasi antarmuka merupakan tampilan dari sistem yang dibangun berdasarkan perancangan antarmukan yang telah dibuat sebelumnya. Antarmuka dari implementasi prediksi jumlah kriminalitas menggunakan metode ELM terdiri dari 5 halaman utama, yaitu halaman *import* data, halaman normalisasi data, halaman *weight*, halaman *training*, serta halaman *testing* dan evaluasi.

5.3.1 Implementasi Halaman *Import* Data

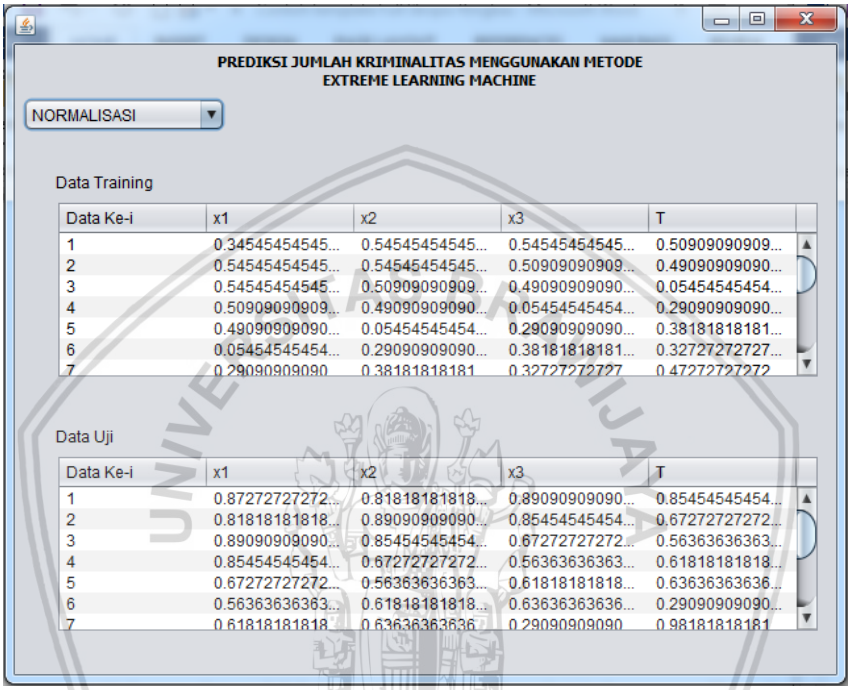
Halaman *import* data merupakan halaman utama pada sistem prediksi jumlah kriminalitas, dimana halaman ini dapat menerima beberapa masukan. Langkah awal yang harus dilakukan pengguna pada halaman *import* data yaitu menekan tombol ”Browse” dengan tujuan untuk memasukkan file berisi data jumlah kriminalitas yang akan diproses. *File* yang dimasukkan berdasarkan format .xls dan akan tampil lokasi *file* pada *text box file*. Setelah itu pengguna memasukkan *input* jumlah fitur, jumlah *hidden layer*, presentase penggunaan data *training* dan data *testing* dan fungsi aktivasi. Langkah berikutnya pengguna menekan tombol ”Tampilkan” yang berguna untuk menyalin data dari *file*, kemudian data diubah ke dalam array sesuai jumlah fitur yang dimasukkan. Data yang ditampilkan berupa tabel. Gambar 5.1 merupakan implementasi halaman *import* data.



Gambar 5.1 Implementasi Halaman *Import* Data

5.3.2 Implementasi Halaman Normalisasi Data

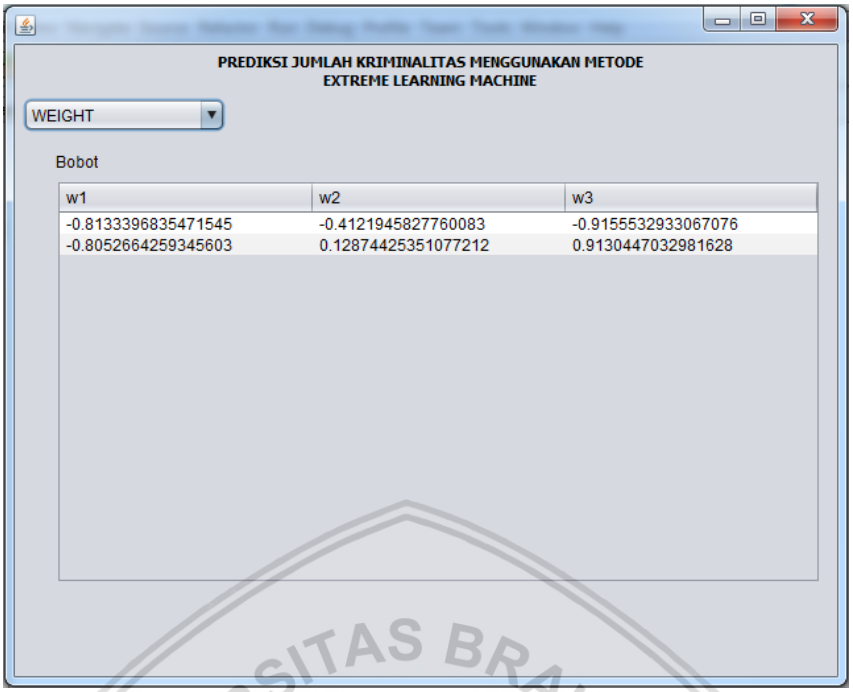
Halaman normalisasi data adalah halaman kedua pada sistem prediksi jumlah kriminalitas yang terdiri dari data latih dan data uji. Masing-masing bagian berisi data jumlah kriminalitas yang dinormalisasi dengan metode *Min-Max Normalization*. Data tersebut telah dilakukan perhitungan sejak awal pengguna menekan tombol ”Tampilkan”. Hasil perhitungan normalisasi ditampilkan dalam bentuk tabel. Gambar 5.2 merupakan implementasi halaman normalisasi data.



Gambar 5.2 Implementasi Halaman Normalisasi Data

5.3.3 Implementasi Halaman *Weight*

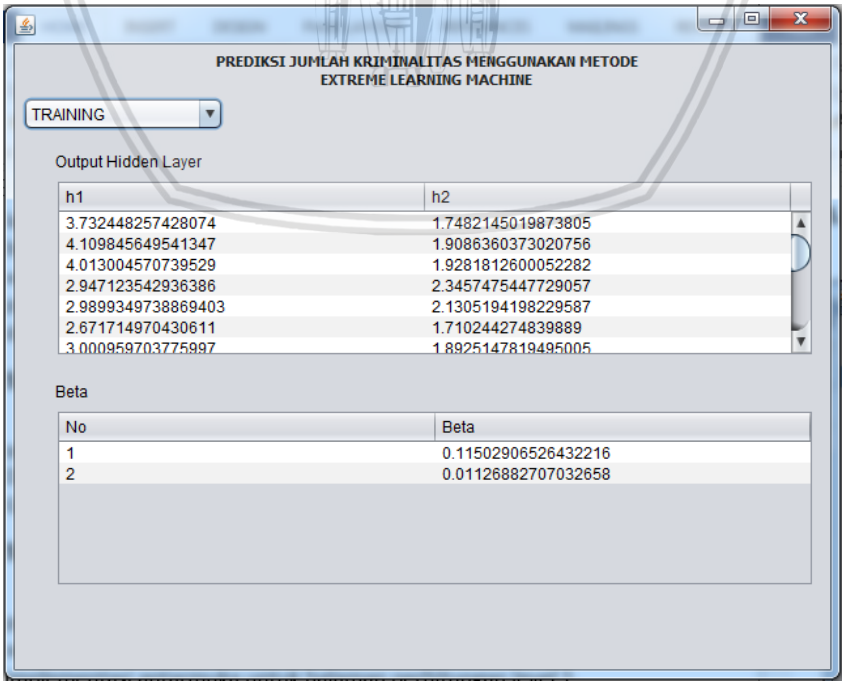
Halaman *weight* adalah halaman ketiga pada sistem prediksi jumlah kriminalitas yang berisi nilai *input weight* atau bobot yang dihitung secara acak (*random*). Nilai bobot tersebut dihitung sejak pengguna menekan tombol ”Tampilkan”, kemudian ditampilkan dalam bentuk tabel dengan ordo sesuai banyaknya jumlah fitur dan jumlah *hidden neuron* yang dimasukan. Gambar 5.3 merupakan implementasi halaman *weight*.



Gambar 5.3 Implementasi Halaman *Weight*

5.3.4 Implementasi Halaman *Training*

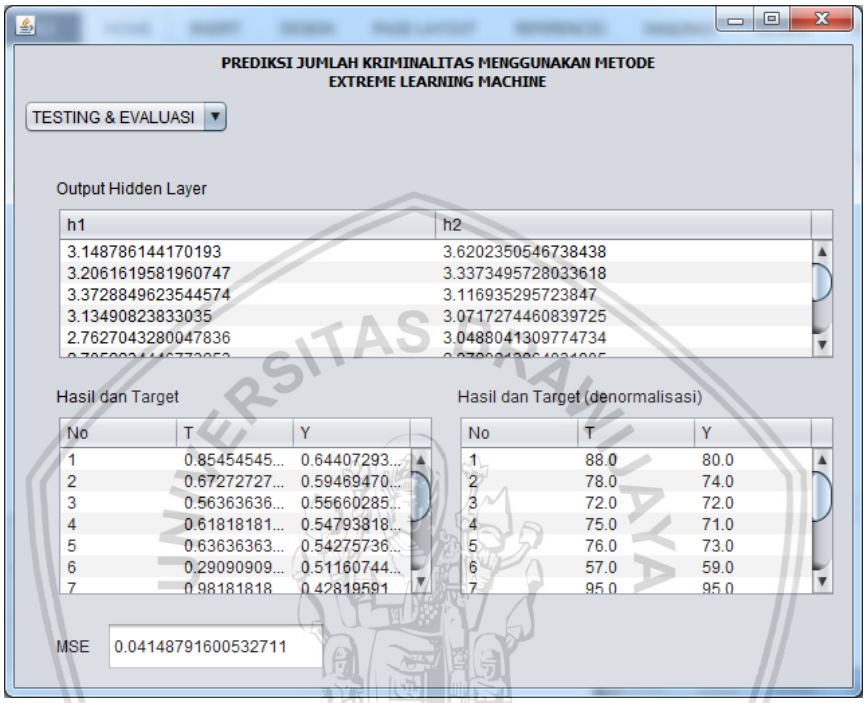
Halaman *training* adalah halaman ketiga pada sistem prediksi jumlah kriminalitas yang berisi nilai *output hidden layer* yang telah diaktivasi pada data *training* dan hasil *output weight* (β). Nilai tersebut dihitung sejak pengguna menekan tombol "Tampilkan", kemudian ditampilkan dalam bentuk tabel. Gambar 5.4 merupakan implementasi halaman *training*.



Gambar 5.4 Implementasi Halaman *Training*

5.3.5 Implementasi Halaman *Testing* dan Evaluasi

Halaman *testing* dan evaluasi adalah halaman terakhir pada sistem prediksi jumlah kriminalitas yang berisi hasil *output hidden layer* yang telah diaktivasi pada data *testing*, hasil dan target yang ternormalisasi, serta hasil dan target yang didenormalisasi, dan juga hasil MSE terbaik. Perhitungan seluruhnya dilakukan sejak awal pengguna menekan tombol ”Tampilkan” dan ditampilkan dalam bentuk tabel. Gambar 5.5 merupakan implementasi halaman *testing* dan evaluasi.



Gambar 5.5 Implementasi Halaman *Testing* dan Evaluasi

BAB 6 PENGUJIAN DAN PEMBAHASAN

Bab ini menjelaskan mengenai hasil pengujian dari sistem prediksi jumlah kriminalitas menggunakan metode ELM. Pengujian yang dilakukan meliputi pengujian variasi fitur data, pengujian perbandingan rasio, pengujian fungsi aktivasi dan pengujian jumlah *neuron* pada *hidden layer*.

6.1 Pengujian Variasi Fitur Data

Pengujian variasi fitur data ini menjelaskan mengenai hasil skenario pengujian variasi fitur data dan analisis skenario pengujian variasi fitur data.

6.1.1 Skenario Pengujian Variasi Fitur Data

Pengujian variasi fitur data dilakukan guna melihat jumlah fitur yang digunakan untuk menghasilkan nilai MSE terbaik. Dalam pengujian ini dilakukan pecobaan sebanyak 10 kali dari jumlah dataset sebanyak 68 data. Jumlah fitur yang akan diuji memiliki jumlah dengan range 2 sampai 7. Jumlah fitur yang digunakan pada pengujian ini merupakan perhitungan jumlah kriminalitas berdasarkan 2 bulan sebelumnya, 3 bulan sebelumnya, 4 bulan sebelumnya hingga 7 bulan sebelumnya. Jumlah *hidden layer* yang digunakan sebanyak 4. Presentase data yang digunakan adalah perbandingan rasio data *training* dan data *testing* sebesar 70%:30% dengan menggunakan fungsi aktivasi sigmoid biner. Setiap kali program dijalankan, nilai MSE yang dihasilkan dari sistem ELM berbeda-beda. Hal tersebut dikarenakan inisialisasi nilai *input weight* dilakukan secara *random* (nilai *input weight* setiap kali percobaan disertakan pada lampiran B). Dan diakhiri dengan evaluasi rata-rata nilai MSE. Hasil pengujian variasi fitur data ditunjukkan dpada Tabel 6.1.

Tabel 6.1 Hasil Pengujian Variasi Fitur Data

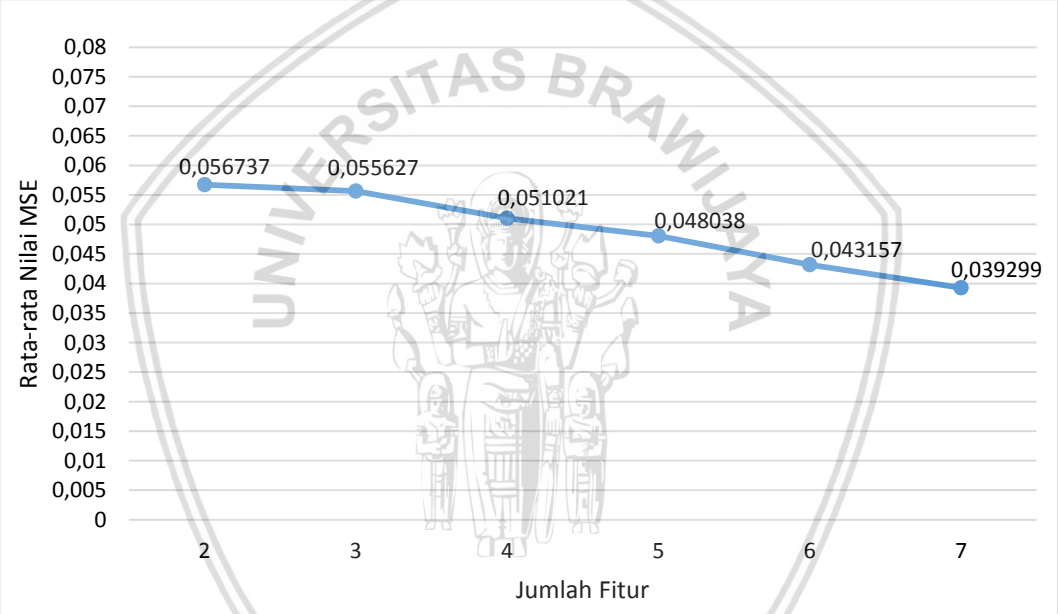
Percobaan ke-*i*	Nilai MSE Pada Jumlah Fitur Data					
	2	**3**	**4**	**5**	**6**	**7**
1	0,051202	0,054492	0,048021	0,051251	0,049895	0,042510
2	0,054745	0,067762	0,059530	0,038674	0,039258	0,046983
3	0,059731	0,047908	0,051256	0,042439	0,047025	0,041356
4	0,055348	0,050222	0,051153	0,055049	0,045243	0,041416
5	0,055877	0,052189	0,054804	0,060704	0,043348	0,031946
6	0,056115	0,056303	0,052996	0,048401	0,045596	0,036724
7	0,056312	0,069019	0,048949	0,047695	0,037929	0,042200
8	0,059074	0,053517	0,046107	0,048230	0,042524	0,039047
9	0,059123	0,059589	0,050226	0,044939	0,042347	0,036907

Tabel 6.1 Hasil Pengujian Variasi Fitur Data (Lanjutan)

Percobaan ke-*i*	Nilai MSE Pada Jumlah Fitur Data					
	2	**3**	**4**	**5**	**6**	**7**
10	0,059844	0,045264	0,047168	0,042994	0,038403	0,033902
Rata-Rata Nilai MSE	0,056737	0,055627	0,051021	0,048038	0,043157	0,039299

6.1.2 Analisis Pengujian Variasi Fitur Data

Berdasarkan Tabel 6.1 diketahui bahwa perolehan rata-rata nilai MSE terkecil terletak pada variasi fitur data sebanyak 7 fitur sebesar 0,039299. Hasil rata-rata nilai MSE pengujian variasi fitur data dapat dilihat pada Gambar 6.1.



Gambar 6.1 Grafik Hasil Pengujian Variasi Jumlah Fitur

Variasi jumlah fitur memiliki tujuan untuk memperoleh hasil pengenalan yang lebih baik dari perolehan jumlah fitur yang tepat. Analisis dilakukan pada pengujian variasi fitur data yaitu besar nilai MSE terhadap pengaruh jumlah fitur. Hasil uji yang diperoleh menunjukkan nilai MSE paling kecil ditunjukkan pada variasi fitur sebanyak 7 fitur yang memiliki nilai 0,039299. Hal ini dapat disimpulkan bahwa semakin banyak masukan jumlah fitur maka hasil prediksi yang dihasilkan cenderung yang terbaik, karena selain fitur juga bergantung pada objek yang digunakan untuk memprediksi dan menghasilkan pola data yang lebih baik.

6.2 Pengujian Rasio Data

Pengujian perbandingan rasio ini menjelaskan mengenai hasil skenario pengujian perbandingan rasio dan analisis skenario pengujian perbandingan rasio.

6.2.1 Skenario Pengujian Rasio Data

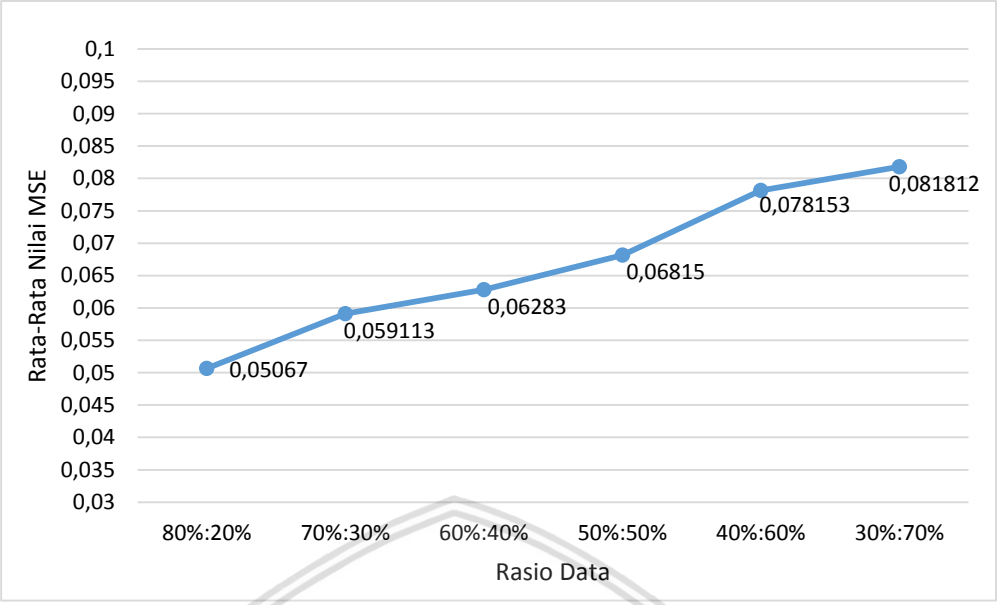
Pengujian rasio data merupakan pengujian berdasarkan data *training* dan data *testing* yang dilakukan untuk mengetahui pengaruh dari pengujian perbandingan rasio data. Dalam pengujian ini dilakukan pecobaan sebanyak 10 kali dari jumlah dataset sebanyak 68 data. Terdapat 6 jenis pengujian rasio data yang digunakan yaitu 80%:20%, 70%:30%, 60%:40%, 50%:50%, 40%:60%, dan 30%:70%. Jumlah fitur data yang digunakan pada pengujian ini sebanyak 7 dan jumlah *hidden layer* sebanyak 4 dengan fungsi aktivasi sigmoid biner. Setiap kali program dijalankan, nilai MSE yang dihasilkan dari sistem ELM berbeda-beda. Hal tersebut dikarenakan inisialisasi nilai *input weight* dilakukan secara *random* (nilai *input weight* setiap kali percobaan disertakan pada lampiran B). Dan diakhiri dengan evaluasi rata-rata nilai MSE. Hasil pengujian rasio perbandingan data ditunjukkan pada Tabel 6.2.

Tabel 6.2 Hasil Pengujian Rasio Perbandingan Data

Percobaan ke-*i*	Nilai MSE Pada Rasio Perbandingan Data					
	80%:20%	**70%:30%**	**60%:40%**	**50%:50%**	**40%:60%**	**30%:70%**
1	0,049518	0,060975	0,065081	0,087862	0,081102	0,074567
2	0,051503	0,062356	0,067346	0,063034	0,071855	0,065657
3	0,064115	0,063287	0,059985	0,054868	0,081416	0,077913
4	0,049897	0,056778	0,054898	0,084759	0,070001	0,067179
5	0,057714	0,064199	0,054364	0,057975	0,077030	0,073573
6	0,039566	0,054694	0,073183	0,070135	0,079373	0,113265
7	0,041361	0,060564	0,069334	0,061980	0,079576	0,098164
8	0,047724	0,060313	0,063059	0,074068	0,085090	0,068140
9	0,062804	0,048831	0,060810	0,073212	0,081099	0,096413
10	0,042500	0,059134	0,060244	0,053606	0,074990	0,083245
Rata-Rata Nilai MSE	0,050670	0,059113	0,062830	0,068150	0,078153	0,081812

6.2.2 Analisis Pengujian Rasio Data

Bardasarkan Tabel 6.2 diketahui bahwa perolehan rata-rata nilai MSE sebesar 0,050670 yang terletak pada pengujian rasio perbandingan 80%:20%. Sehingga rasio pada perbandingan 80%:20% tersebut akan digunakan untuk pengujian berikutnya. Hasil rata-rata nilai MSE pengujian rasio perbandingan data dapat dilihat pada Gambar 6.2.



Gambar 6.2 Grafik Hasil Pengujian Rasio Perbandingan Data

Analisis pengujian rasio data dilakukan untuk mengetahui pengaruh rasio perbandingan data *training* dan data *testing* terhadap nilai *error* (MSE) yang dihasilkan. Data *training* merupakan data latih yang digunakan untuk proses peramalan pada pembelajaran metode ELM. Pada proses *training*, data *training* memberikan nilai *input weight* dan *output weight* yang akan diteruskan pada proses *testing*. Data *testing* merupakan data uji yang digunakan untuk proses perhitungan hasil ramalan dan kesalahan ramalan berdasarkan nilai MSE.

Berdasarkan grafik pada Gambar 6.2, rata-rata nilai MSE terbaik 0,050670 terletak pada rasio perbandingan 80%:20%. Nilai *error* yang dihasilkan tersebut mengalami penurunan cukup signifikan. Hal ini terjadi karena pengaruh data dan nilai *input weight* yang ditentukan secara *random*. Oleh sebab itu banyaknya data *training* dan data *testing* sangat berpengaruh pada perolehan nilai *error*. Semakin banyak data pelatihan (*training*) yang dilakukan maka perolehan nilai *error* semakin baik.

6.3 Pengujian Fungsi Aktivasi

Pengujian fungsi aktivasi ini menjelaskan mengenai hasil skenario pengujian fungsi aktivasi dan analisis skenario pengujian fungsi aktivasi.

6.3.1 Skenario Pengujian Fungsi Aktivasi

Pengujian fungsi aktivasi dilakukan untuk mengetahui nilai *output* yang diinginkan (target) dari suatu nilai input. Dalam pengujian ini dilakukan pecobaan sebanyak 10 kali dari jumlah dataset sebanyak 68 data. Terdapat 2 jenis fungsi aktivasi yang digunakan yaitu fungsi aktivasi sigmoid biner dan fungsi aktivasi sigmoid bipolar. Karena kedua fungsi aktivasi tersebut biasa digunakan dalam peramalan. Presentase data yang digunakan adalah perbandingan rasio data *training* dan data *testing* sebesar 80%:20%, jumlah fitur data yang digunakan pada

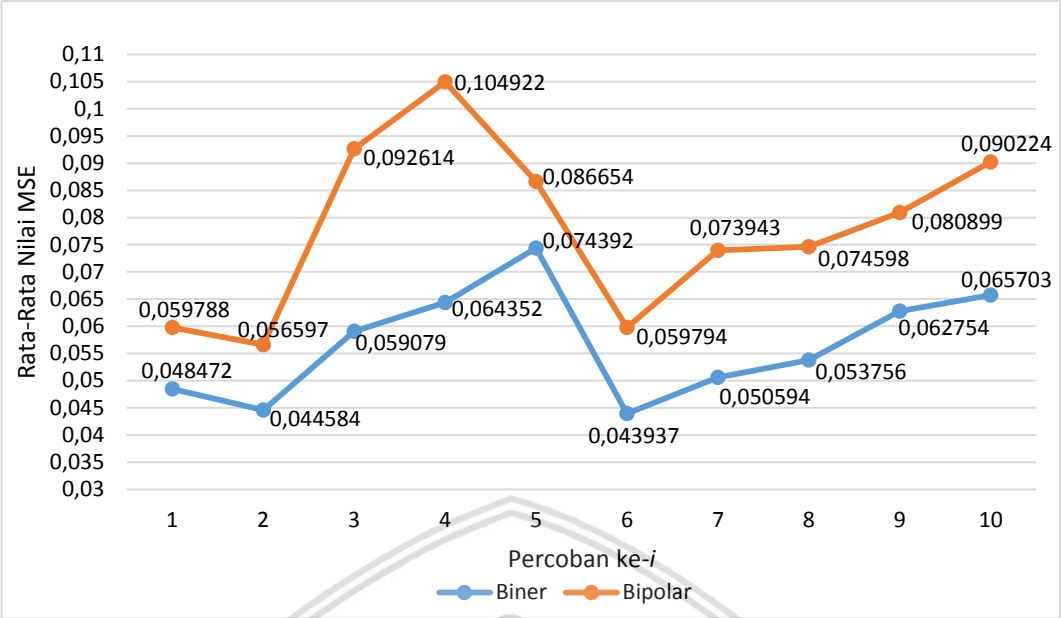
pengujian ini sebanyak 7 dan jumlah *hidden layer* sebanyak 4. Setiap kali program dijalankan, nilai MSE yang dihasilkan dari sistem ELM berbeda-beda. Hal tersebut dikarenakan inisialisasi nilai *input weight* dilakukan secara *random* (nilai *input weight* setiap kali percobaan disertakan pada lampiran B). Dan diakhiri dengan evaluasi rata-rata nilai MSE. Hasil pengujian fungsi aktivasi ditunjukkan pada Tabel 6.3.

Tabel 6.3 Hasil Pengujian Fungsi Aktivasi

Percobaan ke-*i*	Nilai MSE Pada Fungsi Aktivasi	
	Sigmoid Biner	Sigmoid Bipolar
1	0,048472	0,059788
2	0,044584	0,056597
3	0,059079	0,092614
4	0,064352	0,104922
5	0,074392	0,086654
6	0,043937	0,059794
7	0,050594	0,073943
8	0,053756	0,074598
9	0,062754	0,080899
10	0,065703	0,090224
Rata-Rata Nilai MSE	0,056762	0,078003

6.3.2 Analisis Pengujian Fungsi Aktivasi

Berdasarkan Tabel 6.3 diketahui bahwa perolehan rata-rata nilai MSE sebesar 0,056762 yang terletak pada fungsi aktivasi sigmoid biner. Sehingga fungsi aktivasi sigmoid biner tersebut akan digunakan untuk pengujian berikutnya. Hasil rata-rata nilai MSE fungsi aktivasi dapat dilihat pada Gambar 6.3.



Gambar 6.3 Grafik Hasil Pengujian Fungsi Aktivasi

Analisis pengujian fungsi aktivasi bertujuan untuk melakukan transformasi suatu masukan menjadi suatu keluaran tertentu dan mengetahui pengaruh jenis fungsi aktivasi terhadap rata-rata nilai *error* (MSE) yang dihasilkan. Pengujian fungsi aktivasi dievaluasi Berdasarkan grafik pada Gambar 6.3, diperoleh rata-rata nilai MSE terkecil yaitu pada fungsi aktivasi sigmoid biner. Walaupun rata-rata nilai MSE fungsi aktivasi sigmoid biner tidak jauh berbeda dengan rata-rata nilai MSE fungsi aktivasi sigmoid bipolar, akan tetapi fungsi aktivasi sigmoid biner yang memiliki akurasi lebih baik. Hal itu dimungkinkan nilai normalisasi pada penelitian ini memiliki range 0 sampai 1, sehingga nilai yang dihasilkan dari fungsi aktivasi sigmoid biner sangat berpengaruh pada pencapaian target prediksi.

6.4 Pengujian Jumlah *Neuron* pada *Hidden Layer*

Pengujian jumlah *neuron* pada *hidden layer* ini menjelaskan mengenai hasil skenario pengujian jumlah *neuron* pada *hidden layer* dan analisis skenario pengujian jumlah *neuron* pada *hidden layer*.

6.4.1 Skenario Pengujian Jumlah *Neuron* Pada *Hidden Layer*

Pengujian jumlah *neuron* pada *hidden layer* dilakukan untuk memperoleh hasil evaluasi yang lebih baik dengan rata-rata nilai MSE. Ketika program dijalankan, nilai MSE yang dihasilkan berbeda-beda. Hal tersebut dikarenakan nilai bobot atau *input weight* ditentukan secara *random*. Dalam pengujian ini dilakukan pecobaan sebanyak 10 kali dari jumlah dataset sebanyak 68 data. Jumlah *hidden layer* yang akan diuji memiliki jumlah dengan range 1 sampai 7. Jumlah fitur data yang digunakan sebanyak 7. Presentase data yang digunakan adalah perbandingan rasio data *training* dan data *testing* sebesar 80%:20% dengan menggunakan fungsi aktivasi sigmoid biner. Setiap kali program dijalankan, nilai MSE yang dihasilkan dari sistem ELM berbeda-beda. Hal tersebut dikarenakan inisialisasi nilai *input*

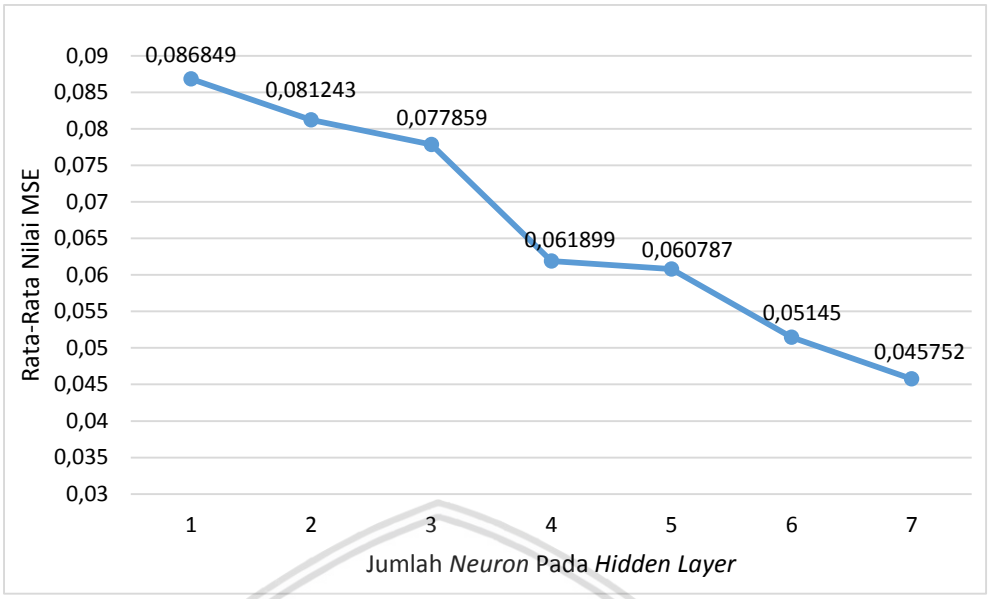
weight dilakukan secara *random* (nilai *input weight* setiap kali percobaan disertakan pada lampiran B). Dan diakhiri dengan evaluasi rata-rata nilai MSE. Tabel 6.4 merupakan hasil pengujian jumlah *neuron* pada *hidden layer*.

Tabel 6.4 Hasil Pengujian Jumlah *Neuron* Pada *Hidden Layer*

Percobaan ke-*i*	Nilai MSE Jumlah *Neuron* Pada *Hidden Layer*						
	1	2	3	4	5	6	7
1	0,075572	0,071898	0,077467	0,057955	0,062902	0,052703	0,04827
2	0,089423	0,081834	0,081322	0,082633	0,056194	0,053952	0,054667
3	0,118990	0,074996	0,089557	0,060255	0,056670	0,063663	0,042626
4	0,070917	0,101173	0,074444	0,067242	0,064291	0,044681	0,043398
5	0,083733	0,087672	0,063605	0,072514	0,061880	0,059554	0,045256
6	0,089441	0,086969	0,082753	0,057991	0,055790	0,048630	0,037662
7	0,076053	0,095137	0,064693	0,064009	0,058333	0,053335	0,046912
8	0,093793	0,074686	0,099627	0,054062	0,068319	0,042199	0,050429
9	0,092641	0,072692	0,067839	0,051453	0,064610	0,051444	0,047403
10	0,077923	0,065377	0,077282	0,050880	0,058878	0,044341	0,040900
Rata-Rata Nilai MSE	0,086849	0,081243	0,077859	0,061899	0,060787	0,051450	0,045752

6.4.2 Analisis Pengujian Jumlah *Neuron* Pada *Hidden Layer*

Berdasarkan Tabel 6.4 diketahui bahwa perolehan rata-rata nilai MSE sebesar 0,045752 yang terletak pada jumlah *Hidden Layer* sebanyak 7. Hasil rata-rata nilai MSE pengujian jumlah *neuron* pada *hidden layer* dapat dilihat pada Gambar 6.4.



Gambar 6.4 Grafik Hasil Pengujian Jumlah Neuron Pada Hidden Layer

Neuron-neuron yang terdapat pada *hidden layer* metode *Extreme Learning Machine* merupakan unit yang melakukan proses perhitungan dengan mengolah *input* menjadi *output*. Di sisi lain, *hidden layer* mempunyai parameter antar layer yang dihubungkan dengan nilai yang berbeda. Hal tersebut memungkinkan adanya sebuah perbedaan hasil yang diperoleh oleh setiap unit *neuron*.

Analisis dilakukan pada pengujian jumlah *neuron* pada *hidden layer* yaitu rata-rata nilai MSE terhadap jumlah *hidden layer*. Dari Gambar 6.4, rata-rata nilai MSE mengalami penurunan yang cukup jauh. Hal ini dapat disimpulkan bahwa semakin banyak masukan jumlah *hidden layer* maka semakin kecil rata-rata nilai MSE. Hasil pengujian diperoleh rata-rata nilai MSE terbaik ditunjukkan pada jumlah *hidden layer* 7 yaitu sebesar 0,045752.

BAB 7 PENUTUP

Bab ini memuat kesimpulan terhadap penelitian yang telah dilakukan berdasarkan hasil perancangan, implementasi dan pengujian sistem serta diberikan saran untuk penelitan selanjutnya.

7.1 Kesimpulan

Berdasarkan hasil yang telah diperoleh, maka kesimpulan yang dapat ditarik dari pengujian dan pembahasan dari prediksi jumlah kriminalitas di Kabupaten Probolinggo menggunakan metode *Extreme Learning Machine* (ELM) adalah sebagai berikut:

1. Metode *Extreme Learning Machine* dapat digunakan dalam prediksi jumlah kriminalitas. Hal tersebut dilakukan dengan melibatkan jumlah *hidden neuron* dan nilai *input weight* yang menghasilkan keluaran dengan perolehan nilai *error* terkecil. Pada *hidden layer* terdiri dari beberapa *node* yang biasa disebut *hidden node* berfungsi sebagai unit pemrosesan yang menghubungkan antara kedua *layer* yaitu *input layer* dan *output layer.* Data masukan pada *input layer* akan terhubung dengan seluruh *hidden node* melalui parameter penghubung berupa *input weight* dan data tersebut akan diolah menjadi keluaran untuk diteruskan ke *layer* selanjutnya. Hasil keluaran dari *hidden node* akan terhubung dengan *node-node* pada *output layer* melalui parameter penghubung berupa *output weight* (β).
2. Perbandingan jumlah data *training* dan data *testing* serta penambahan jumlah fitur berpengaruh terhadap keluaran prediksi yang dihasilkan. Selain itu, nilai random dari *input weight* yang dihasilkan juga berpengaruh terhadap perolehan nilai MSE. Berdasarkan hasil pengujian menggunakan jumlah fitur sebanyak 7, fungsi aktivasi sigmoid biner dan perbandingan data *training* dan data *testing* yaitu 80%:20% diperoleh nilai MSE terendah yaitu sebesar 0,037662 dengan simpangan rata-rata jumlah kriminalitas sebesar 4,2.

7.2 Saran

Berdasarkan penelitian tentang prediksi jumlah kriminalitas menggunakan metode *Extreme Learning Machine* (ELM) dapat diberikan dengan beberapa saran sebagai berikut:

1. Untuk penelitian mendatang, peneliti dapat menambahkan beberapa parameter lain yang merupakan faktor-faktor dalam jumlah kriminalitas. Hal ini bertujuan untuk menghasilkan prediksi yang lebih objektif. Semakin banyak parameter faktor terjadinya tindak kejahatan, maka akan memperoleh hasil yang lebih baik.
2. Untuk penelitian mendatang, peneliti dapat melakukan optimasi bobot (*input weight*) sehingga diperoleh bobot terbaik untuk meningkatkan kemampuan jaringan dalam mengenali pola jumlah kriminalitas serta meningkatkan hasil yang diperoleh dari penelitian sebelumnya.

DAFTAR PUSTAKA

Cipta, S. P., 2016. Penerapan Algoritma Evolving Neural Network Untuk Prediksi Curah Hujan. *JTIULM,* Volume 1, pp. 1-8.

Fardani, D. P., Wuryanto, E. & Werdiningsih, I., 2015. Sistem Pendukung Keputusan Peramalan Jumlah Kunjungan Pasien Menggunakan Metode Extreme Learning Machine (Studi Kasus: Poli Gigi RSU DR. Wahidin Sudiro Husodo Mojokerto). *Journal of Information Systems Engineering and Business Intelligence,* Volume 1, p. 33.

Huang, G. B., Zhu, Q. Y. & Siew, C. K., 2005. Extreme Learning Machine : Theory and applications. *Elsevier science : Neurocomputing,* Volume 70(2006), pp. 489-501.

Huang, G., Zhu, Q. & Siew, C., 2004. *Extreme Learnig Machine : A New Learning Scheme of Feedforward Neural Networks.* Budapest,Hungary: Nayang Avenue.

Hyndman, R. J. & Koehler, A. B., 2005. Another Look at Measures of Forecast Accurancy. *International Journal of Forecasting,* Volume 22, pp. 679-688.

Jain, Y. K. & Bhandare, S. K., 2011. Min Max Normalization Based Data Perturbation Method for Privacy Protection. *International Journal of Computer & Communication Technology,* Volume 2, p. 48.

Kartono, K., 1999. *Patologi Sosial.* Jakarta: Raja Grafindo Persada.

Khotimah, B. K., Sari, E. M. & Yulianarta, H., 2010. Kinerja Metode Extreme Learning Machine (ELM) pada Sistem Peramalan. *Jurnal Ilmiah SimanteC,* Volume 1, p. 186.

Kusumadewi, S., 2003. *Artificial Intelligence (Teknik dan Aplikasinya).* Yogyakarta: Graha Ilmu.

Mahdiyah, U., Irawan, M. I. & Imah, E. M., 2015. Study Comparison Backpropagation, Support Vector Machine, And Extreme Learning Machine For Bioinformatics Data. *Ilmu Komputer dan Informasi,* pp. 53-59.

Mendome, K., Nainggolan, N. & Kekenusa, J., 2016. Penerapan Model ARIMA dalam Memprediksi Jumlah Tindak Kriminalitas di Wilayah POLRESTA Manado Provinsi Sulawesi Utara. *Jurnal MIPA UNSRAT,* Volume 5(2), p. 114.

Munawarah, R., Soesanto, O. & Faisal, M. R., 2016. Penerapan Metode Support Vector Machine Pada Diagnosa Hepatitis. *Kumpulan jurnaL Ilmu Komputer (KLIK).*

Pangaribuan, J. J., 2016. Mendiagnosis Penyakit Diabetes Melitus dengan Menggunakan Metode Extreme Learning Machine. *Jurnal ISD,* Volume 2, p. 32.

Patro, S. G. K. & Sahu, K. K., 2015. Normalization: A Preprocessing Stage. *India: Department of SCE&IT, VSSUT, Burla.*

Pradasari, N. I., Pontia, F. T. & Triyanto, D., 2013. Aplikasi Jaringan Syaraf Tiruan Untuk Memprediksi Penyakit Saluran Pernafasan Dengan Metode Backpropagation. *Jurnal Ilmu Komputer,* pp. p.20-30.

Siang, J. J., 2009. *Jaringan Syaraf Tiruan dan Pemrogramannya Menggunakan Matlab.* Yogyakarta: Andi.

Singh, R. & Balasundaram, S., 2007. Application of Extreme Learning Machine Method for Time Series Analysis. *International Journal of Intelligent Technology,* Volume 2(4), p. 256.