PREDIKSI NILAI *CRYPTOCURRENCY BITCOIN* MENGGUNAKAN ALGORITME *EXTREME LEARNING MACHINE* (ELM)

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:
Rahmat Faizal
NIM: 155150201111040

PROGRAM STUDI TEKNIK INFORMATIKA
JURUSAN TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2019

PENGESAHAN

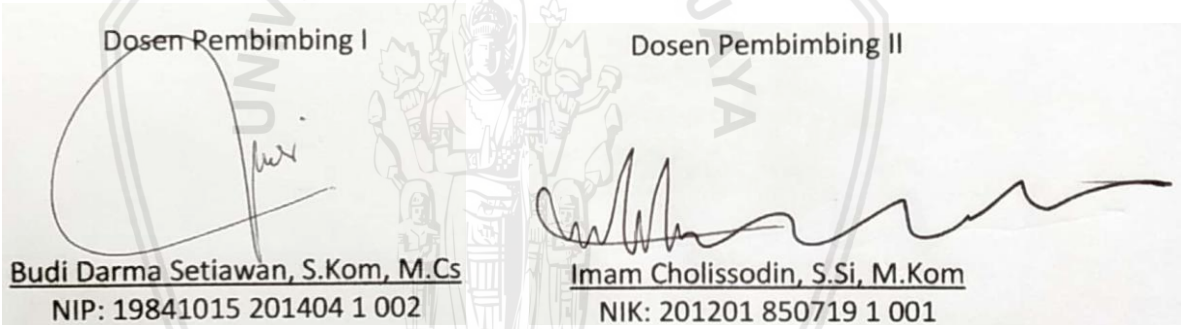
PREDIKSI NILAI *CRYPTOCURRENCY BITCOIN* MENGGUNAKAN ALGORITME *EXTREME LEARNING MACHINE* (ELM)

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh :
Rahmat Faizal
NIM: 155150201111040

Skripsi ini telah diuji dan dinyatakan lulus pada
06 Maret 2019
Telah diperiksa dan disetujui oleh:



Mengetahui
Ketua Jurusan Teknik Informatika

PERNYATAAN ORISINALITAS

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang sepengatuhan saya, di dalam naska 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 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, 06 Februari 2019

Rahmat Faizal

NIM: 155150201111040

PRAKATA

Puji Syukur penulis panjatkan kehadirat Allah SWT yang telah melimpahkan rahmat, hidayah serta karunia-Nya sehingga dapat menyelesaikan skripsi dengan judul "Prediksi Nilai *Cryptocurrency Bitcoin* Menggunakan Algoritme *Extreme Learning Machine* (ELM)". Dalam proses pengerjaan skripsi berupa skripsi ini, penulis tidak lepas dari bantuan, bimbingan, semangat serta doa yang diberikan oleh berbagai pihak kepada penulis. Oleh karena itu, ucapan terima kasih disampaikan kepada:

1. Kepada Tekyun dan Pak Guru selaku kedua orang tua penulis yang selalu memberikan dukungan, semangat, doa selama proses pengerjaan skripsi ini sehinga dapat terselesaikan.
2. Bapak Budi Darma Setiawan, S.Kom, M.Cs selaku Dosen Pembimbing I yang dengan tulus ikhlas membimbing dan membantu penulis dalam menyelesaikan skripsi ini.
3. Bapak Imam Cholisoddin, S.Si, M.Kom selaku Dosen Pembimbing II yang dengan sabar membimbing dan memberikan masukan serta arahan untuk penulis sehingga dapat menyelesaikan skripsi ini.
4. Bapak Wayan Firdaus Mahmudy, S,Si, M.T, Ph.D selaku Dekan Fakultas Ilmu Komputer Universitas Brawijaya Malang.
5. Bapak Heru Nurwarsito, Ir., M.Kom selaku Wakil Dekan I serta Dosen Pembimbing Akademik yang sudah sabar membimbing penulis selama perkuliahan.
6. Bapak Tri Astoto Kurniawan, S.T, M.T, Ph,D selaku Ketua Jurusan Teknik Informatika Fakultas Ilmu Komputer Universitas Brawijaya Malang.
7. Bapak Agus Wahyu Widodo, S.T, M.Cs selaku Ketua Program Studi Teknik Informatika Fakultas Ilmu Komputer Universitas Brawijaya Malang.
8. M. Shidqi Fadlilah, Indriya Dewi Onantya, Dwi Suci Ariska Yanti, Khalisma Frinta, Kevin Azwega, M. Rafi Farhan, Galih Ariwanda, Fadhlillah Ikhsan, serta Faishal Farras Wasito sebagai sahabat serta tempat untuk mencurahkan keluh kesah penulis selama proses perkuliahan.
9. Teman-teman wolu octd (Cia, Aan, Fikri, Iwank, Luthfia, Mustika, Ninis, Dimas, Onne, Bayu, Sayyid, Sara, Tiwi, Vike, Abil) yang sudah menjadi sahabat penulis sejak SMA dan memberikan dukungan serta doa dalam pengerjaan skripsi ini.
10. Teman-temen SPV Raja Brawijaya 2017 yang menjadi wadah saya untuk berproses dan menjadi orang yang lebih baik.
11. Seluruh teman-teman penulis yang telah memberikan kelancaran yang tidak dapat penulis sebutkan satu persatu.

Penulis menyadari bahwa skripsi ini jauh dari kata sempurna, untuk itu penulis mengharapkan kritik dan saran yang bersifat membangun dengan harapan dapat menjadi masukan penulis dan memberikan manfaat bagi pihak yang membutuhkan.

Malang, 06 Februari 2019

Penulis
rhmtfaizal@gmail.com

ABSTRAK

Rahmat Faizal, Prediksi Nilai *Cryptocurrency Bitcoin* Menggunakan Algoritme *Extreme Learning Machine* (ELM)

Pembimbing: Budi Darma Setiawan, S.Kom, M.Cs dan Imam Cholissodin, S.Si, M.Kom

Bitcoin merupakan salah satu bentuk *cryptocurrency* yang dilirik masyarakat karena pengelolaannya yang terdesentraliasi, kerahasiaan yang terjaga, serta prosesnya yang mudah. Namun, mata uang digital ini mengalami fluktuasi yang ekstrim yang membuat beberapa pemilik aset merasa dirugikan. Banyak cara dilakukan untuk mencegahnya seperti melihat pergerakan nilai secara terus menerus yang, melakukan aksi tanpa mempertimbangkan prospek kedepannya, atau bahkan membiarkannya sampai waktu yang ditentukan oleh pemilik aset. Tentu saja hal itu tidak efisien mengingat tujuan utama penyimpanan aset adalah mendapatkan keuntungan. Maka dari itu, diperlukan sistem yang dapat memprediksi nilai *Bitcoin* secara tepat dan akurat sehingga dapat membantu mengurangi kerugian serta menjadi bahan pertimbangan dalam proses jual beli *cryptocurrency Bitcoin*. Penelitian ini memiliki tujuan untuk mendapatkan nilai prediksi *cryptocurrency Bitcoin* menggunakan algoritme *Extreme Learning Machine* (ELM). Berdasarkan hasil implementasi serta analisis yang telah dilakukan menggunakan data *Bitcoin* dari tanggal 1 Mei 2018 sampai dengan 1 Agustus 2018 diperoleh nilai kesalahan terkecil menggunakan *Mean Absolute Percentage Error* (MAPE) sebesar 2,657% dengan jumlah fitur sebanyak 2, jumlah *hidden neuron* sebanyak 4, persentase jumlah data latih sebesar 80%, serta rentang nilai bobot dengan rentang [-1.8, 1.8].

Kata kunci: prediksi, *cryptocurrency bitcoin, extreme learning machine,* regresi*, mean absolute percentage error*

ABSTRACT

Rahmat Faizal, ***Prediction of Cryptocurrency Bitcoin Using Extreme Learning Machine (ELM) Algorithm***

Supervisors: Budi Darma Setiawan, S.Kom, M.Cs dan Imam Cholissodin, S.Si, M.Kom

Bitcoin is one of cryptocurrency which is popular among people due to decentralized management, well-maintained confidentiality, and easy process. But, this type of cryptocurrency is extremely volatile which makes the owner feel aggrieved. Lots of actions have been taken to overcome this by seeing the statistic movement over and over, Taking actions without considering the future prospect, or make the the asset being untouched until the considered time. Those are inneficient regarding the goal is to get the profit.Therefore, the need of system which can predict the value of Bitcoin accurately and efficiently so it can help decreasing the risk of losing and could give another consideration on trading cryptocurrency Bitcoin. This research has a purpose to obtain the value of cryptocurrency Bitcoin using Extreme Learning Machine (ELM) algorithm. Based on the implementation and analysis conducted using Bitcoin Data from May 1th, 2018 until August 1th, 2018, it can be obtained that the smallest error value using Mean Average Percentage Error (MAPE) is 2,657% with the number of features are 2, the number of hudden neuron is 4, and the percentage of training data is 80%, also the range of weights with range [-1.8, 1.8].

Keywords: *prediction, cryptocurrency bitcoin, extreme learning machine, regression, mean, absolute percentage error*

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR KODE PROGRAM

DAFTAR LAMPIRAN

BAB 1 PENDAHULUAN

1.1 Latar Belakang

Kemajuan teknologi serta globalisasi yang makin menjamur membuat segala bidang berkembang pesat, khususnya di bidang ekonomi. Salah satu bukti berkembangnya teknologi di bidang ekonomi adalah dengan diciptakannya *cryptocurrency* atau uang virual yang ada di dunia maya. Berbicara tentang *cryptocurrency*, terdapat banyak sekali ragamnya. Namun, yang paling menjadi sorotan adalah *Bitcoin*. *Bitcoin* merupakan salah satu bentuk *cryptocurrency* yang dibuat oleh Satoshi Nakamoto. Pengelolaan *Bitcoin* sendiri tidak dipegang oleh satu bank pusat atau perusahaan, melainkan aset yang ada dalam *Bitcoin* dikelola oleh pengguna *Bitcoin* yang identitasnya tidak dimunculkan (Velankar et al., 2018). Dengan adanya pengelolaan seperti itu, maka *Bitcoin* menjadi sebuah mata uang digital yang banyak dilirik kalangan masyarakat untuk menyimpan asetnya dalam bentuk *Bitcoin*. Selain pengelolaannya yang tersedentralisasi, kerahasiaan yang terjaga, pembelian *Bitcoin* terbilang mudah karena penghapusan dokumen serta langkah-langkah yang menguras waktu (Pratama, 2017).

Seiring berjalannya waktu, nilai *Bitcoin* mengalami fluktuasi dengan indeks volatilitas mencapai 4,61% (Radityo, Munajat dan Budi, 2017). Hal ini disebabkan karena adanya permintaan dan penawaran yang terjadi antara orang yang menjual serta membeli *Bitcoin*. Jika seseorang menjual seluruh *Bitcoin*nya, maka nilai *Bitcoin* akan turun karena penawaran yang naik. Lalu, jika seseorang ingin membeli banyak *Bitcoin*, maka nilai *Bitcoin* akan naik karena permintaan yang naik. Hal inilah yang membuat fluktuasi nilai *Bitcoin* terkesan sangat ekstrim dan membuat investor yang menyimpan asetnya dalam bentuk *Bitcoin* bingung dalam memutuskan harus menjualnya atau meyimpannya sampai nilai jualnya dapat memberikan keuntungan kepada pemilik aset tersebut (Luno, 2017). Faktor lain yang menyebabkan terjadinya fluktuasi ekstrim ini adalah karena tidak adanya nilai dasar seperti pada mata uang konvensional.

Lalu, dengan fluktuasi yang terjadi pada mata uang digital *Bitcoin* membuat pemilik aset tidak ingin dirugikan dengan fenomena ini. Berbagai cara dilakukan mulai dari melihat statistik pergerakan nilai *Bitcoin* di laman-laman resmi dan terpercaya secara terus menerus, menjual *Bitcoin* yang mereka miliki ketika nilai jual sedang tinggi tanpa mempertimbangkan prospek kedepannya, atau bahkan membiarkan aset mereka tertahan sampai waktu yang ditentukan oleh pemilik aset. Hal tersebut tergolong tidak efisien. Maka dari itu, diperlukan sebuah sistem prediksi yang dapat menyelesaikan permasalahan tersebut. Dengan adanya sistem ini, pemilik aset dapat memprediksi nilai *Bitcoin* yang akan datang serta dapat digunakan sebagai bahan pertimbangan dalam proses jual beli mata uang digital *Bitcoin*.

Untuk melakukan prediksi nilai *cryptocurrency Bitcoin*, banyak metode yang bisa digunakan untuk membangun sistem prediksi ini. Salah satunya adalah yang dilakukan oleh Radityo dkk (2017) yang mana melakukan prediksi menggunakan

algoritme *Backpropagation*. Namun, kelemahan dari algoritme ini adalah kecepatan dalam mempelajari pola data yang lama serta terkadang tidak bisa mengatasi *local minima*. Oleh karena itu, dalam penelitian ini penulis menggunakan algoritme *Extreme Learning Machine* (ELM) yang dapat mengatasi permasalahan pada algoritme *Backpropagation*. Algoritme ini merupakan salah satu algoritme jaringan syaraf tiruan yang mana memiliki keunggulan dari segi kecepatan serta tingkat akurasi. Algoritme ini juga banyak digunakan oleh peneliti untuk melakukan klasifikasi maupun prediksi. Penelitian yang dilakukan oleh Mosabeth dkk (2018) yang melakukan prediksi harga pasar daging sapi menggunakan algoritme *Extreme Learning Machine* (ELM) menggunakan data *time series* didapatkan hasil terbaik yaitu nilai MAPE sebesar 0,344% dengan perbandingan data latih dengan data latih yaitu 90%:10%, rentang bobot antara -1 dan 1, jumlah *hidden neuron* 7, serta menggunakan fungsi aktivasi sigmoid biner dan jumlah fitur 3.

Penelitian selanjutnya yang dilakukan oleh Permatasi dkk (2018) yang melakukan estimasi hasil poduksi benih kenaf yang mana mendapatkan nilai MAPE terbaik sebesar 0,160% dengan jumlah *neuron* pada *hidden layer* sejumlah 8 serta menggunakan fungsi aktivasi *sigmoid biner* dan persentase jumlah data latih dan data uji sebesar 90%:10% (Permatasari dan Rahayudi, 2018). Selanjutnya, penelitian yang dilakukan oleh Jauhari dkk (2018) yang melakukan prediksi nilai tukar rupiah Indonesia terhadap dollar amerika serikat menggunakan *Reccurent Extreme Learning Machine*. Dari penelitian tersebut didapatkan hasil MAPE sebesar 0.069502%.

Berdasarkan penelitian-penelitian yang sudah dilakukan, tantangan yang ada dalam penerapan algoritme *Extreme Learning Machine* (ELM) adalah mendapatkan tingkat kesalahan terkecil dari kombinasi beberapa parameter, yaitu jumlah hari, jumlah *hidden neuron,* jenis fungsi aktivasi, serta perbandingan antara data latih dengan data uji. Oleh karena itu, penulis memutuskan untuk melakukan penelitian dengan judul "Prediksi Nilai *Cryptocurrency Bitcoin* menggunakan Algoritme *Extreme Learning Machine* (ELM)".

1.2 Rumusan Masalah

Berdasarkan penjelasan latar belakang tersebut, maka rumusan masalah adalah sebagai berikut:

1. Berapa jumlah fitur terbaik pada algoritme *Extreme Learning Machine* (ELM) untuk memprediksi nilai *cryptocurrency Bitcoin*?
2. Berapa jumlah *hidden neuron* terbaik pada algoritme *Extreme Learning Machine* (ELM) untuk memprediksi nilai *cryptocurrency Bitcoin*?
3. Berapa rentang bobot masukan terbaik pada algoritme *Extreme Learning Machine* (ELM) untuk memprediksi nilai *cryptocurrency Bitcoin*?

1.3 Tujuan

1. Mendapatkan jumlah fitur terbaik pada algoritme Extreme Learning Machine (ELM) untuk memprediksi nilai cryptocurrency *Bitcoin*.
2. Mendapatkan jumlah hidden neuron terbaik pada algoritme Extreme Learning Machine (ELM) untuk memprediksi nilai cryptocurrency *Bitcoin*.
3. Mendapatkan rentang bobot masukan terbaik pada algoritme Extreme learning Machine (ELM) dalam memprediksi nilai *cryptocurrency Bitcoin*.

1.4 Manfaat

Manfaat yang diperoleh dari penelitian ini adalah mengetahui apakah algoritme ELM dapat diterapkan untuk memprediksi nilai *cryptocurrency Bitcoin* serta menjadi referensi secara akademis kepada peneliti lain.

1.5 Batasan masalah

Untuk menghindari meluasnya masalah, maka diberikan batasan masalah sebagai berikut:

1. Data yang digunakan dalam penelitian ini adalah data nilai *crytocurrency Bitcoin* yang berasal dari laman investing.com perhari mulai dari 1 Mei 2018 sampai 1 Agustus 2018.
2. Data nilai *cryptocurrency Bitcoin* yang diambil ada data penutupan nilai *cryptocurrency* perhari.
3. Perhitungan tingkat kesalahan prediksi pada penelitian ini menggunakan perhitungan MAPE (*Mean Absolute Percentage Error*).

1.6 Sistematika Pembahasan

Sistematika penyusunan proposal ini ditujukan untuk memberikan gambaran dan uraian dari laporan proposal secara garis besar meliputi beberapa bab sebagai berikut:

BAB 1 PENDAHULUAN

Pada bab pendahuluan menguraikan mengenai latar belakang, rumusan masalah, tujuan, manfaat, serta sistematika penyusunan laporan sistem rekomendasi pemilihan jenis kendaraan roda empat.

BAB 2 LANDASAN KEPUSTAKAAN

Pada bab ini menguraikan teori-teori yang menjadi referensi dalam penyusunan laporan proposal projek akhir. Pada bagian landasan kepustakaan terdiri atas dua bagian, yaitu kajian pustaka yang memuat penelitian-penelitian sebelumnya yang pernah ada dan pada bagian kedua berisi tentang dasar teori yang digunakan untuk memperkuat latar belakang pada Bab 1.

BAB 3 METODOLOGI

Menguraikan tentang metodologi dan langkah kerja yang dilakukan dalam proses perancangan dan implementasi sistem. Tahapan alur metodologi meliputi studi literatur, pengumpulan data, analisis dan perancangan, implementasi, pengujian, dan penarikan kesimpulan.

BAB 4 PERANCANGAN

Menjelaskan secara rinci algoritme yang digunakan mulai dari diagram alir per proses, perhitungan manual, perancangan antarmuka, serta rencana pengujian.

BAB 5 IMPLEMENTASI

Mengimplementasikan dasar teori yang telah dipelajari dan analisis perancangan yang telah dilakukan pada tahapan sebelumnya.

BAB 6 PENGUJIAN DAN ANALISIS

Menguraikan hasil pengujian dan analisis hasil implementasi sistem yang telah dilakukan. Pengujian yang dilakukan meliputi pengujian fungsional dan pengujian akurasi.

BAB 7 PENUTUP

Menguraikan kesimpulan yang telah diperoleh dari pembutuan dan pengujian perangkat lunak yang dikembangkan dalam proposal ini serta saran-saran untuk pengembangan lebih lanjut.

BAB 2 LANDASAN KEPUSTAKAAN

2.1 Kajian Pustaka

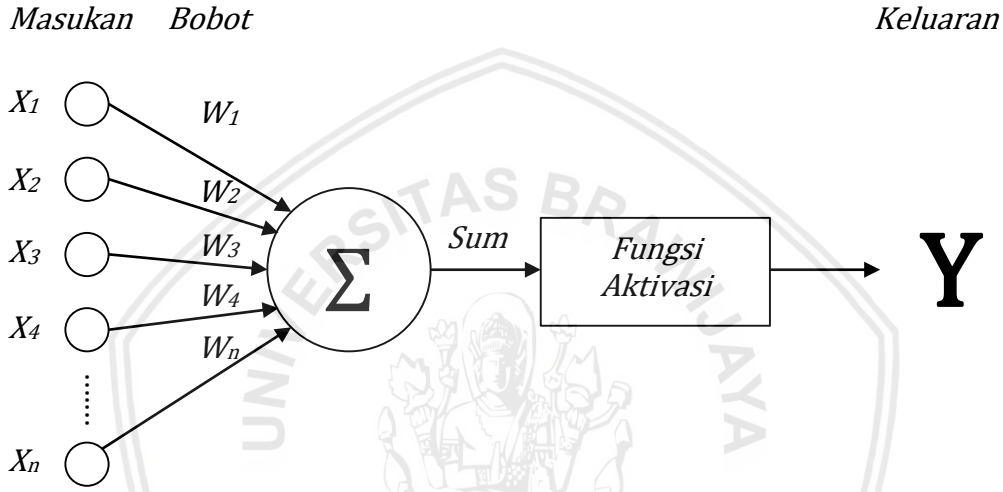
Tabel 2.1 Perbandingan Metode Penelitian

No	Judul	Objek	Metode	Keluaran
		Parameter	Proses	Hasil Penelitian
1	Ship Rolling Motion Prediction Based on Extreme Learning Machine (Huixuan, Yuchao dan Hongmei, 2015)	Data historis percobaan model gulir kapal	*Extreme Learning Machine*	- Hasil Prediksi - Nilai MSE
		Data sudut gulir	- Proses *Training* - Proses *Testing* - Perhitungan MSE	- Nilai MSE terkecil adalah $5.4743 * 10^{-5}$ pada percobaan sudut 150^{o}
2	Estimasi Hasil Produksi Benis Tanaman Kenaf (*Hibiscus Cannabinus L.*) Menggunakan *Metode Extreme Learning Machine (*ELM) Pada Balai Penelitian Tanaman Pemanis dan Serat (Balittas) (Permatasari dan Rahayudi, 2018)	Data karakterisasi tanaman kenaf tahun 2013 sebanyak 100 data.	*Extreme Learning Machine* (ELM)	- Hasil estimasi - Nilai evaluasi sistem
		Data karakterisasi tanaman kenaf denan parameter umur tanaman saat berbunga, diameter batang bagian bawah, jumlah kapsul masak, dan berat benih 10 tanaman.	- Normalisasi data - Proses *Training* - Proses *Testing* - Denormalisasi data - Evaluasi sistem	- Nilai rata-rata MAPE terbaik sebesar 0,160%
3	Prediksi Nilai Tukar Rupiah Indonesia Terhadap Dolar Amerika Serikat Menggunakan Metode Reccurent Extreme Learning Machine Neural Network (Jauhari, Cholissodin dan Dewi, 2017)	Data nilai tukar rupiah terhadap dolar amerika serikat mulai tanggal 1 Januari 2009 sampai 11 Maret 2017.	*Reccurent Extreme Learning Machine Neural Network*	Hasil Prediksi
		- Data nilai tukar rupiah terhadap dolar amerika serikat - Jumlah fitur - Jumlah *hidden neuron* - Jumlah *context neuron*	- Normalisasi data - Proses *Traning* - Proses *Testing* - Hasil Prediksi	Nilai MAPE yang didapat sebesar 0.069502%

4	Prediksi Jumlah Kriminaitas Menggunakan Metode Extreme Learning Machine (Studi Kasus Di Kabupaten Probolinggo) (Dewi, Cholissodin dan Santoso, 2018)	Data sekunder krinimalitas Kabupaten Probolinggo	*Extreme Learning Machine* (ELM)	Hasil Prediksi Jumlah Kriminalitas
		Jumlah fitur berdasarkan masukan pengguna dari data historis kriminalitas Kabubaten Probolinggo dari tahun 2012-2017	- Normalisasi - Inisialisasi *input weight* - *Training* - *Testing* - Perhitungan nilai MSE - Denormalisasi - Hasil prediksi	Diperoleh arsitektur jaringan maksimum dengan rincian: - Jumlah fitur sebanyak 7 - Perbandingan rasio data yaitu 80%:20% - Jumlah *hidden neuron* sebanyak 7 - Fungsi aktivasi sigmoid biner
5	Prediksi Indeks Harga Konsumen (IHK) Kelompok Perumahan, Air, Listrik, Gas, dan Bahan Bakar Mengunakan Metode *Extreme Learning Machine* (Fitriyani, Setiawan dan Perdana, 2018)	Data Indeks Harga Konsumen (IHK) Kelompok Perumahan, Air, Listrik, Gas, dan Bahan Bakar	*Extreme Learning Machine* (ELM)	Hasil prediksi dan nilai RMSE
		Jumlah fitur berdasarkan masukan pengguna dari data *time series* periode 2011-2017	- Normalisasi Data - Pelatihan - Pengujian - Denormalisasi - Menghitung nilai evaluasi - Hasil prediksi dan RMSE	Nilai RMSE minimum sebesar 0,72 dengan rincian: - Jumlah fitur 7 - Jumlah data latih 30 dan data uji 11 - Jumlah hidden neuron 7 - Fungsi aktivasi sigmoid biner

2.1.1 Jaringan Syaraf Tiruan

Jaringan Syaraf Tiruan (JST) merupakan model komputasi yang diadaptasi cara kerja otak manusia seperti kemampuannya untuk belajar. Namun, cara kerja JST jauh lebih sederhana daripada cara kerja otak manusia. JST ini memiliki kesamaan dengan manusia yang memiliki beberapa *neuron* yang berkaitan satu sama lain. *Neuron-neuron* tersebut menyalurkan informasi yang diterima oleh satu *neuron* ke *neuron* yang lain. Informasi yang dibawa oleh *neuron* ini disimpan pada suatu nilai tertentu yang disebut bobot. Untuk lebih lanjut mengenai struktur *node* atau *neuron* di jaringan syaraf tiruan ini ditunjukkan pada Gambar 2.1 berikut ini.



Gambar 2.1 Struktur Jaringan Syaraf Tiruan

Dalam struktur jaringan syaraf tiruan ini, terdapat lapisan masukan (*input layer*) sejumlah n yang terdiri dari beberapa *neuron* yaitu $X_1, X_2, X_3,\ldots,X_n$. Lalu, masing-masing masukan yang memiliki nilai akan dikalikan dengan bobot (*weight*) $W_1, W_2, W_3,\ldots,W_n$ yang sudah diinisialisasi sebelumnya dengan rentang nilai yang sudah ditentukan. Setelah penjumlahan masing-masing masukan dan bobot telah dilakukan, maka proses selanjutnya adalah menjumlahkan seluruh *neuron* dari hasil penjumlahan tersebut dan dimasukkan kedalam sebuah fungsi aktivasi untuk menentukan nilai ambang (*threshold*) *neuron* tersebut. Untuk meneruskan informasi yang didapatkan, nilai masukan hasil aktivasi harus dapat melewati nilai ambang tertentu sehingga dapat diteruskan menjadi keluaran untuk dikirim ke *neuron* lain.

2.1.2 Fungsi Aktivasi Sigmoid Biner

Fungsi aktivasi merupakan sebuah fungsi untuk menentukan nilai keluaran sebuah *neuron*. Dalam algoritme *Extreme Learning Machine* (ELM). Fungsi aktivasi digunakan baik dalam proses *training* maupun proses *testing*. Terdapat banyak fungsi aktivasi yang dapat digunakan dalam algorite *Extreme Learning Machine*

(ELM). Salah satunya adalah fungsi aktivasi sigmoid biner yang menghailkan keluaran dalam rentang 0 sampai 1. Dibawah ini merupakan rumus persamaan fungsi aktivasi sigmoid biner.

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

Keterangan:

$f(x)$ = Fungsi aktivasi sigmoid biner.

x = Nilai matriks H_{init} yang sudah ditambah bias.

e = Bilangan euler dengan nilai pembulatan 2,718.

2.2 *Cryptocurrency Bitcoin*

Bitcoin merupakan suatu mata uang digital (*cryptocurrency*) yang diciptakan oleh seseorang yang tidak diketahui identitasnya namun mengatasnamakan dirinya Satoshi Nakamoto. Berbeda dengan mata uang konvensional, *Bitcoin* tidak membutuhkan perantara dalam setiap transaksi yang dilakukan. Dalam artian, *Bitcoin* tidak diawasi dan diatur oleh bank maupun pemerintah sehingga tidak dikekang oleh regulasi apapun dari negara manapun. Untuk mendapatkan *Bitcoin* sendiri dapat dilakukan dengan melakukan pembelian (*exchange*) melalui *marketplace-marketplace* yang tersedia di pasaran. Selain itu, mendapatkan *Bitcoin* dapat dilakukan dengan melakukan penambnagan (*mining*) yang membutuhkan komputer berspesifikasi tinggi untuk memecahkan permasalahan matematika kompleks. Setelah kita berhasil memecahkan permasalahan tersebut, kita akan dihadiahi dengan beberapa *Bitcoin* (Yekkin, Aratari dan Pagliery, 2018).

Statistik menunjukkan bahwa selama tahun 2017, *Bitcoin* telah menguat lebih dari 1000%. Dan tentu saja hal ini diikuti dengan meningkatnya jumlah pengguna yang menyimpan asetnya dalam bentuk *Bitcoin* (Setiawan, 2017). Seperti halnya mata uang, *Bitcoin* memiliki karakteristik yang sama dengan mata konvensional pada umumnya yaitu fluktuasi nilai tukar. Fluktuasi ini terjadi karena adanya permintaan dan penawaran antara orang yang menjual dan membeli *Bitcoin*. Namun, yang menjadi perbedaan disini adalah fluktuasi yang terjadi pada *Bitcoin* terkesan ekstrim karena tidak adanya sistem yang mengontrol harga *Bitcoin*. Selain itu, nilai dasar juga menjadi faktor lain yang menyebabkan terjadinya fluktuasi ini. Faktor lain yang mempengaruhi adalah spekulasi pemilik aset untuk menjual atau membeli *Bitcoin*. Selain spekulasi pemilik aset, volume dan persebaran *Bitcoin* dan penipuan yang melibatkan *Bitcoin* menjadi faktor lain yang menyebabkan fluktuasi *Bitcoin* dinilai ekstrim (Admin, 2017).

2.3 Prediksi

Dalam menyelesaikan permasalahan yang akan datang dan tidak dapat dipastikan, biasanya dapat diselesaikan dengan model pendekatan. Model pendekatan yang dimaksud adalah model yang sesuai dengan perilaku atau pola-pola dari data yang aktual, sehingga dapat dilakukan prediksi. Prediksi menghasilkan suatu nilai yang menggambarkan kondisi suatu entitas berdasarkan

data aktual. Prediksi diperlukan untuk memperkirakan bagaimana keadaan yang akan datang sehingga dapat dijadikan suatu acuan dalam pengambilan keputusan serta perencanaan kedepannya. Prediksi sendiri di dalamnya terdapat klasifikasi dan regresi. Klasifikasi yang dimaksud adalah penggolongan suatu entitas ke dalam kelompok-kelompok tertentu sesuai dengan standar tertentu. Salah satu contoh klasifikasi adalah mendiagnosa penyakit seorang pasien termasuk dalam penyakit apa berdasarkan parameter-parameter yang mempengaruhi. Selain klasifikasi, terdapat juga regresi yang dapat digunakan untuk melakukan prediksi berdasarkan hubungan antara 2 parameter atau lebih. Regresi dapat melakukan prediksi untuk mendapatkan suatu nilai yang menggambarkan kondisi yang akan dating berdasarkan parameter-parameter yang mempengaruhi.

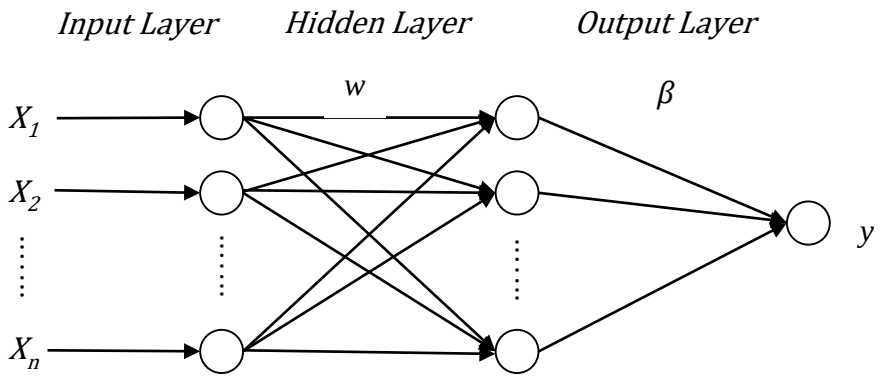
Pada penelitian ini, penulis menggunakan regresi untuk melakukan prediksi. Hal ini dikarenakan penelitian ini bertujuan untuk mendapatkan nilai secara numerik pada masa yang akan datang. Dalam prediksi secara regresi terdapat dua metode yang dapat digunakan, yaitu metode *time series* dan *causal*. Pada metode time series, regresi dilakukan menggunakan data dengan rentang waktu tertentu. Contohnya adalah melakukan prediksi nilai tukar rupiah terhadap dollar amerika serikat berdasarkan data pada tahun 2015-2018. Sementara untuk metode *causal*, regresi dilakukan perdasarkan pola hubungan sebab-akibat antara variabel yang diprediksi dengan variabel yang mempengaruhi. Contohnya adalah menentukan jumlah barang yang cacat berdasarkan rata-rata suhu ruangan per harinya.

Selain itu, prediksi pada dasarnya memiliki jangka waktu, yaitu jangka pendek, menengah, dan panjang. Untuk jangka pendek dilakukan dalam jangka waktu kurang dari satu tahun dan biasanya digunakan dalam prediksi konsumen, curah hujan, dan tingkat produktivitas. Lalu untuk prediksi jangka menengah dilakukan dalam rentang waktu tiga bulan sampai tiga tahun. Biasanya jenis prediksi ini digunakan untuk perencanaan anggaran modal dan pembelian produk baru. Dan untuk prediksi jangka panjang, dilakukan dalam jangka waktu tiga tahun keatas.

2.4 Algoritme *Extreme Learning Machine* (ELM)

Algoritme *Extreme Learning Machine* (ELM) merupakan algoritme *feedforward* dan tergolong baru dalam jaringan syaraf tiruan yang seringkali disebut sebagai *Single Layer Feedforward Neural Networks* (SLFNs). Kecepatan pembelajaran terhadap data pada algoritme ini ribuan kali lebih cepat daripada algoritme lain dan mendapatkan hasil generalisasi yang lebih baik. Selain itu, algoritme ini cenderung menghasilnya nilai error yang kecil (Huang, Zhu dan Siew, 2006).

Pada algoritme ini, terdapat *input layer, hidden layer*, serta *output layer* yang dimana masing-masing memiliki *neuron-neuron* yang terhubung satu sama lain. Penggunaan bobot dan bias yang dibangkitkan secara acak diperlukan oleh algoritme ini untuk mempercepat proses pembelajaran (Sun et al., 2008). Selain mempercepat proses pembelajaran, pembangkitan acak ini dilakukan untuk menghindari hasil prediksi yang tidak stabil. Arsitektur ELM ditunjukan pada Gambar 2.2.



Gambar 2.2 Struktur Algoritme ELM

Berdasarkan Gambar 2.2, dapat dikatakan bahwa proses algoritme ini berawal dari nilai *input* diolah dengan bobot dan bias yang berikutnya diproses setiap *neuron* yang pada akhirnya akan menghasilnya nilai akhir pada *output layer*.

2.4.1 Min-Max *Normalization*

Normalisasi adalah suatu proses untuk menyamakan satuan pada data yang memiliki rentang nilai yang luas serta variasi data yang beragam. Dengan normalisasi, akan didapatkan data yang memiliki satuan yang sama dengan nilai yang kecil serta terbebas dari *outlier*. Proses normalisasi data dalam penelitian ini menggunakan metode Min-Max *Normalization*. Di bawah ini merupakan persamaan Min-Max *Normalization*.

$$x' = \frac{x-min}{max-min} \tag{2.4}$$

Keterangan:

x' = Nilai hasil normalisasi data

x = Nilai asli data

min = Nilai minimum dari dataset

max = Nilai maksimum dari dataset

2.4.2 Proses *Training*

Dalam ELM, terdapat proses pembelajaran (*training*) yang bertujuan untuk melatih sistem sehingga menghasilkan prediksi yang akurat. Berikut ini adalah tahapan-tahapan dalam proses *training* (Liang et al., 2006).

1. Bangkitkan bobot dan bias secara acak dengan rentang yang sudah ditentukan.
2. Mendapatkan nilai *output hidden layer* (*H*) menggunakan fungsi aktivasi yang telah ditentukan yang berbentuk matriks. Untuk mendapatkanya, kita perlu mendapatkan $H_{init.}$ Berikut ini adalah rumus persamaan untuk mendapatkan $H_{init.}$

$$H_{init} = X * Wt \quad (2.5)$$

Keterangan:

H_{init} = Nilai dari hasil perkalian matriks X dengan Wt.

X = Nilai dari data.

Wt = Nilai dari bobot yang di *transpose.*

3. Setelah mendapatkan *Hinit*, langkah selanjutnya adalah adalah mendapatkan *output hidden layer* (*H*) dengan menggunakan fungsi aktivasi. Untuk fungsi aktivasi, dalam persamaan ini digunakan fungsi aktivasi sigmod biner sebagai contoh.

$$H = \frac{1}{1+e^{(-(Hinit+bias))}} \quad (2.6)$$

Keterangan:

H = Nilai *ouput hidden layer.*

e = Bilangan euler dengan nilai pembulatan 2,718.

$Hinit$ = Nilai Hinit yang akan dimasukkan kedalam fungsi aktivasi.

$bias$ = Nilai bias yang dibangkitkan secara acak sesuai jumlah fitur.

4. Menghitung matriks *moore-penrose pseudo inverse.* Berikut ini merupakan rumus persamaan untuk mendapatkan matriks *moore-penrose pseudo inverse.* Dalam algoritme ini, matriks *moore-penrose pseudo inverse* digunakan karena kemampuannya untuk mendapatkan nilai *inverse* dari matriks yang memiliki ordo berbeda (Barata dan Hussein, 2012).

$$H^+ = (H^T * \ H)^{-1} * H^T \quad (2.7)$$

Keterangan:

H^+ = Matriks *moore-penrose pseudo inverse* dengan fungsi aktivasi.

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

5. Menghitung nilai *output weight.* Berikut ini merupakan persamaan untuk menghitung *output weight*.

$$\beta = \ H^+ * T \quad (2.8)$$

Keterangan:

β = Nilai matriks *output weight*.

H^+ = Nilai matriks *moore-penrose pseudo inverse.*

T = Nilai matriks target.

6. Menghitung nilai *output layer* pada proses *training*. Berikut ini adalah persamaan untuk menghitungnya.

$$Y = H \ * \ \beta \quad (2.9)$$

Keterangan:

Y = Nilai matriks *output layer*.

H = Nilai *output hidden layer* menggunakan fungsi aktivasi.

β = Nilai matriks *output weight*.

2.4.3 Proses *Testing*

Tujuan dari proses ini adalah untuk memvalidasi nilai prediksi sekaligus mengevaluasi algoritme ELM dari proses *training* yang sudah dilakukan sebelumnya. Langkah-langkahnya kurang lebih sama dengan proses *training*, yang membedakan adalah tidak adanya perhitungan matriks *moore-penrose pseudo inverse* serta tidak adanya perhitungan *output weight*. Berikut ini adalah langkah-langkah pada proses *testing.*

1. Membangkitkan bobot dan bias secara acak.
2. Menghitung matriks H_{init} menggunakan Persamaan 2.5..
3. Menghitung matriks *H* menggunakan fungsi aktivasi yang sudah ditentukan sesuai dengan Persamaan 2.6.
4. Menghitung nilai *output layer* pada proses *testing* menggunakan Persamaan 2.9.

2.4.4 Proses Denormalisasi Data

Proses denormalisasi data ini bertujuan untuk mendapatkan nilai aktual hasil prediksi. Berikut ini merupakan persamaan untuk mendapatkannya.

$$x = x' * (max - min) + min \quad (2.10)$$

Keterangan:

x = Nilai hasil prediksi setelah denormalisasi.

x' = Nilai hasil prediksi sebelum denormalisasi.

min = Nilai minimum dari dataset

max = Nilai maksimum dari dataset

2.4.5 *Mean Absolute Percentage Error* (MAPE)

Mean Absolute Percentage Error (MAPE) merupakan suatu metode evaluasi yang sangat umum digunakan untuk mengukur seberapa tepat atau akurat suatu prediksi. MAPE menjadi umum karena kemampuannya untuk mempresentasikan nilai *error* yang mudah dipahami dibandingan dengan metode yang lain (Kim dan Kim, 2016).

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{\hat{y}_i - y_i}{y_i} * 100\right| \quad (2.11)$$

Keterangan:

$\hat{y}_i$ = Hasil prediksi.

y_i = Nilai aktual.

n = banyaknya data yang diuji.

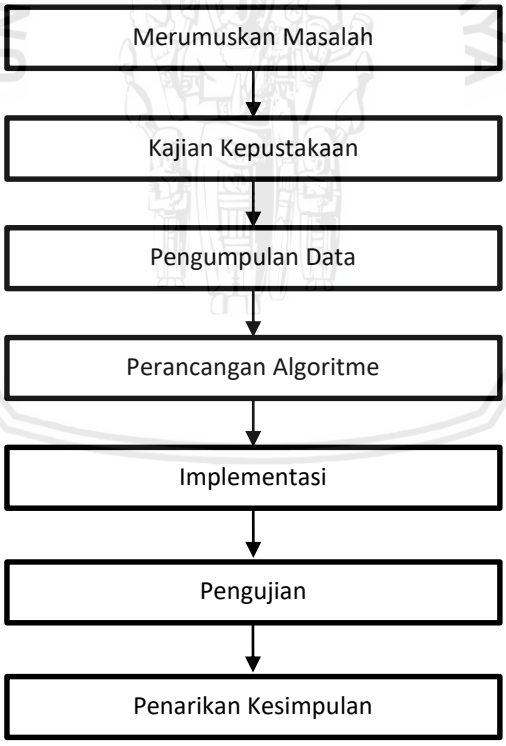
BAB 3 METODOLOGI

3.1 Tipe Penelitian

Penelitian ini bersifat non-implementatif yang menekankan pada pengamatan fenomena atau kondisi-kondisi tertentu, analisis hubungan antar fenomena yang diamati untuk mendapatkan hasil penelitian sebagai produk utamanya. Pada penelitian ini, proses eksperimen dilakukan untuk mendapatkan produk/artefak. Sementara itu, pendekatan yang dilakukan dalam penelitian ini adalah pendekatan deskriptif yang menjelaskan mengenai karakteristik objek penelitian berdasarkan fenomena yang diteliti berdasarkan hasil analisis data yang diperoleh. Produk utama yang dihasilkan adalah hasil investigasi.

3.2 Tahapan Penelitian

Dalam melakukan sebuah penelitian, terdapat tahapan-tahapan yang harus dilakukan secara urut. Tujuannya adalah agar penelitian kita tidak mengalami kesulitan pada saat pelaksanaan penelitian. Tahapan penelitian harus dirancang sebaik mungkin. Oleh karena itu, penulis merancang tahapan penelitian yang dilakukan pada penulisan skripsi ini pada Gambar 3.1.



Gambar 3.1 Tahapan Penelitian

3.3 Studi Kepustakaan

Pada fase ini, studi kepustakaan dilaksanakan untuk mendapatkan landasan teori yang akan digunakan sebagai sumber dalam penulisan skripsi, serta berperan

sebagai dasar untuk mengimplementasikan Algoritme *Extreme Learning Machine* (ELM) untuk prediksi nilai *cryptocurrency Bitcoin*, yang mana komponen-komponennya antara lain:

1. Jaringan Syaraf tiruan
2. Algoritme *Extreme Learning Machine* (ELM).
3. *Cryptocurrency Bitcoin*.
4. *Mean Average Percentage Error* (MAPE).

Literatur serta entitas pendukung dari penelitian ini berupa buku, jurnal, skripsi, dan laman *website*.

3.4 Analisis Kebutuhan Sistem

Analisis kebutuhan sistem adalah tahap untuk menganalisis apa saja yang dibutuhkan oleh sebuah sistem sebelum dapat diimplementasikan ke dalam sebuah kode program. Spesifikasi lingkungan implementasi yang digunakan dalam pembuatan sistem ini sebagai berikut.

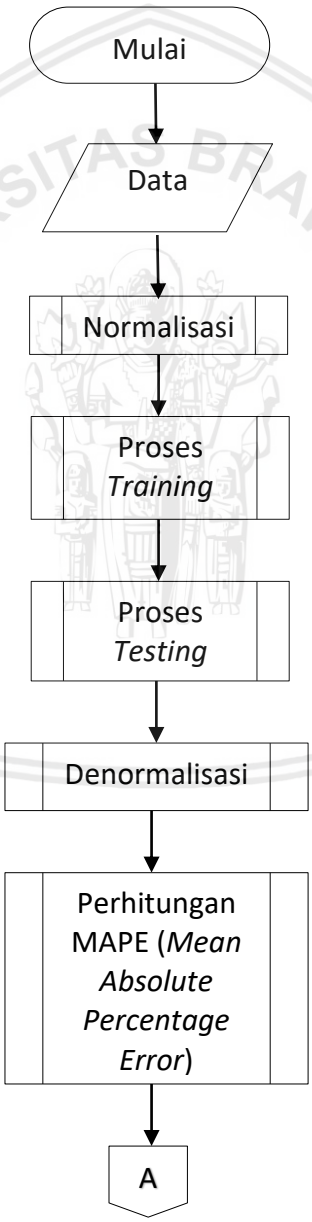
1. Kebutuhan perangkat keras (*hardware*) meliputi:
   - *Laptop* dengan *processor* minimal *intel core i5 8th Generation*.
   - RAM 8GB.
   - Monitor dengan bentang layar 14".
2. Kebutuhan perangkat keras (*software*) meliputi:
   - Sistem operasi *Windows* 10.
   - *Microsoft Office* 2013.
   - *Google Chrome Browser*.
   - *Visual Studio* 2017.
3. Kebutuhan data, yaitu data pergerakan harian nilai tukar *Bitcoin* yang berasal dari laman investing.com.

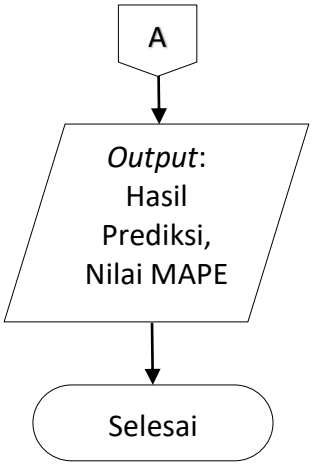
3.5 Pengumpulan Data

Untuk pengumpulan data, data yang diambil berasal dari data pergerakan harian nilai tukar *Bitcoin* dari laman investing.com. Data ini merupakan data *time series* harian dari tanggal 1 Mei 2018 sampai dengan 1 Agustus 2018.

3.6 Perancangan Algoritme

Perancangan algoritme dilakukan untuk mengidentifikasi komponen-komponen yang dibutuhkan serta perhitungan secara manual berdasarkan data yang digunakan saat proses implementasi prediksi nilai *cryptocurrency Bitcoin* menggunakan algoritme *Extreme Learning Machine* (ELM). Dengan adanya perhitungan secara manual, akan didapatkan langkah-langkah penyelesaian masalah dengan menggunakan algoritme *Extreme Learning Machine* (ELM). Untuk penerapannya, terdapat beberapa tahapan yang harus dilakukan yang terdapat pada Gambar 3.2.





Gambar 3.2 Diagram alir proses prediksi

3.7 Implementasi

Implementasi aplikasi dilakukan dengan menggunakan bahasa pemrograman C# dan alat-alat pendukung lainnya agar sesuai dengan perancangan yang telah diidentifikasikan sebelumnya.

3.8 Pengujian

Pengujian dilakukan untuk mengetahui apakah sistem berjalan dengan baik dan sesuai dengan spesifikasi kebutuhan yang telah ditetapkan sebelumnya. Pengujian dilakukan meliputi pengujian fungsional sistem dan pengujian akurasi.

3.9 Kesimpulan

Kesimpulan dilkaukan setelah semua tahapan yang dijelaskan telah selesai dilkaukan. Kesimpulan diambil dari hasil pengujian dan analisis metode. Tahap akhir dari penulisan adalah saran yang dimaksudkan untuk memperbaiki kesalahan yang terjadi serta memberikan pertimbangan untuk pengembangan selanjutnya.

BAB 4 PERANCANGAN

4.1 Formulasi Permasalahan

Pada penelitian ini, penulis ingin menyelesaikan permasalahan dalam memprediksi nilai *cryptocurrency Bitcoin* menggunakan algoritme (ELM). Masukan yang digunakan untuk sistem prediksi ini adalah data historis nilai *cryptocurrency Bitcoin* berdasarkan penutupan pasar perharinya yang didapatkan dari lama investing.com. Data historis tersebut diambil selama 3 bulan terhitung dari tanggal 1 Juni 2018 sampai tangal 1 Agustus 2018. Parameter yang digunakan dalam perhitungan ini adalah jumlah hari sebelumnya, fungsi aktivasi serta jumlah *hidden neuron*. Dalam penelitian ini, normalisasi dilakukan baik pada data latih maupun data uji menggunakan Min-Max *Normalization*. Untuk menghitung tingkat kesalahan berdasarkan hasil prediksi, harus dilakukan proses denormalisasi sehingga dapat dilakukan perhitungan tingkat kesalahan menggunakan *Mean Absolute Percentage Error* (MAPE). Keluaran yang didapat dari hasil prediksi ini adalah nilai MAPE dan hasil prediksi. Data sampel historis nilai *cryptocurrency Bitcoin* ditunjukkan pada Tabel 4.1.

Tabel 4.1 Data Sampel Historis Nilai *Cryptocurrency Bitcoin*

Tanggal	Nilai
1 Juni 2018	105690000
2 Juni 2018	106102000
3 Juni 2018	107510000
4 Juni 2018	106566000
5 Juni 2018	107095000
6 Juni 2018	107453000

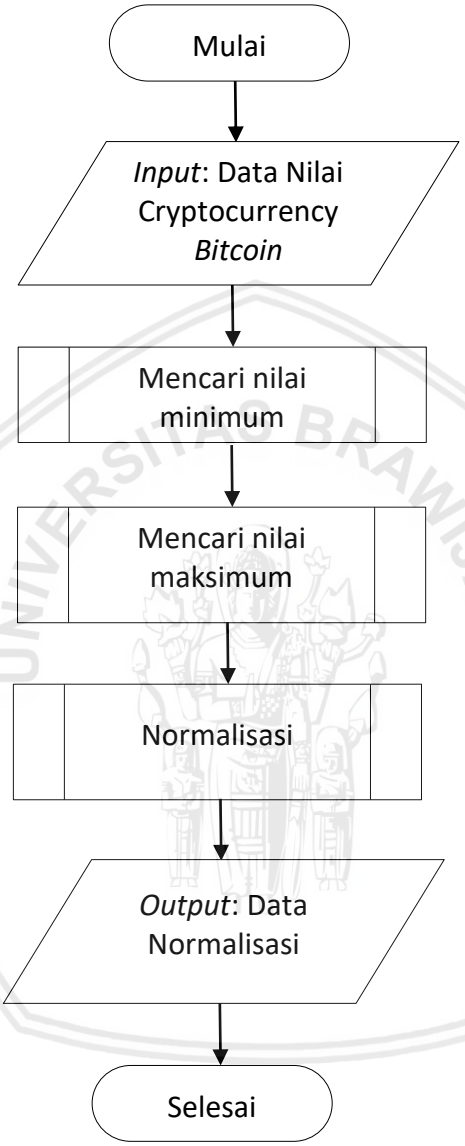
4.2 Siklus Algoritme *Extreme Learning Machine* (ELM)

Siklus algoritme ELM merupakan langkah-langkah penyelesaian masalah secara urut. Pada sub bab berikutnya, akan dijelaskan mengenai siklus algoritme yang meliputi proses prediksi menggunakan algoritme ELM serta pemaparan mengenai proses normalisasi, denormalisasi, dan proses menghitung tingkat kesalahan menggunakan metode MAPE (*Mean Absolute Percentage Error*).

4.2.1 Normalisasi Data

Normalisasi data merupakan sebuah proses yang dilakukan untuk mendapatkan data yang memiliki rentang yang sama. Hal ini dilakukan karena data yang digunakan sebagai nilai masukan memiliki satuan yang tidak sama, seperti puluhan hingga milyaran. Oleh karena itu, normalisasi dilakukan untuk

mentransformasikan nilai data menjadi lebih kecil antara rentang 0 sampai dengan 1. Pada penelitian ini, normalisasi data dilakukan menggunakan metode Min-Max Normalization. Diagram alir proses normalisasi ditunjukkan pada Gambar 4.1 sebagai berikut.



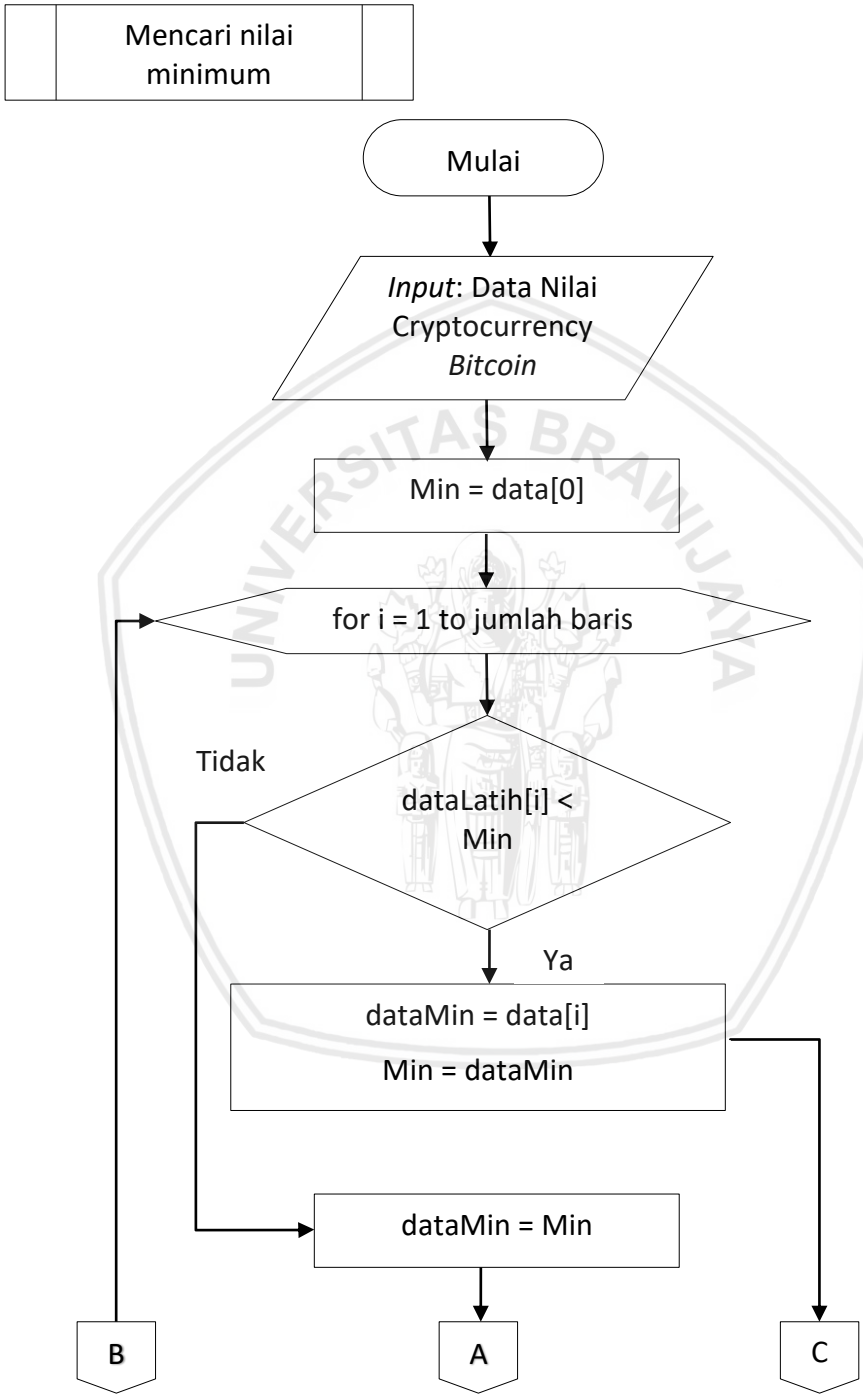
Gambar 4.1 Diagram Alir Normalisasi Data

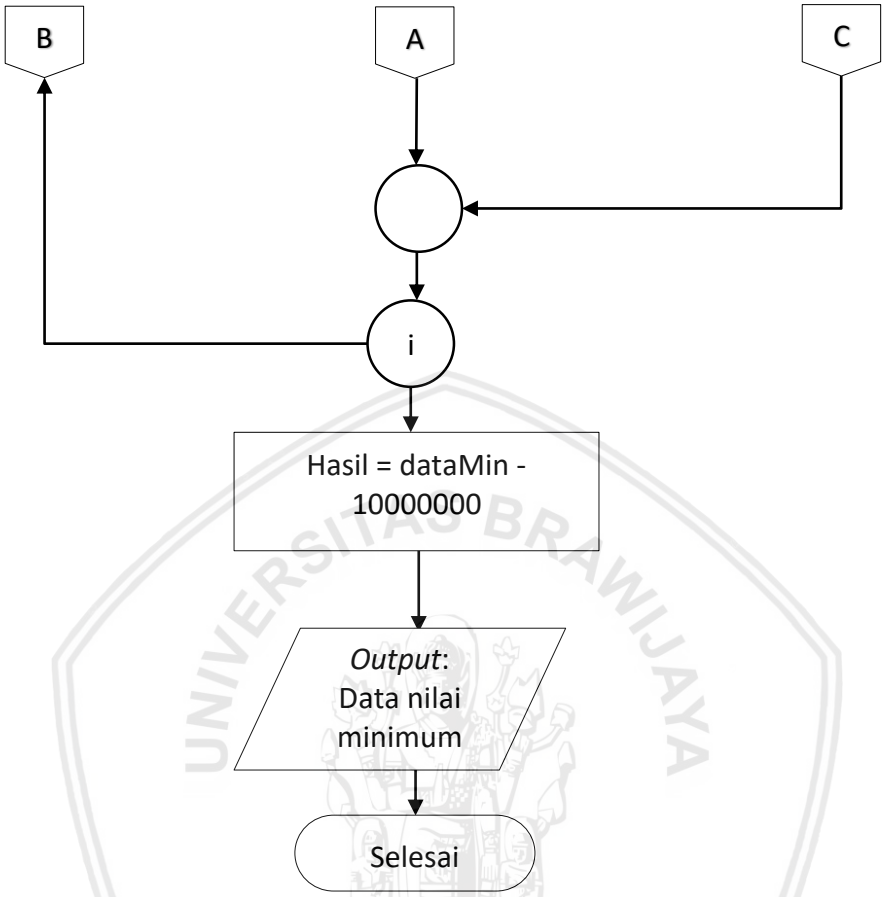
Langkah-langkah proses prediksi nilai *cryptocurrency Bitcoin* menggunakan algoritme ELM berdasarkan Gambar 4.1 adalah sebagai berikut:

1. Sistem menerim masukan berupa nilai cryptocurrency *Bitcoin* sesuai dengan jumlah fitur yang telah ditetapkan sebelumnya.
2. Mencari nilai minimum yang ditunjukkan pada Gambar 4.2.
3. Mencari nilai maksimum yang ditunjukkan pada Gambar 4.3.

4. Menghitung nilai normalisasi untuk setiap data ditunjukkan pada Gambar 4.4.

Pada proses mencari nilai maksimum dan minimum, nilai maksimum akan ditambah dengan 10.000.000 dan nilai minimum akan dikurangi dengan 10.000.000 untuk mencegah adanya *outlier*.

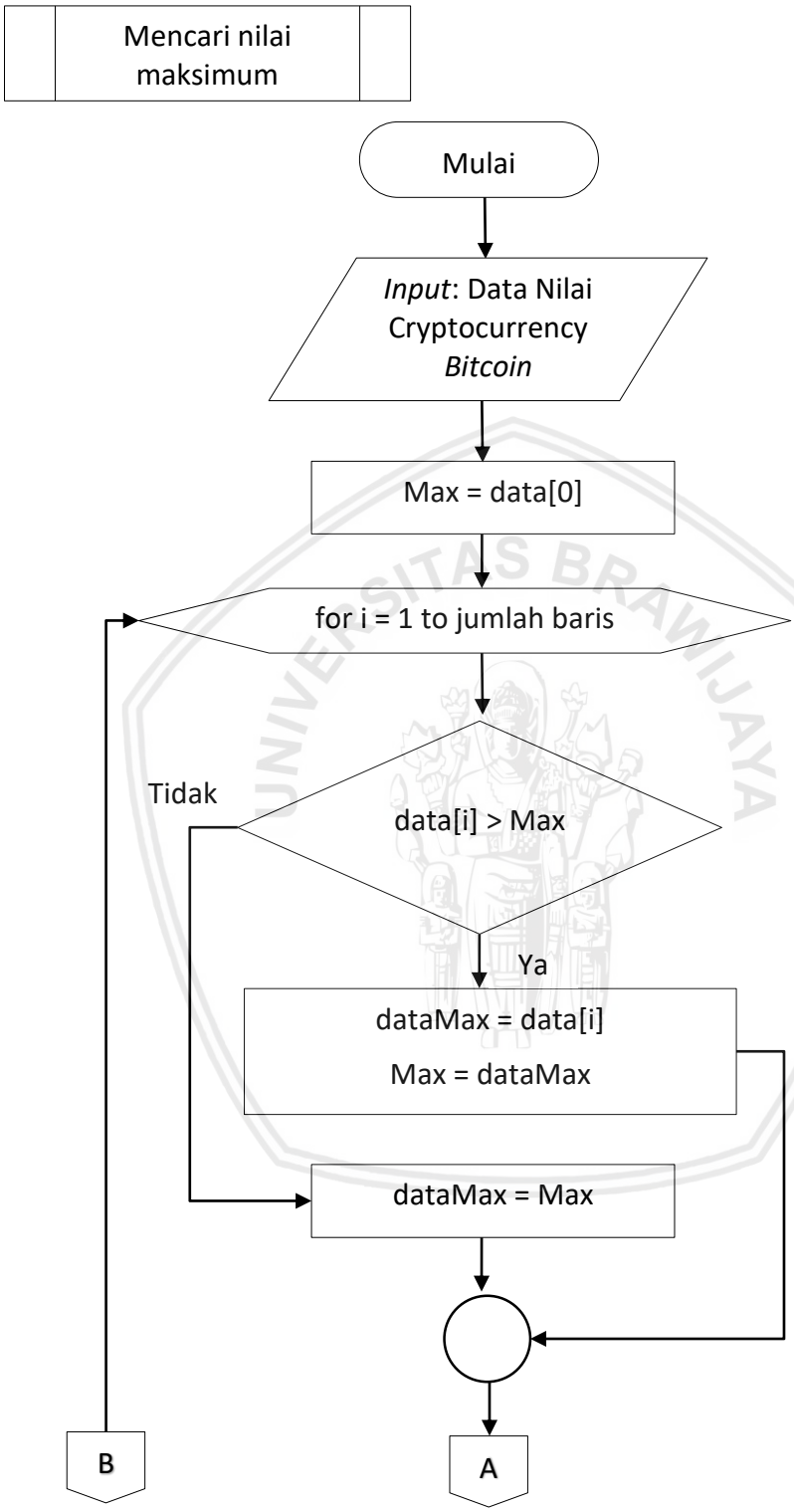


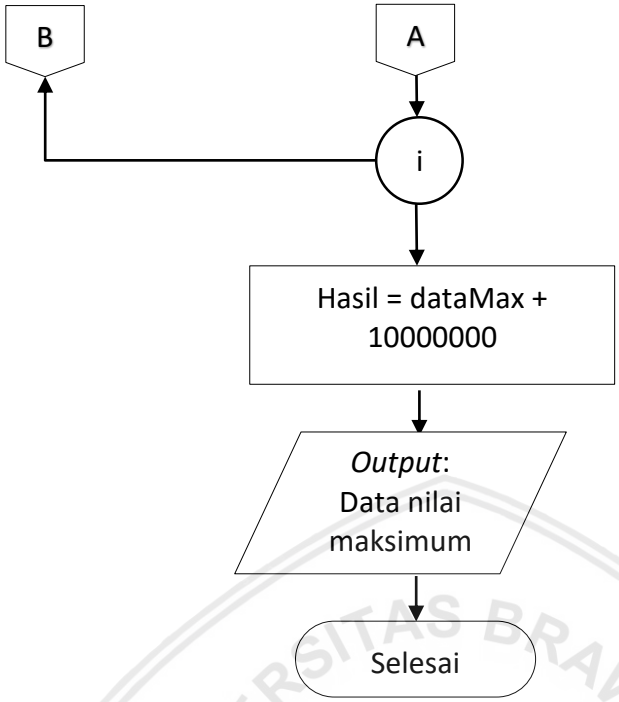


Gambar 4.2 Diagram Alir Proses Mencari Nilai Minimum

Langkah-langkah proses mencari nilai minimum berdasarkan Gambar 4.2 adalah sebagai berikut:

1. Sistem menerima masukan berupa nilai cryptocurrency *Bitcoin*.
2. Sistem mencari nilai terkecil.
3. Keluaran dari sistem adalah nilai terkecil.

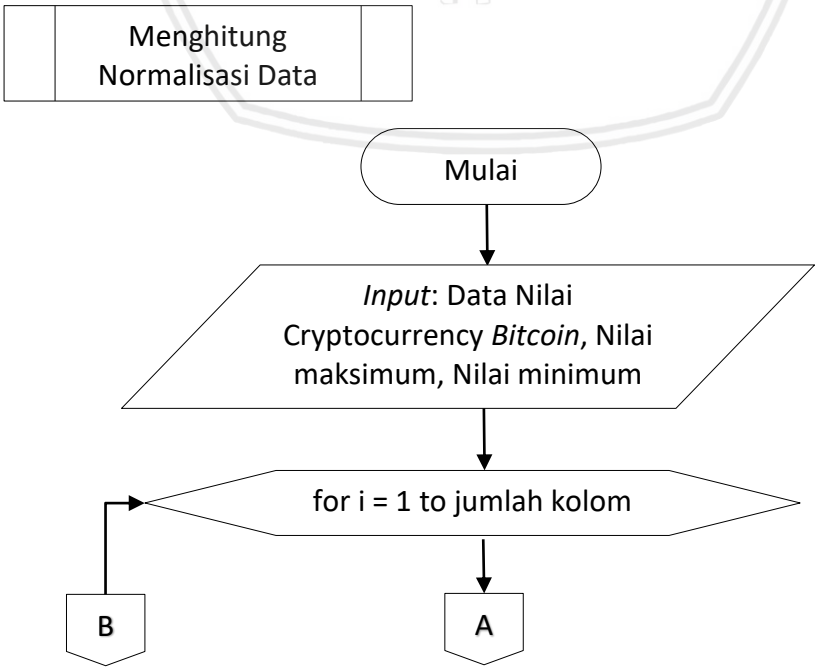

Mencari nilai maksimum
Mulai
Input: Data Nilai Cryptocurrency *Bitcoin*
Max = data[0]
for i = 1 to jumlah baris
data[i] > Max
Tidak
Ya
dataMax = data[i]
Max = dataMax
dataMax = Max
B
A

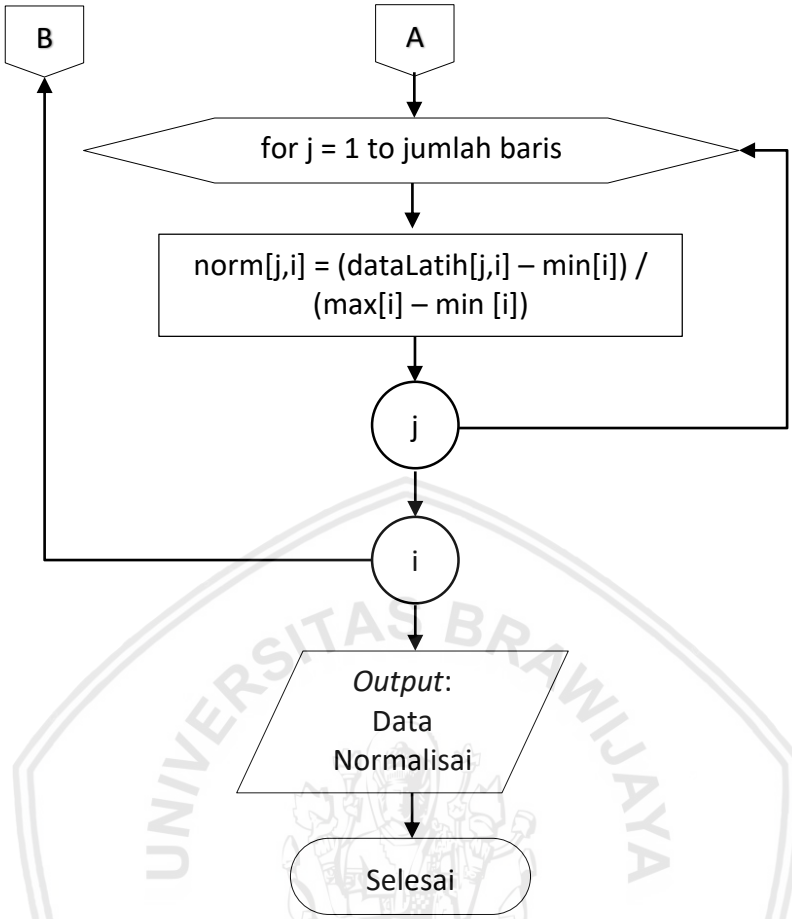


Gambar 4.3 Diagram Alir Proses Mencari Nilai Maksimum

Langkah-langkah proses mencari nilai maksimum berdasarkan Gambar 4.3 adalah sebagai berikut:

1. Sistem menerima masukan berupa nilai cryptocurrency *Bitcoin*.
2. Sistem mencari nilai terbesar.
3. Keluaran dari sistem adalah nilai terbesar.





Gambar 4.4 Diagram Alir Proses Normalisasi

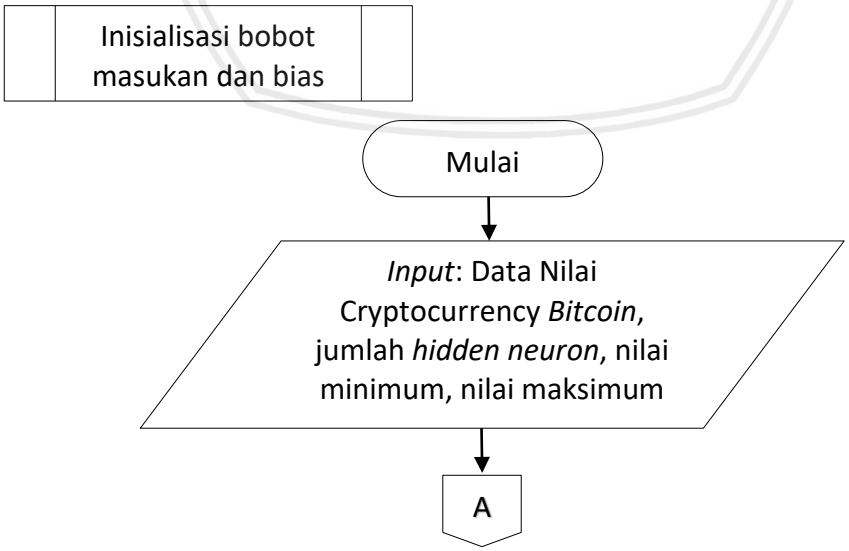
Langkah-langkah proses normalisasi berdasarkan Gambar 4.4 adalah sebagai berikut:

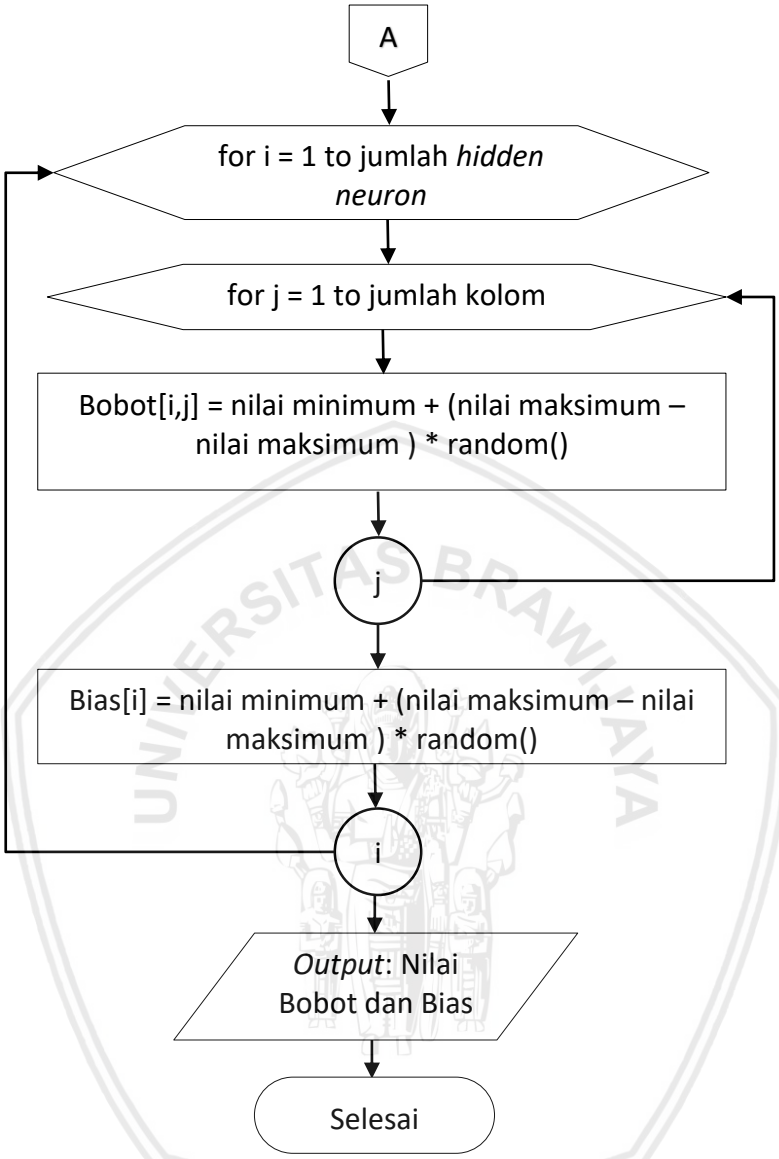
1. Sistem menerima masukan berupa nilai cryptocurrency *Bitcoin* sesuai dengan jumlah fitur yang telah ditetapkan sebelumnya, nilai minimum dan maksimum.
2. Sistem melakukan perulangan sejumlah kolom data.
3. Sistem melakukan perulangan sejumlah baris data.
4. Sistem melakukan proses normalisasi menggunakan Persamaan 2.4 untuk setiap data.
5. Keluaran sistem adalah nilai normalisasi dari setiap data.

4.2.2 Proses Prediksi Nilai Cryptocurrency *Bitcoin* Menggunakan Algoritme *Extreme Learning Machine* (ELM)

Prediksi nilai cryptocurrency *Bitcoin* menggunakan algoritme *Extreme Learning Machine* (ELM) memiliki tujuan untuk mendapatkan nilai prediksi yang mendekati target serta nilai tingkat kesalahan terkecil berdasarkan proses ini. Diagram alir proses prediksi telah ditunjukkan sebelumnya pada Gambar 3.2. Langkah-langkah proses prediksi nilai cryptocurrency *Bitcoin* menggunakan algoritme *Extreme Learning Machine* (ELM) adalah sebagai berikut:

1. Sistem menerima masukan yaitu nilai cryptocurrency *Bitcoin*, jenis fungsi aktivasi yang digunakan, jumlah *hidden neuron* yang digunakan untuk prediksi, serta perbandingan jumlah data latih dan data uji.
2. Inisialisasi bobot masukan dan bias yang ditunjukkan pada Gambar 4.5.
3. Menghitung nilai normalisasi data menggunakan Persamaan 2.4. Untuk alur prosesnya terdapat pada Gambar 4.4.
4. Melakukan proses *training* berdasarkan hasil normalisasi data yang telah dilakukan. Diagram alir ditunjukkan pada Gambar 4.6.
5. Melakukan proses *testing* untuk mendapatkan hasil prediksi. Diagram alir ditunjukkan pada Gambar 4.16.
6. Menghitung nilai denormalisasi untuk mendapatkan hasil prediki dari prses testing menggunakan Persamaan 2.10. Untuk diagram alir proses denormalisasi ditunjukkan pada Gambar 4.17.
7. Menghitung nilai MAPE (*Mean Absolute Percentage Error*) untuk mendapatkan tingkat kesalahan menggunakan Persamaan 2.11. Diagram alir untuk proses perhitungan nilai MAPE ditunjukkan pada Gambar 4.18.
8. Keluaran sistem adalah hasil prediksi setelah didenormalisasi dan nilai MAPE.





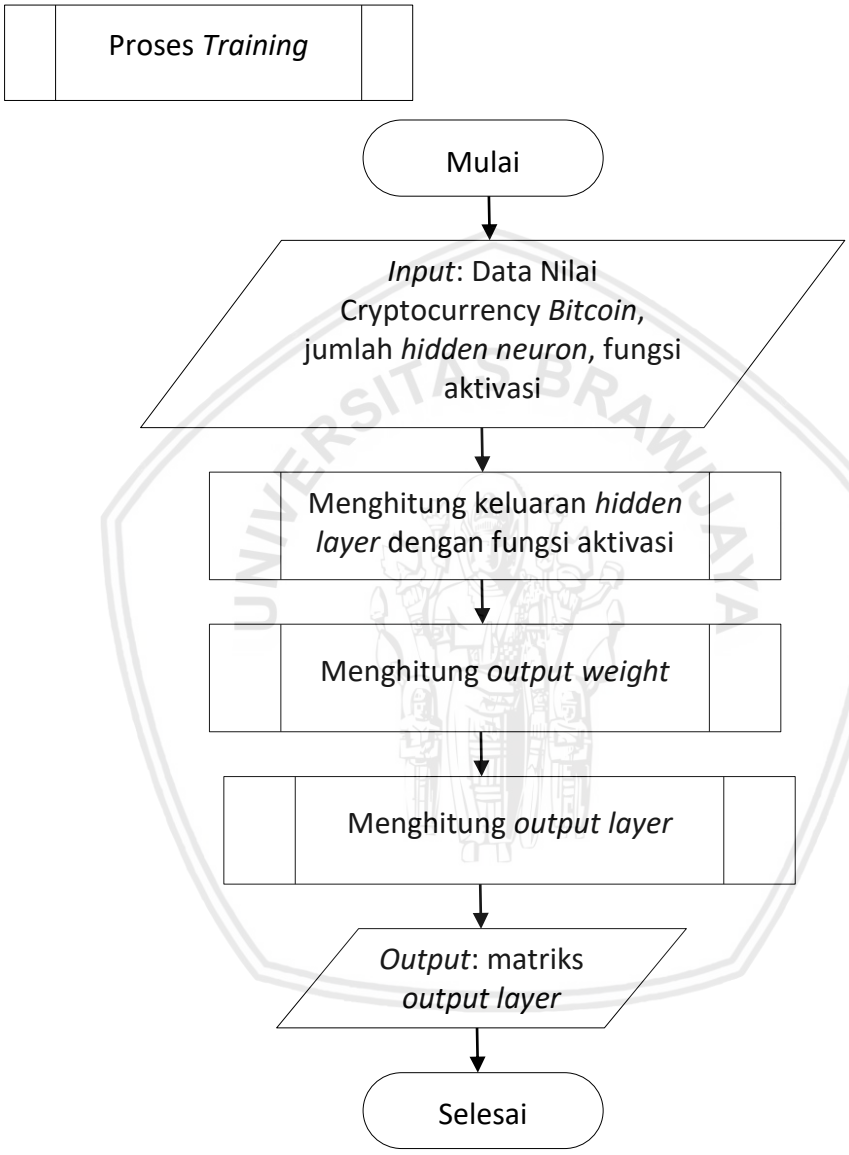
Gambar 4.5 Diagram Alir Proses Inisialisasi Bobot Masukan dan Bias

Langkah-langkah proses inisialisasi bobot masukan dan bias berdasarkan Gambar 4.5 adalah sebagai berikut:

1. Sistem menerima masukan berupa nilai cryptocurrency *Bitcoin* sesuai dengan jumlah fitur yang telah ditetapkan sebelumnya, nilai minimum dan maksimum untuk bobot dan bias, serta jumlah *hidden neuron.*
2. Sistem melakukan perulangan sejumlah hidden neuron.
3. Sistem melakukan perulangan sejumlah kolom data dikurangi 1.
4. Sistem menghitung nilai bobot dan bias secara acak dengan rentang nilai sesuai yang sudah ditetapkan sebelumnya.
5. Sistem mengeluarkan nilai bobot masukan dan bias untuk digunakan pada proses selanjutnya.

4.2.3 Proses *Training* Algoritme *Extreme Learning Machine* (ELM)

Tujuan dilakukannya proses pelatihan (*training*) adalah untuk mendapatkan bobot masukan, bias, dan bobot keluaran yang digunakan pada proses pengujian (*testing*). Diagram alir proses *training* algoritme ELM ditunjukkan pada Gambar 4.6.



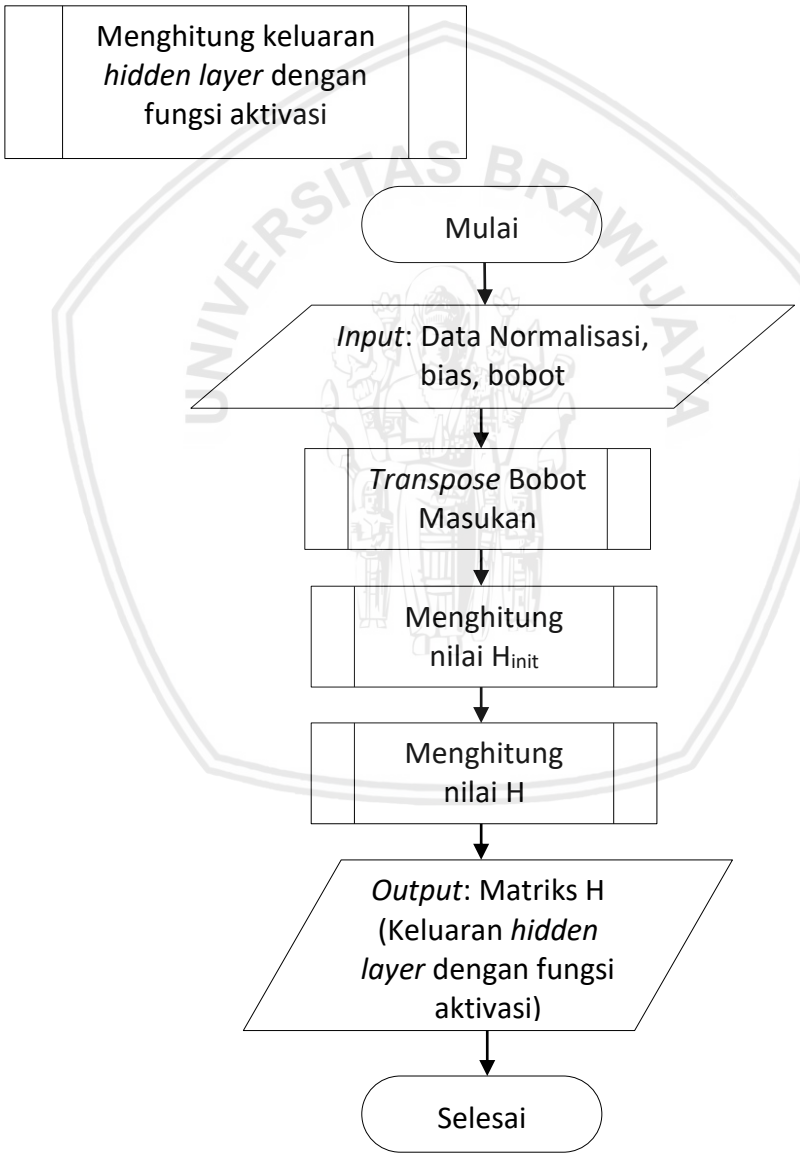
Gambar 4.6 Diagram Alir Proses *Training*

Langkah-langkah diagram alir proses *training* algoritme ELM berdasarkan Gambar 4.6 adalah sebagai berikut:

1. Sistem menerima masukkan (*input*) berupa data nilai cryptocurrency *Bitcoin* sebagai data latih, lalu jumlah *hidden neuron* serta jenis fungsi aktivasi.
2. Menghitung keluaran *hidden layer* dengan fungsi aktivasi yang didalamnya harus menghitung nilai H_{init} menggunakan Persamaan 2.5. Setelah mendapatkan nilai H_{init}, kita dapat menghitung keluaran hidden layer

menggunakan Persamaan 2.6. Diagram alir proses menghitung keluaran *hidden layer* dengan fungsi aktivasi ditunjukkan pada Gambar 4.7.

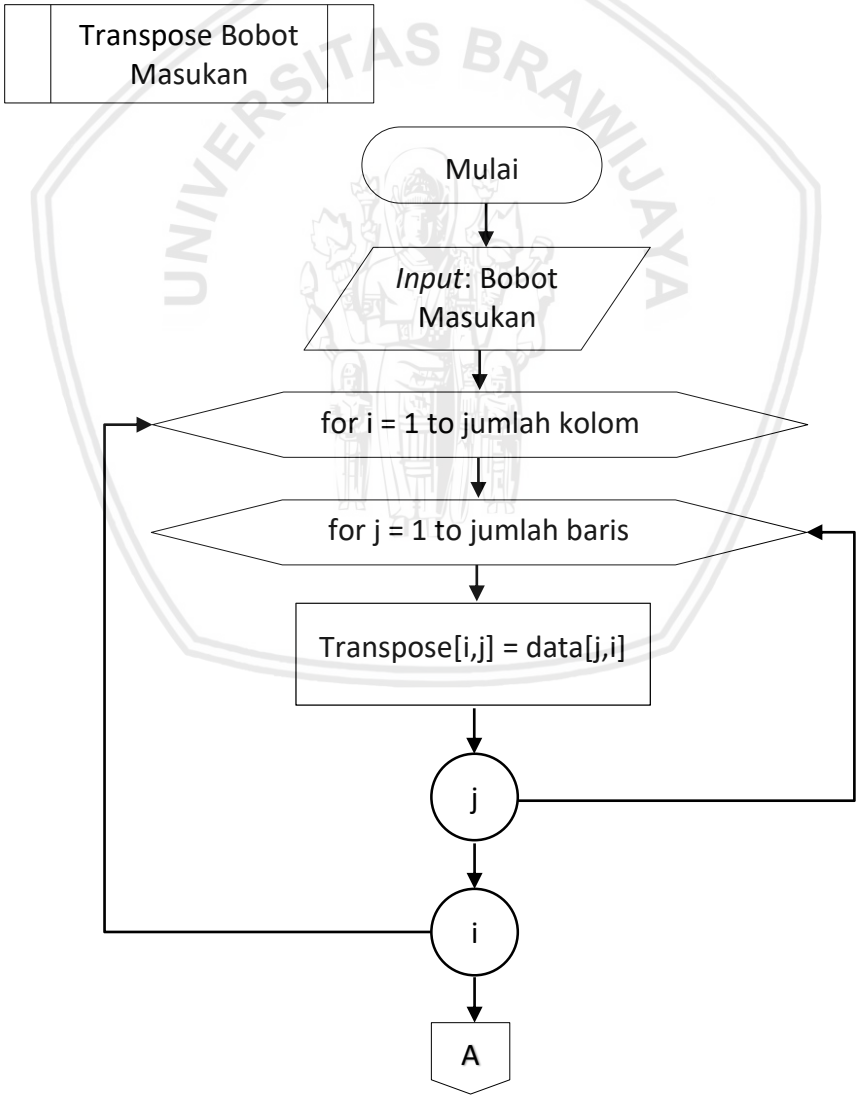
3. Menghitung *output weight* yang didalamnya kita harus menghitung matriks *moore-penrose pseudo inverse* (H^+) menggunakan Persamaan 2.7. Setelah mendapatkannya, kita dapat menghitung *output weight* menggunakan Persamaan 2.8. Diagram alir proses menghitung *output weight* ditunjukkan pada Gambar 4.11.
4. Menghitung *output layer* menggunakan Persamaan 2.9. Diagram alir menghitung *output layer* ditunjukkan pada Gambar 4.13.
5. Keluaran sistem yaitu matriks *output layer*.

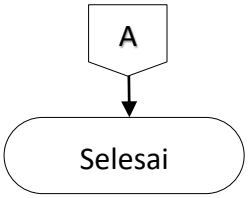


Gambar 4.7 Diagram Alir Proses Menghitung Keluaran Hidden Layer Dengan Fungsi Aktivasi

Langkah-langkah diagram alir proses menghitung keluaran *hidden layer* dengan fungsi aktivasi berdasarkan Gambar 4.7 adalah sebagai berikut:

1. Sistem menerima masukan berupa data yang sudah dinormalisasi beserta bias.
2. Sistem melakukan proses *transpose* bobot masukan. Diagram alir *transpose* bobot masukan ditunjukkan pada Gambar 4.8.
3. Sistem melakukan proses perhitungan nilai H_{init} menggunakan Persamaan 2.5. Diagram alir perhitungan nilai H_{init} ditunjukkan pada Gambar 4.9.
4. Sistem melakukan proses perhitungan nilai H menggunakan Persamaan 2.6. Diagram alir perhitungan nilai H ditunjukkan pada Gambar 4.10.
5. Keluaran sistem berupa matriks H (keluaran *hidden layer* dengan fungsi aktivasi).

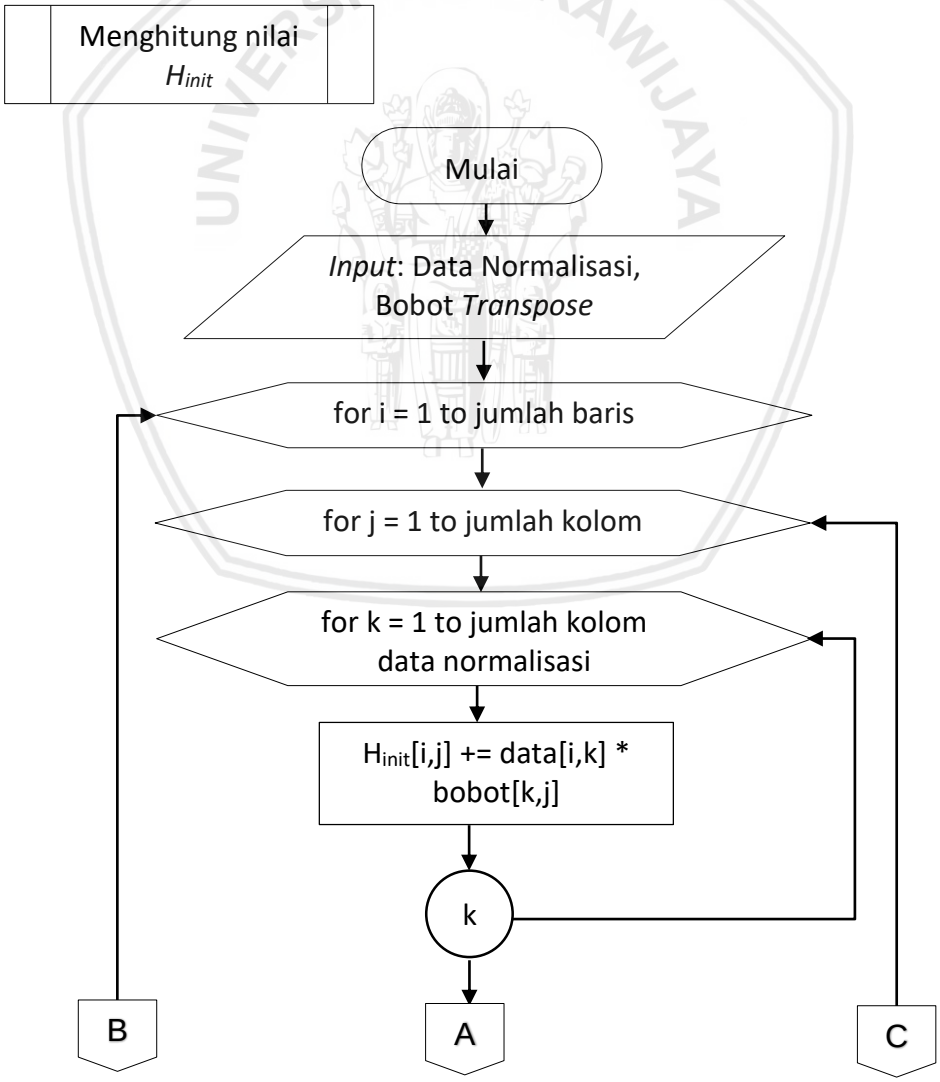


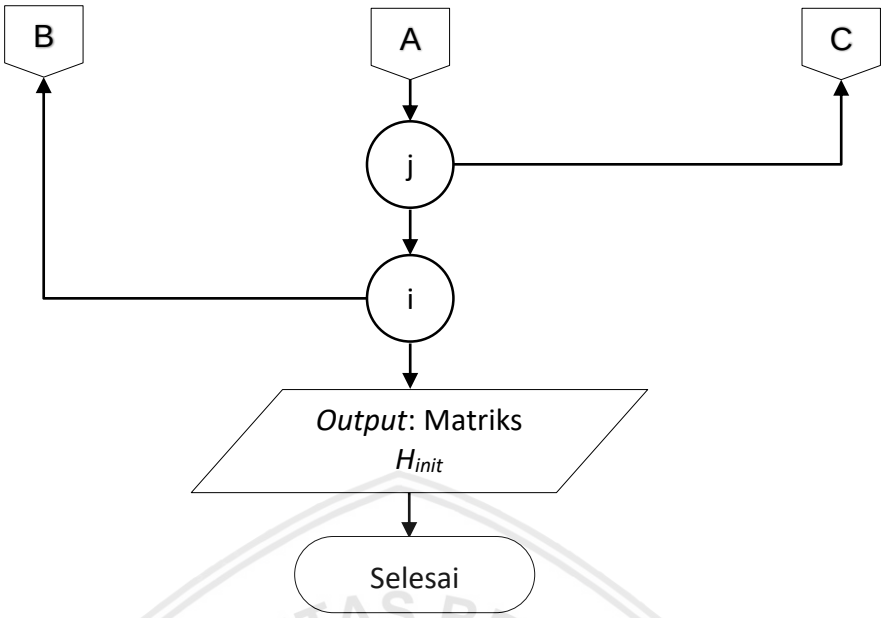


Gambar 4.8 Diagram Alir Proses *Transpose* Bobot Masukan

Langkah-langkah diagram alir *transpose* bobot masukan berdasarkan Gambar 4.8 adalah sebagai berikut:

1. Sistem menerima masukan berupa bobot masukan yang akan dilakukan proses transpose.
2. Sistem melakukan perulangan sejumlah baris dan kolom bobot.
3. Sistem melakukan proses transpose bobot masukan.
4. Keluaran sistem adalah Matriks transpose bobot masukan

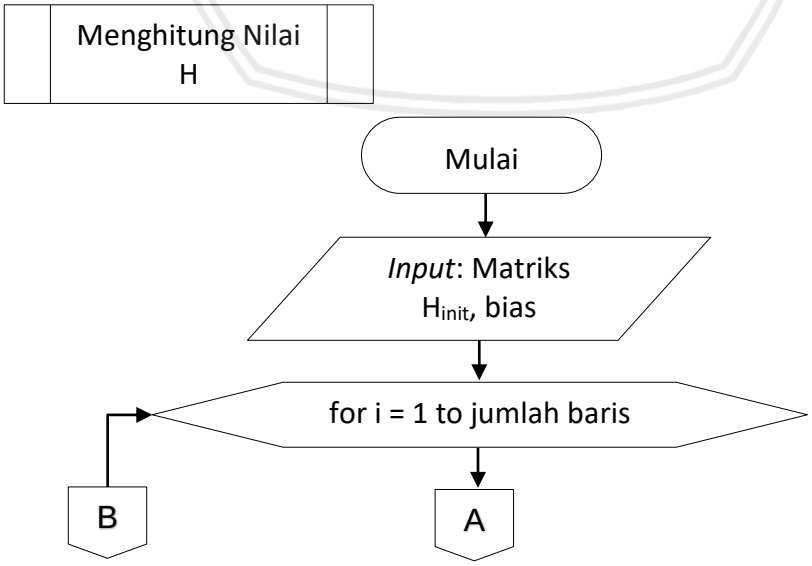


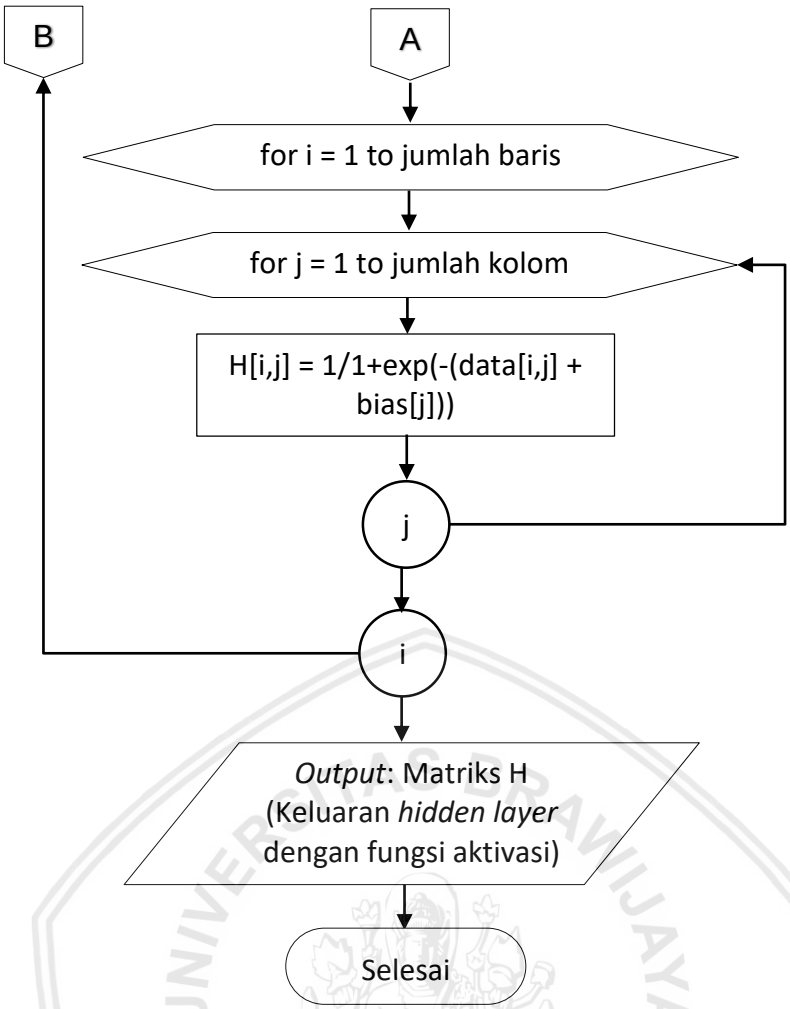


Gambar 4.9 Diagram Alir Menghitung Nilai Hinit

Langkah-langkah diagram alir *transpose* bobot masukan berdasarkan Gambar 4.9 adalah sebagai berikut:

1. Sistem menerima masukan berupa data normalisasi serta bobot yang sudah di*transpose*.
2. Sistem melakukan perulangan.
3. Sistem melakukan perkalian matriks antara data normalisasi dengan bobot *transpose*.
4. Keluaran sistem adalah matriks H_{init}.

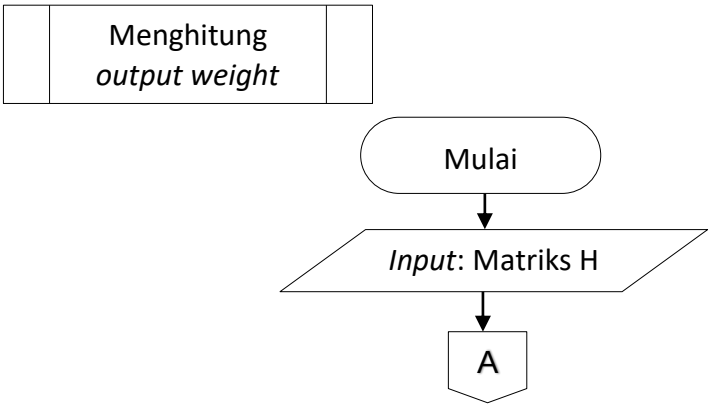


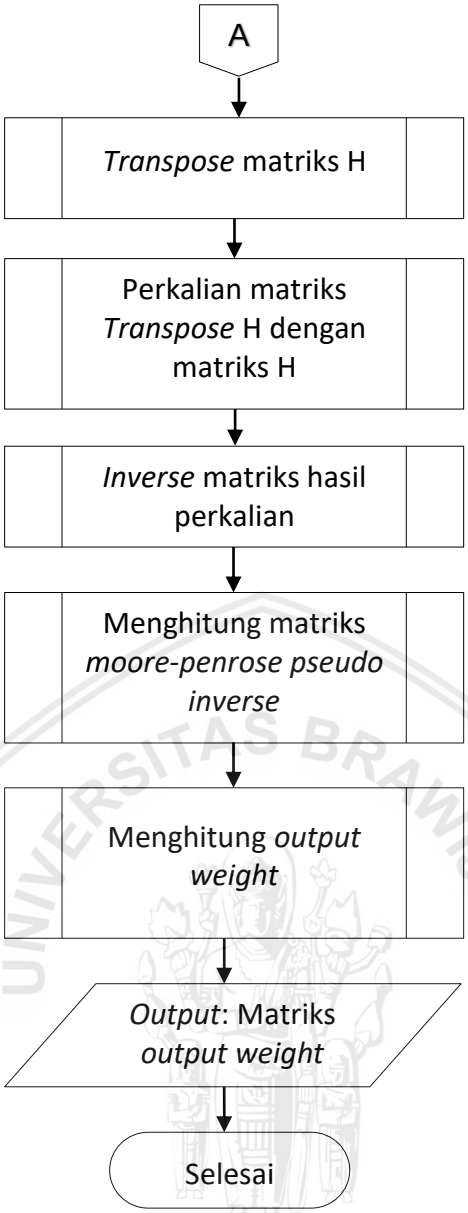


Gambar 4.10 Diagram Alir Menghitung Nilai H

Langkah-langkah diagram alir menghitung nilai H berdasarkan Gambar 4.10 adalah sebagai berikut:

1. Sistem menerima masukan berupa matriks H_{init} beserta nilai bias.
2. Sistem melakukan perulangan.
3. Sistem melakukan seleksi kondisi untuk menentukan fungsi aktivasi. Setelah fungsi aktivasi ditentukan, maka dilakukan perhitungan nilai H menggunakan fungsi aktivasi tersebut.
4. Keluaran sistem adalah matriks H (keluaran *hidden layer* dengan fungsi aktivasi).



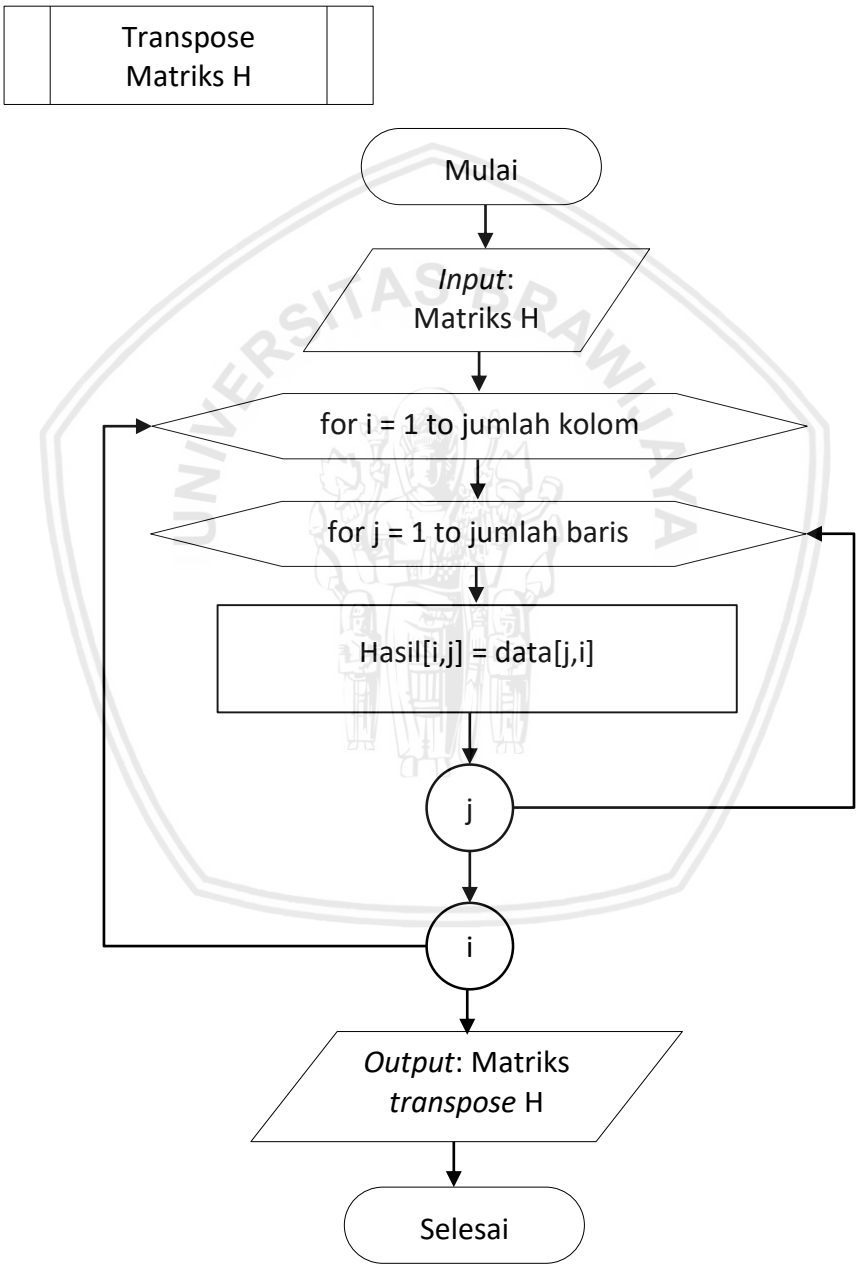


Gambar 4.11 Diagram Alir Menghitung *Output Weight*

Langkah-langkah diagram alir menghitung nilai *output weight* berdasarkan Gambar 4.11 adalah sebagai berikut:

1. Sistem menerima masukan berupa matriks H.
2. Sistem melakukan proses *transpose* terhadap matriks H. Diagram alir *transpose* matriks H ditunjukkan pada Gambar 4.12.
3. Sistem melakukan perkalian antara matriks *transpose* H dengan matriks H. Diagram alir perkalian matriks ditunjukkan pada Gambar 4.13.
4. Sistem melakukan proses inverse matriks hasil perkalian antara matriks *transpose* H dengan matriks H. Diagram alir inverse matriks hasil perkalian antara matriks *transpose* H dengan matriks H ditunjukkan pada Gambar 4.14.

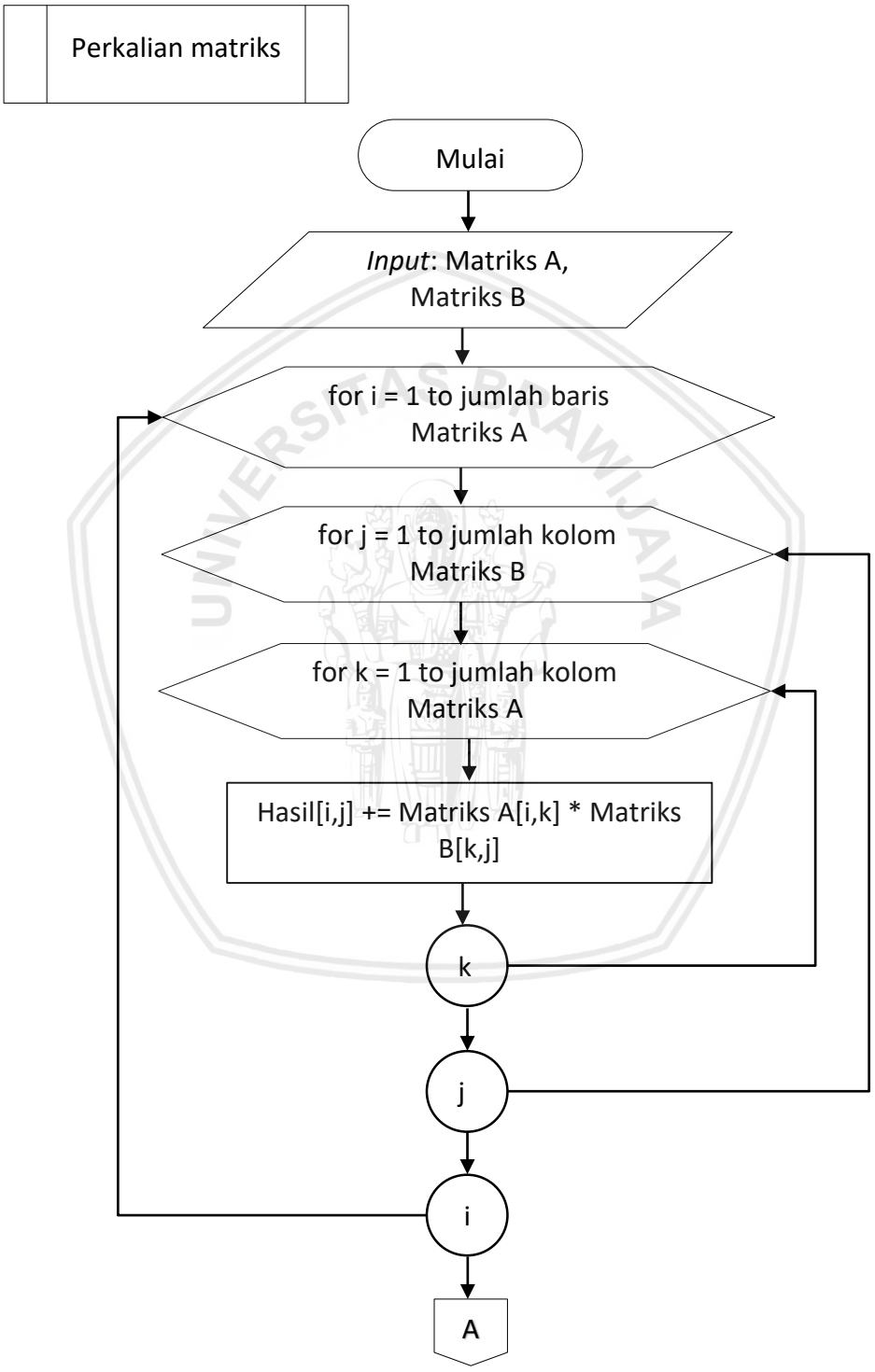
5. Sistem melakukan perhitungan untuk mendapatkan matriks *moore-penrose pseudo inverse* menggunakan Persamaan 2.7. Diagram alir perkalian matriks ditunjukkan pada Gambar 4.13.
6. Sistem melakukan perhitungan untuk menghitung nilai *output weight* menggunakan Persamaan 2.8. Diagram alir menghitung *output weight* ditunjukkan pada Gambar 4.13.
7. Keluaran sistem adalah matriks *output weight*.

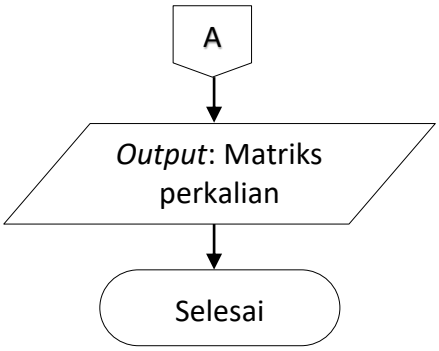


Gambar 4.12 Diagram Alir *Transpose* Bobot Masukan

Langkah-langkah diagram alir menghitung nilai *output weight* berdasarkan Gambar 4.12 adalah sebagai berikut:

1. Sistem menerima masukan berupa matriks H.
2. Sistem melakukan perulangan sejumlah kolom dan baris pada matriks H.
3. Keluaran sistem adalah matriks *transpose* H.

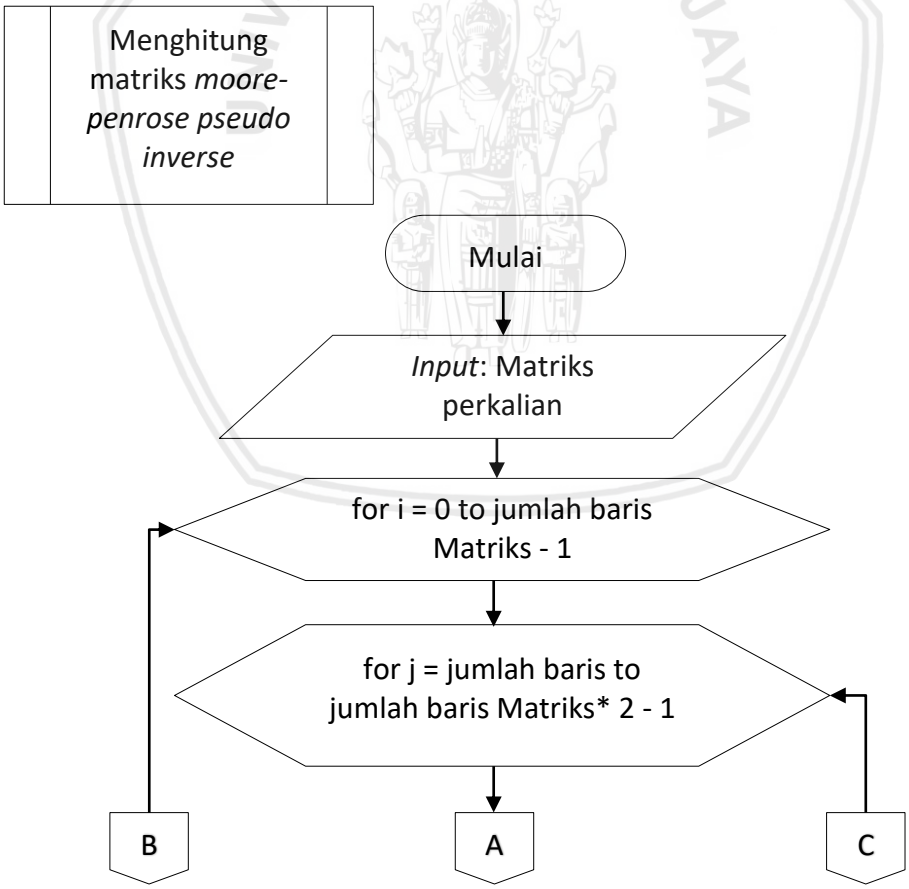


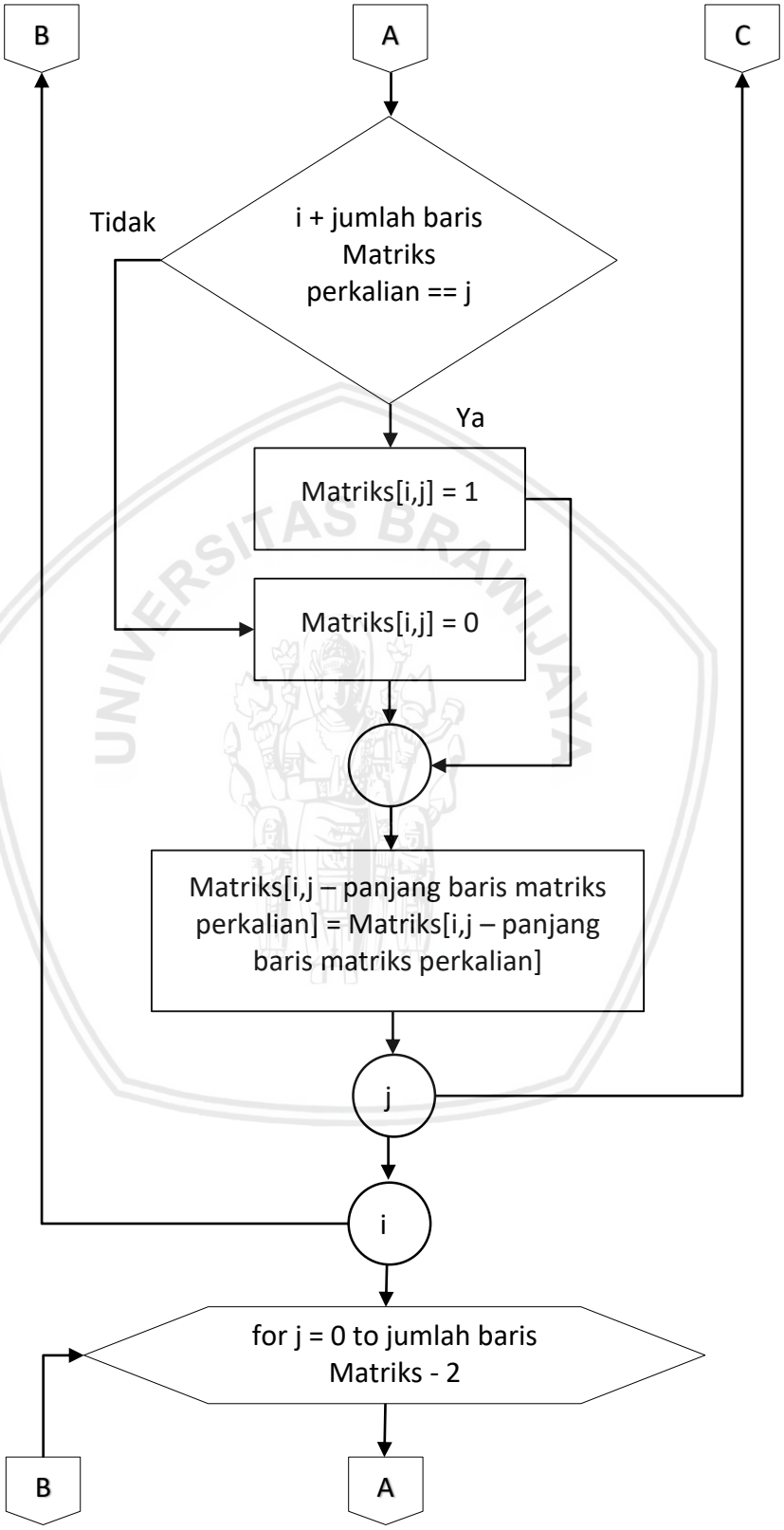


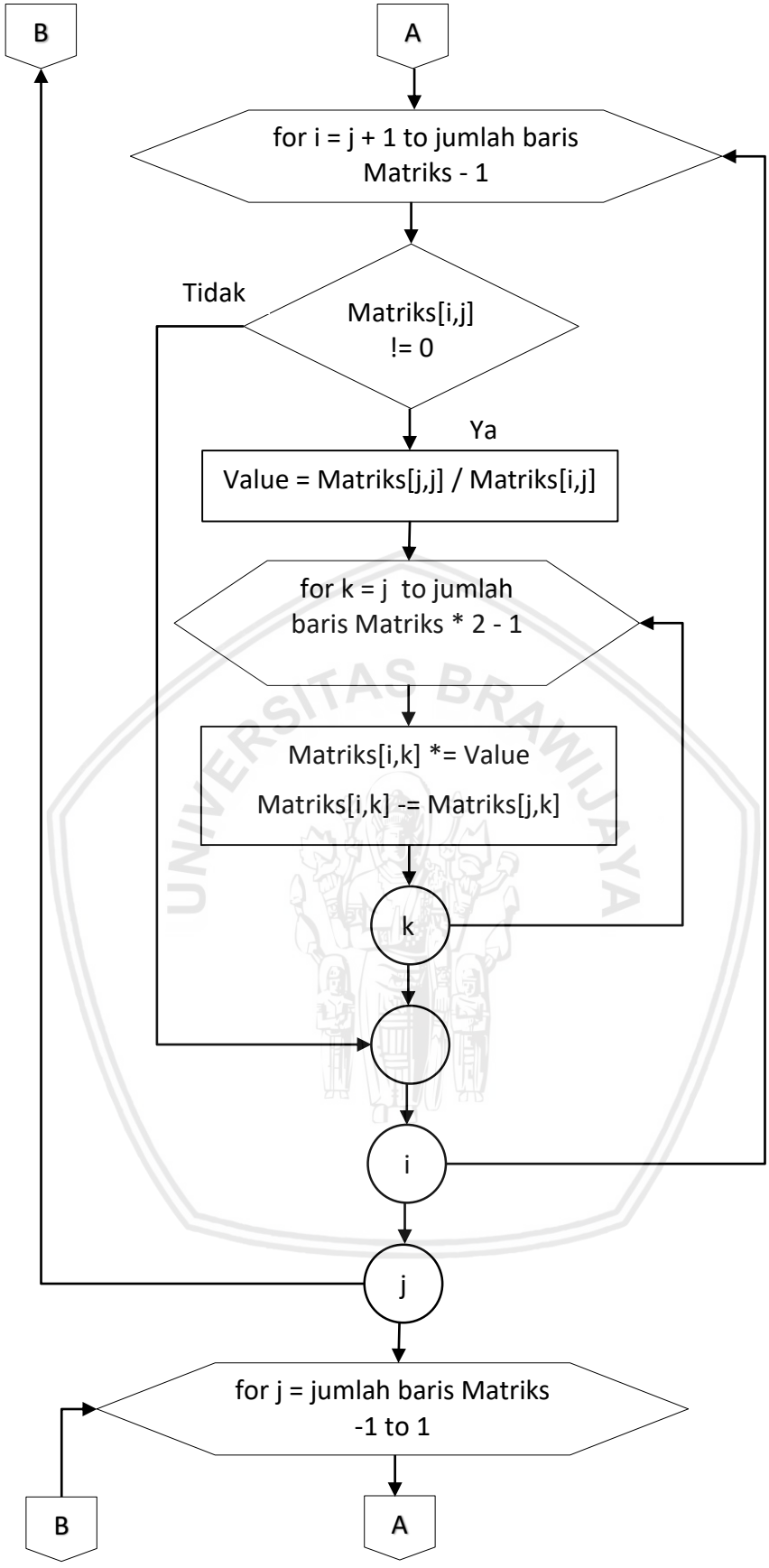
Gambar 4.13 Diagram Alir Matriks Perkalian

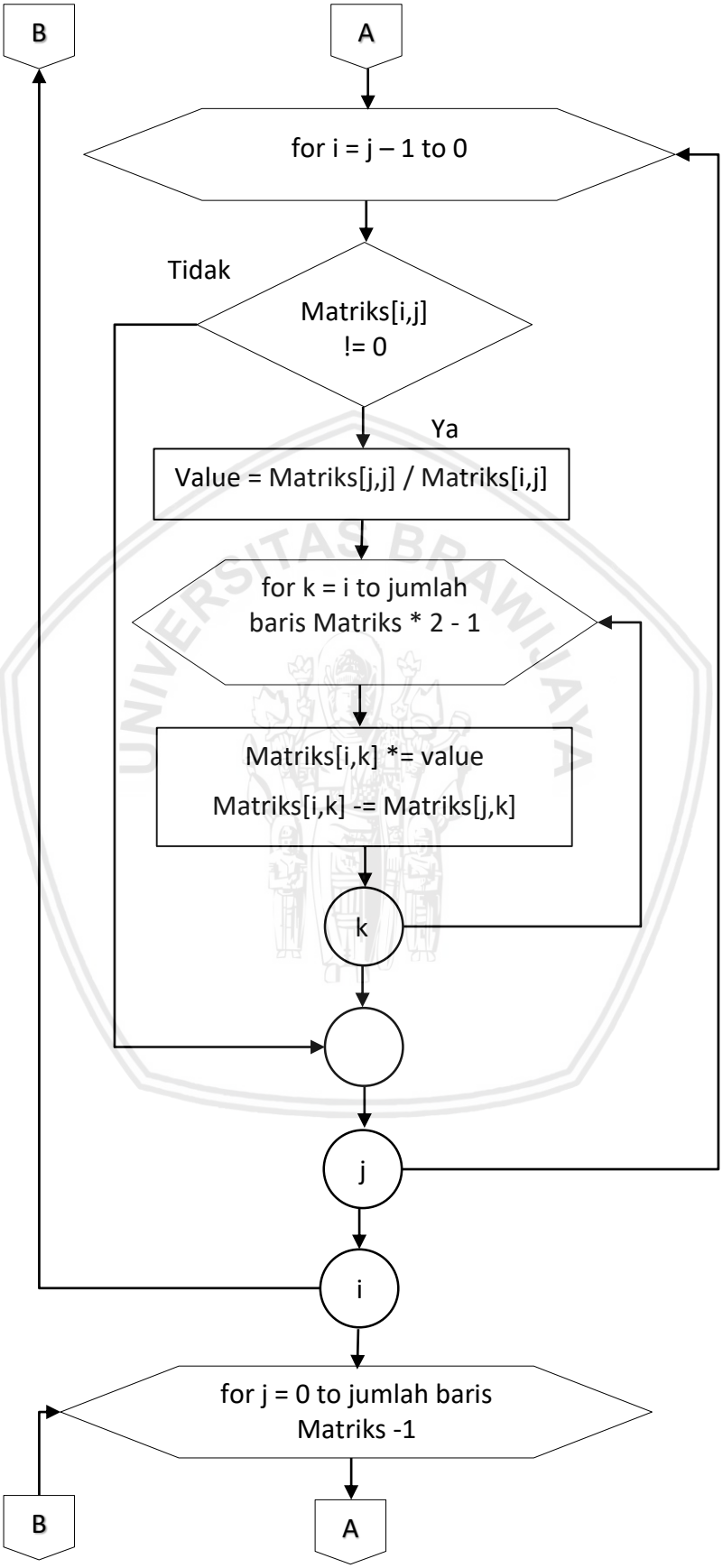
Langkah-langkah diagram alir menghitung nilai *output weight* berdasarkan Gambar 4.13 adalah sebagai berikut:

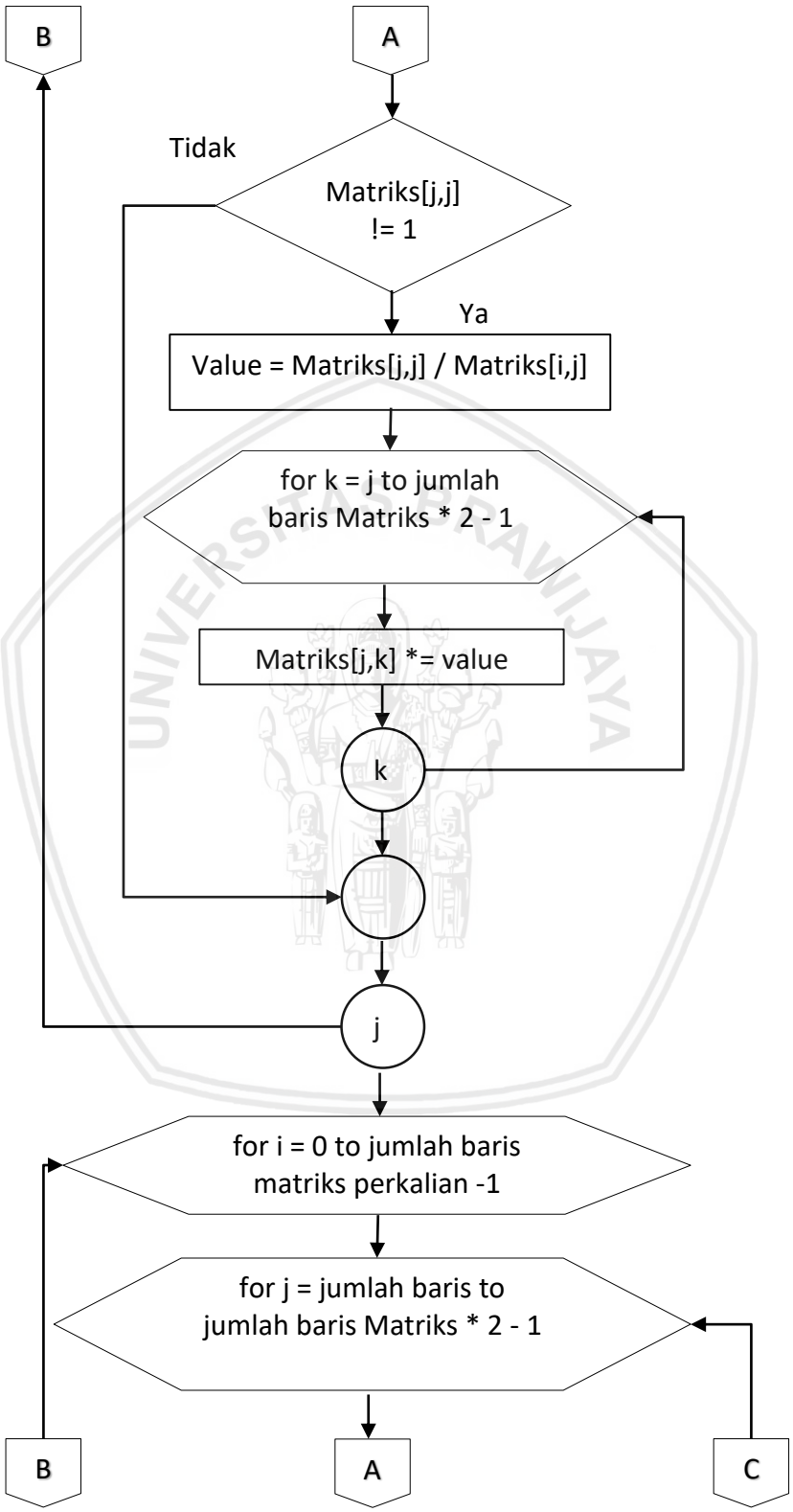
1. Sistem menerima masukan berupa matriks A dan matriks B yang akan dilakukan proses perkalian
2. Sistem melakukan proses perkalian dengan melakukan perkalian elemen pada kolom matriks A dengan elemen pada baris matriks B.
3. Keluaran sistem adalah matriks hasil perkalian.

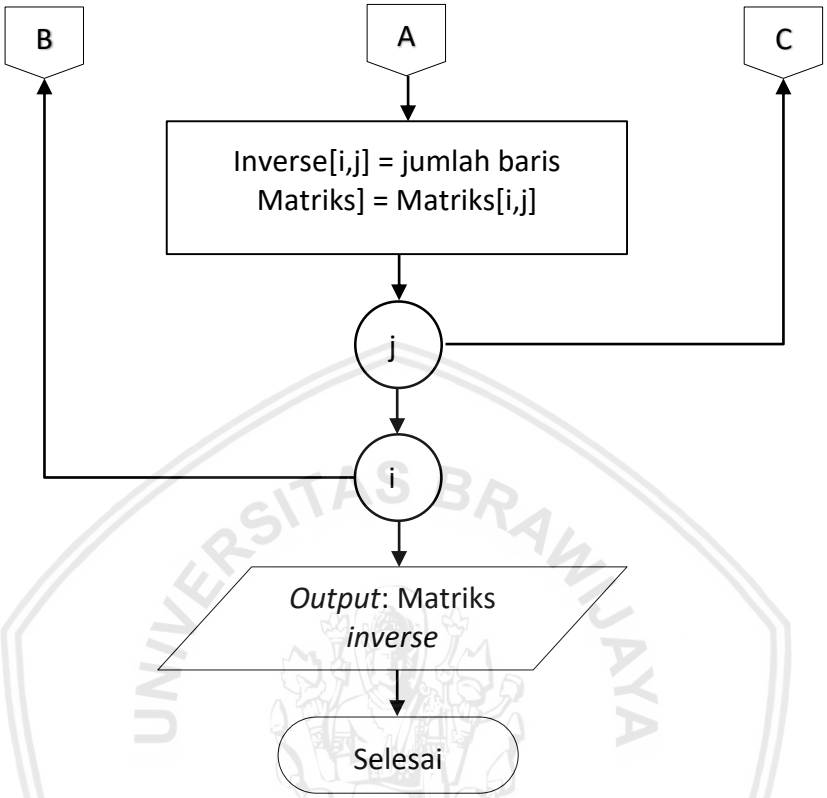



B
A
C
Tidak
i + jumlah baris Matriks perkalian == j
Ya
Matriks[i,j] = 1
Matriks[i,j] = 0
Matriks[i,j – panjang baris matriks perkalian] = Matriks[i,j – panjang baris matriks perkalian]
j
i
for j = 0 to jumlah baris Matriks - 2
B
A


B
A
for i = j + 1 to jumlah baris Matriks - 1
Tidak
Matriks[i,j] != 0
Ya
Value = Matriks[j,j] / Matriks[i,j]
for k = j to jumlah baris Matriks * 2 - 1
Matriks[i,k] *= Value
Matriks[i,k] -= Matriks[j,k]
k
i
j
for j = jumlah baris Matriks -1 to 1
B
A


B
A
for i = j – 1 to 0
Tidak
Matriks[i,j]
!= 0
Ya
Value = Matriks[j,j] / Matriks[i,j]
for k = i to jumlah
baris Matriks * 2 - 1
Matriks[i,k] *= value
Matriks[i,k] -= Matriks[j,k]
k
j
i
for j = 0 to jumlah baris
Matriks -1
B
A

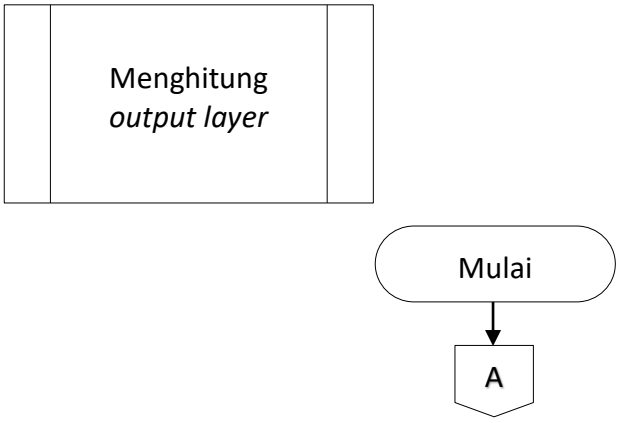

B
A
Tidak
Matriks[j,j]
!= 1
Ya
Value = Matriks[j,j] / Matriks[i,j]
for k = j to jumlah
baris Matriks * 2 - 1
Matriks[j,k] *= value
k
j
for i = 0 to jumlah baris
matriks perkalian -1
for j = jumlah baris to
jumlah baris Matriks * 2 - 1
B
A
C

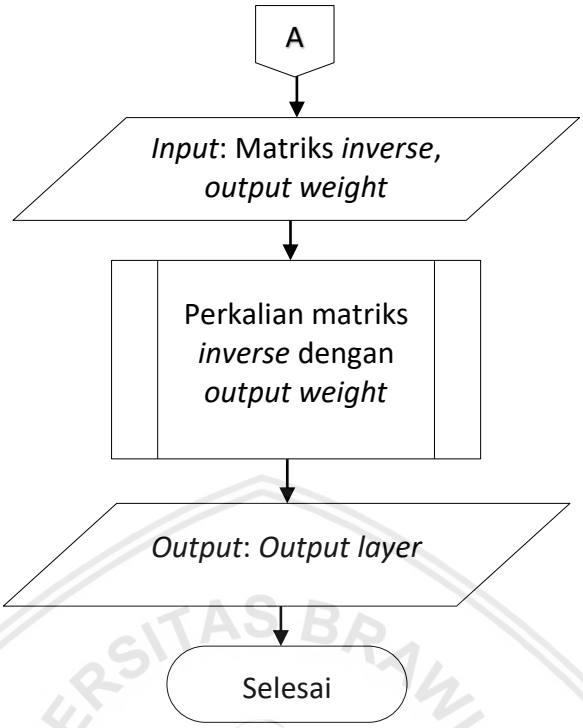


Gambar 4.14 Diagram Alir Menghitung Matriks *Moore-Penrose Pseudo Inverse*

Langkah-langkah diagram alir menghitung nilai *output weight* berdasarkan Gambar 4.14 adalah sebagai berikut:

1. Sistem menerima masukan berupa matriks hasil perkalian yang akan dilakukan proses *inverse*.
2. Sistem melakukan proses *inverse* terhadap setiap data yang terdapat dalam matriks.
3. Keluaran sistem adalah matriks hasil *inverse* yang telah dilakukan sistem.





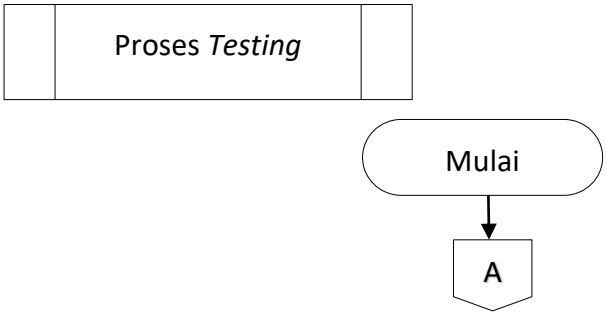
Gambar 4.15 Diagram Alir Menghitung Output Layer

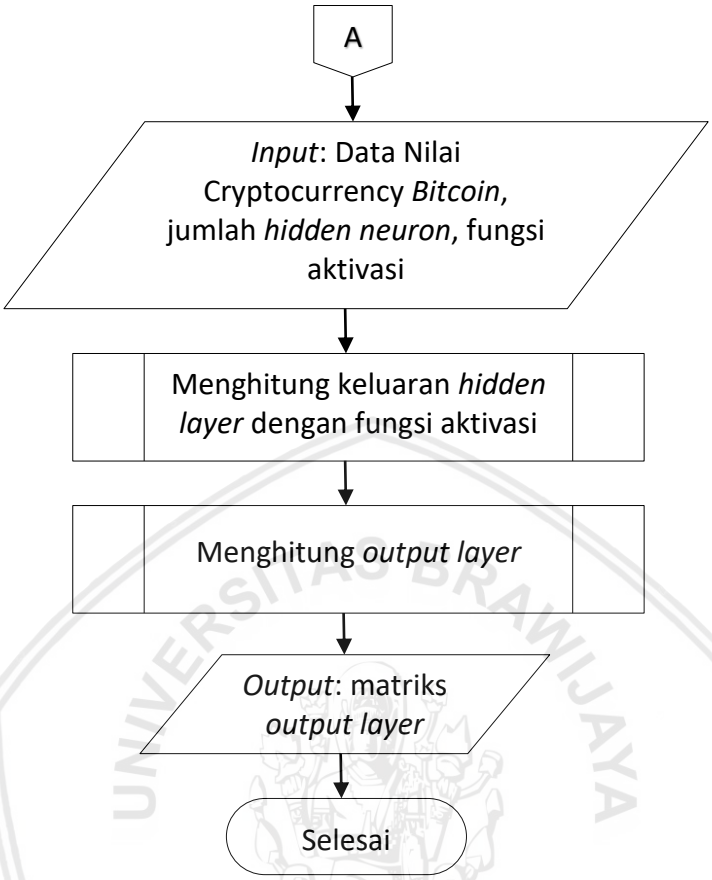
Langkah-langkah diagram alir menghitung nilai *output weight* berdasarkan Gambar 4.15 adalah sebagai berikut:

1. Sistem menerima masukan berupa matriks *inverse* dan *output weight*.
2. Sistem melakukan perkalian yang ditunjukkan pada Gambar 4.13.
3. Keluaran sistem adalah nilai *output layer*.

4.2.4 Proses *Testing* Algoritme *Extreme Learning Machine* (ELM)

Tujuan dari proses ini *testing* adalah untuk mendapatkan nilai bobot masukan, bias, serta nilai output layer. Untuk *output weight* sendiri didapat dari hasil *training*. Diagram alir proses testing algoritme ELM ditunjukkan pada Gambar 4.16.





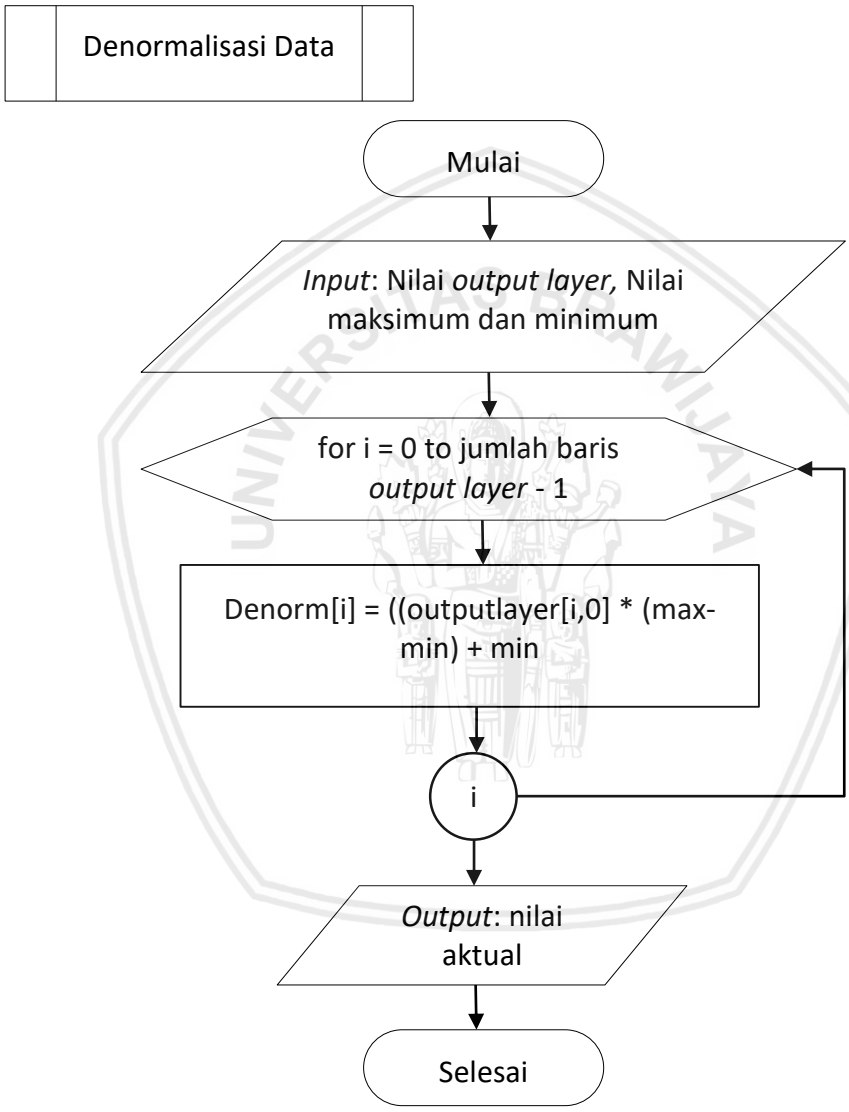
Gambar 4.16 Diagram Alir Proses *Testing* Algoritme ELM

Langkah-langkah diagram alir proses *testing* algoritme ELM berdasarkan Gambar 4.16 adalah sebagai berikut:

1. Sistem menerima masukkan (*input*) berupa data nilai cryptocurrency *Bitcoin* sebagai data latih, lalu jumlah *hidden neuron* serta jenis fungsi aktivasi.
2. Menghitung keluaran *hidden layer* dengan fungsi aktivasi yang didalamnya harus menghitung nilai H_{init} menggunakan Persamaan 2.5. Setelah mendapatkan nilai H_{init}, kita dapat menghitung keluaran hidden layer menggunakan Persamaan 2.6. Diagram alir proses menghitung keluaran *hidden layer* dengan fungsi aktivasi ditunjukkan pada Gambar 4.7.
3. Menghitung *output layer* menggunakan Persamaan 2.9. Diagram alir menghitung *output layer* ditunjukkan pada Gambar 4.13.
4. Keluaran sistem adalah matriks *output layer* dari hasil proses *testing*.

4.2.5 Denormalisasi Data

Proses denormalisasi adalah proses dalam algoritme ELM untuk mendapatkan nilai aktual hasil prediksi. Nilai aktual tersebut nantinya akan digunakan untuk menguji tingkat kesalan menggunakan metode MAPE (Mean Absolute Pencertage Error). Denormalisasi data terdapat pada tahap akhir proses *training* dan *testing*. Dibawah ini akan ditunjukkan diagram alir denormalisasi data pada Gambar 4.17.



Gambar 4.17 Diagram Alir Denormalisasi Data

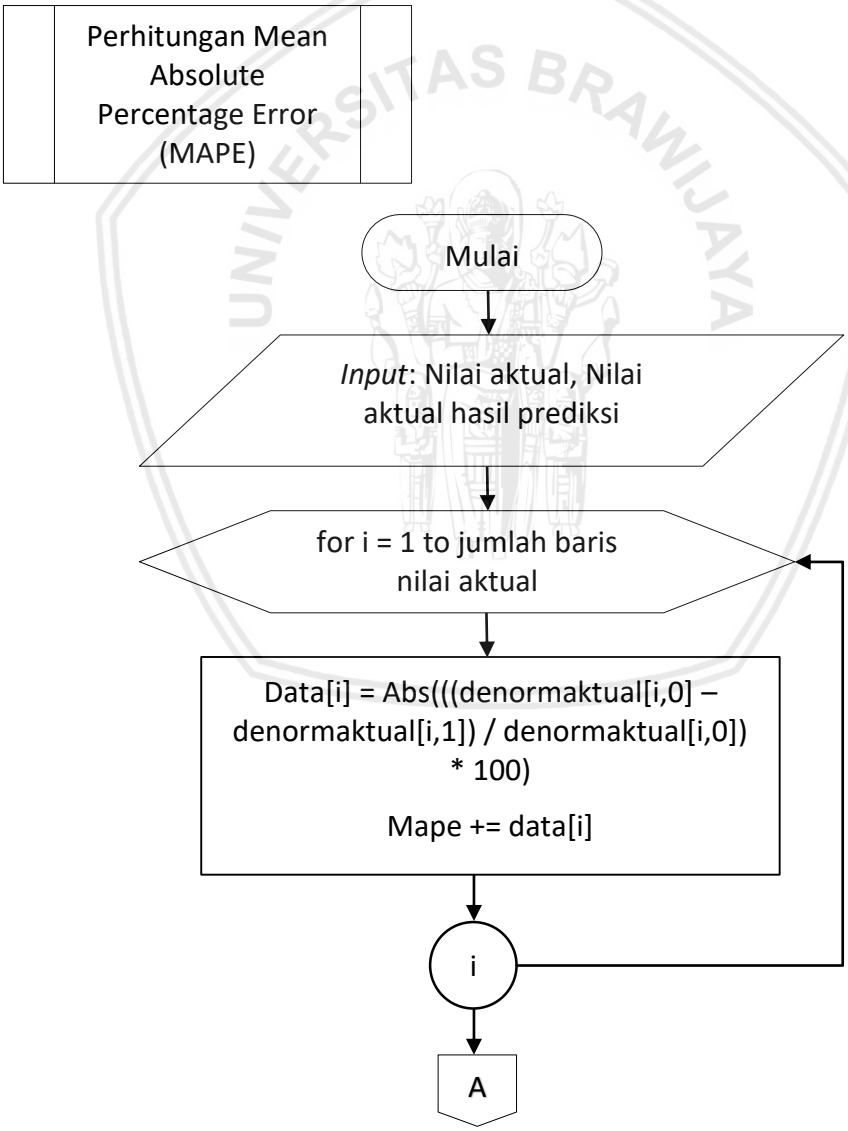
Langkah-langkah diagram alir denormalisasi data berdasarkan Gambar 4.17 adalah sebagai berikut:

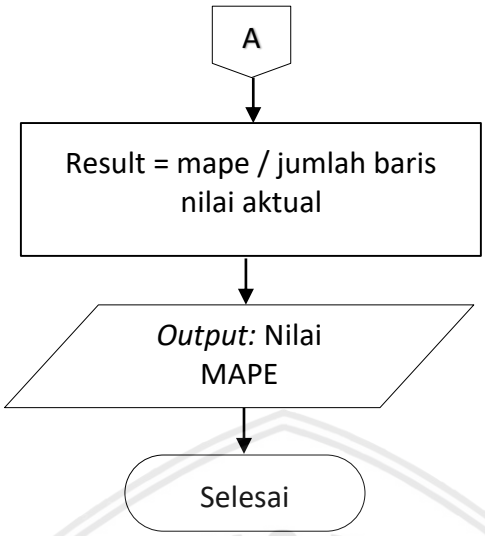
1. Sistem menerima masukkan nilai *output layer*, nilai maksimum dan minimum per fitur.
2. Sistem melakukan perulangan sebanyak jumlah baris pada nilai *output layer*.

3. Sistem melakukan perhitungan untuk mendapatkan hasil denormalisasi data.
4. Keluaran sistem adalah nilai aktual hasil prediksi.

4.2.6 Perhitungan *Mean Absolute Percentage Error* (MAPE)

Perhitungan *Mean Absolute Percentage Error* (MAPE) dilakukan untuk mendapatkan tingkat kesalahan dari prediksi yang telah dilakukan sebelumnya. Dengan menggunakan MAPE, kita dapat mendapatkan nilai selisih antara nilai aktual dengan nilai prediksi. Diagram alir perhitungan MAPE ditunjukkan pada Gambar 4.18.





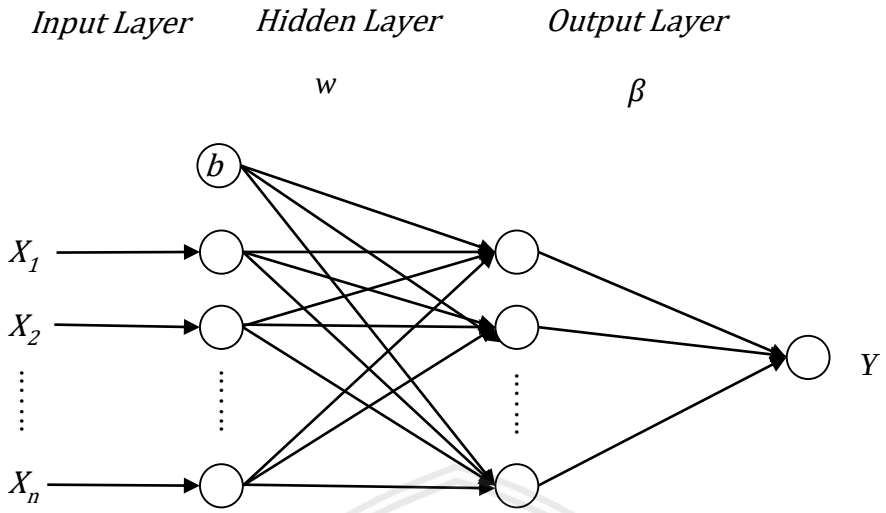
Gambar 4.18 Diagram Alir Perhitungan *Mean Absolute Percentage Error* (MAPE)

Langkah-langkah diagram alir perhitungan *Mean Absolute Percentage Error* (MAPE) berdasarkan Gambar 4.18 adalah sebagai berikut:

1. Sistem menerima masukkan berupa nilai aktual dan nilai aktual hasil prediksi.
2. Sistem melakukan perulangan sebanyak jumlah baris pada nilai aktual
3. Sistem melakukan perhitungan untuk mendapatkan nilai MAPE.
4. Keluaran sistem adalah nilai MAPE.

4.3 Perhitungan Manual

Pada bagian ini, akan dijelaskan mengenai perhitungan manual dalam algoritme *Extreme Learning Machine* (ELM). Tujuan dari manualisasi ini adalah untuk membuktikan apakah perancangan algoritme yang telah dirancang sebelumnya sudah sesuai. Dalam perhitungan manual ini terdapat langkah-langkah yang sudah didefinisikan sebelumnya pada Gambar 3.2. Struktur perhitungan ELM yang digunakan pada perhitungan manual ini adalah sebagai berikut.



Gambar 4.19 Arsitektur ELM

Perhitungan manual yang dilakukan dalam penelitian ini menggunakan 15 sampel data dengan jumlah fitur sebanyak 7. Fitur yang digunakan dalam penelitian ini adalah nilai *Bitcoin* pada beberapa hari sebelum hari target. Pembagian data dalam perhitungan manual ini adalah 80% data dijadikan sebagai data latih dan 20% data sebagai data uji. Jumlah *hidden neuron* yang digunakan sebanyak 2, serta fungsi aktivasi yang digunakan dalam perhitungan manual ini adalah fungsi aktivasi sigmoid biner. Data yang digunakan dalam perhitungan manual ini ditunjukkan pada Tabel 4.2.

Tabel 4.2 Tabel Data Latih dan Data Uji

Data Nilai *Cryptocurrency Bitcoin*								
No.	X1	X2	X3	X4	X5	X6	X7	Y
1	105690000	106102000	107510000	106566000	107095000	107453000	107350000	107041000
2	106102000	107510000	106566000	107095000	107453000	107350000	107041000	107200000
3	107510000	106566000	107095000	107453000	107350000	107041000	107200000	100000000
4	106566000	107095000	107453000	107350000	107041000	107200000	100000000	99338000
5	107095000	107453000	107350000	107041000	107200000	100000000	99338000	94779000
6	107453000	107350000	107041000	107200000	100000000	99338000	94779000	91289000
7	107350000	107041000	107200000	100000000	99338000	94779000	91289000	96424000
8	107041000	107200000	100000000	99338000	94779000	91289000	96424000	93737000
9	107200000	100000000	99338000	94779000	91289000	96424000	93737000	94606000
10	100000000	99338000	94779000	91289000	96424000	93737000	94606000	93850000
11	99338000	94779000	91289000	96424000	93737000	94606000	93850000	96500000
12	94779000	91289000	96424000	93737000	94606000	93850000	96500000	96145000

13	91289000	96424000	93737000	94606000	93850000	96500000	96145000	96300000
14	96424000	93737000	94606000	93850000	96500000	96145000	96300000	95782000
15	93737000	94606000	93850000	96500000	96145000	96300000	95782000	87214000

Pada Tabel 4.2, kolom X1, X2, …, Xn adalah nilai *Bitcoin* pada 7 hari sebelum nilai target (Y).

4.3.1 Perhitungan Normalisasi Data

Pada perhitungan normalisasi data, data yang digunakan adalah data pada Tabel 4.2. Pada proses ini, normalisasi data dilakukan menggunakan Min-Max Normalization. Dibawah ini akan dijabarkan perhitungan manual untuk normalisasi data berdasarkan Persamaan 2.4.

Langkah 1: Tentukan nilai maksimum dan minimum. Untuk nilai maksimum akan ditambahkan 10000000 sementara untuk nilai minimum akan dikurangi 10000000. Tujuannya untuk mengantisipasi adanya data baru yang berada diluar nilai maksimum ataupun minimum pada data asli. Dibawah ini adalah hasil nilai maksimum dan minimum yang ditunjukkan pada Tabel 4.3.

Tabel 4.3 Tabel Nilai Maksimum dan Minimum

MAX	MIN
117510000	77214000

Langkah 2: Lakukan normalisasi data untuk mendapatkan nilai yang akan digunakan untuk proses selanjutnya. Contoh perhitungan normalisasi adalah sebagai berikut:

$$x'_{1,1} = \frac{x_{1,1} - min}{max - min} = \frac{105690000 - 77214000}{117510000 - 77214000} = 0{,}706671$$

Seluruh data yang sudah dinormalisasi kemudian digunakan sebagai masukan yang membentuk suatu pola. Hasil perhitungan normalisasi dapat dilihat pada Tabel 4.4.

Tabel 4.4 Tabel Normalisasi Data

Data Nilai *Cryptocurrency Bitcoin*								
No.	X1	X2	X3	X4	X5	X6	X7	Y
1	0.706671	0.716895	0.751836	0.72841	0.741538	0.750422	0.747866	0.740198
2	0.716895	0.751836	0.72841	0.741538	0.750422	0.747866	0.740198	0.744143
3	0.751836	0.72841	0.741538	0.750422	0.747866	0.740198	0.744143	0.565466
4	0.72841	0.741538	0.750422	0.747866	0.740198	0.744143	0.565466	0.549037
5	0.741538	0.750422	0.747866	0.740198	0.744143	0.565466	0.549037	0.435899

6	0.750422	0.747866	0.740198	0.744143	0.565466	0.549037	0.435899	0.34929
7	0.747866	0.740198	0.744143	0.565466	0.549037	0.435899	0.34929	0.476722
8	0.740198	0.744143	0.565466	0.549037	0.435899	0.34929	0.476722	0.410041
9	0.744143	0.565466	0.549037	0.435899	0.34929	0.476722	0.410041	0.431606
10	0.565466	0.549037	0.435899	0.34929	0.476722	0.410041	0.431606	0.412845
11	0.549037	0.435899	0.34929	0.476722	0.410041	0.431606	0.412845	0.478608
12	0.435899	0.34929	0.476722	0.410041	0.431606	0.412845	0.478608	0.469798
13	0.34929	0.476722	0.410041	0.431606	0.412845	0.478608	0.469798	0.473645
14	0.476722	0.410041	0.431606	0.412845	0.478608	0.469798	0.473645	0.46079
15	0.410041	0.431606	0.412845	0.478608	0.469798	0.473645	0.46079	0.248164

4.3.2 Inisialisasi Bobot Masukan

Setelah dilakukan proses normalisasi data, maka langkah selanjutnya adalah inisialisasi bobot masukan sesuai dengan jumlah *hidden neuron* dan jumlah hari yaitu 2 dan 7. Penentuan nilai bobot masukan dilakukan secara acak dalam bentuk matriks dengan ordo jumlah *hidden neuron* x jumlah hari, sehingga matriks pada kasus ini memiliki ordo 2x7 dengan jumlah data sebanyak 12. Nilai bobot masukan yang akan tampil memiliki rentang yang sudah ditentukan yaitu [-1,1] yang ditunjukkan pada Tabel 4.6.

Tabel 4.5 Tabel Inisialisasi Bobot Masukan

Bobot Masukan						
X1	X2	X3	X4	X5	X6	X7
-0.21	-0.15	0.25	-0.75	0.17	-0.19	-0.93
0.42	-0.52	0.5	0.08	0.08	0.71	-0.28

Setelah mendapatkan nilai bobot masukan, selanjutnya adalah melakukan proses *transpose* terhadap matriks bobot masukan yang bertujuan untuk mendapatkan matriks baru sehingga dapat digunakan untuk proses *training*. Sebelumnya untuk matriks bobot masukan memiliki ordo 2 x 7, untuk matriks *transpose* bobot masukan memiliki ordo 7 x 2 yang ditunjukkan pada Tabel 4.6.

Tabel 4.6 Tabel Matriks *Transpose* Bobot Masukan

Transpose Bobot Masukan		
X1	-0.21	0.42
X2	-0.15	-0.52

X3	0.25	0.5
X4	-0.75	0.08
X5	0.17	0.08
X6	-0.19	0.71
X7	-0.93	-0.28

4.3.3 Inisialisasi Bias Masukan

Langkah selanjutnya setelah mendapatkan matriks transpose bobot masukan, kita harus melakukan inisialisasi bias masukan yang didapatkan secara acak. Bias masukan ini berbentuk matriks dengan ordo 1 x jumlah *hidden neuron*. Sama halnya dengan proses pada bobot masukan, nilai bias masukan memiliki rentang [-1,1] yang ditunjukkan pada Tabel 4.7.

Tabel 4.7 Tabel Inisialisasi Bias Masukan

Bias	
0.35	-0.52

4.3.4 Perhitungan Proses *Training*

Langkah 1: Menghitung nilai H_{init} sesuai dengan Persamaan 2.5. Dibawah ini adalah contoh untuk melakukan perhitungannya.

$$Hinit_{1,1} = \big((0.706671 * -0.21) + (0.716895 * -0.15) + (0.751836 * 0.25) + (0.72841 * -0.75) + (0.741538 * 0.17) + (0.750422 * -0.19) + (0.747866 * -0.93)\big) = -1.32632$$

Tabel 4.8 *Tabel Matriks Hinit Proses Training*

Matriks Hinit	
-1.32632	0.740927
-1.34028	0.717432
-1.35013	0.74481
-1.17893	0.7846
-1.12801	0.661657
-1.0564	0.668919
-0.82047	0.60212
-0.97309	0.39997
-0.84329	0.579493
-0.75235	0.406306

-0.84715	0.440358
-0.78246	0.466249

Langkah 2: Setelah mendapatkan matriks Hinit, selanjutnya adalah mendapatkan matriks H menggunakan fungsi aktivasi. Untuk fungsi aktivasi sendiri menggunakan fungsi aktivasi sigmoid biner. Dibawah ini akan diberikan contoh perhitungannya.

$$H(x)_{1,1} = \frac{1}{1 + e^{(-(Hinit+bias))}} = \frac{1}{1 + e^{(-(-1.32632+0.35))}} = 0.273623$$

Hasil perhitungan nilai H ditunjukkan pada Tabel 4.9.

Tabel 4.9 Tabel Matriks H Proses *Training*

Matriks H	
0.273623	0.555008
0.270857	0.549198
0.268915	0.555967
0.303872	0.565767
0.31475	0.535355
0.330395	0.537161
0.384505	0.520518
0.349079	0.470028
0.379119	0.514869
0.400747	0.471607
0.378212	0.4801
0.39354	0.486566

Langkah 3: Setelah mendapatkan matriks *H*, langkah selanjutnya adalah mendapatkan matriks *Moore-Penrose Pseudo Inverse* sesuai Persamaan 2.7. Pada bagian ini akan dibagi menjadi beberapa langkah-langkah yang lebih spesifik.

Langkah 3.1: Pada langkah ini, diperlukan matriks *H* yang telah di*transpose*. Sehingga matriks yang terbentuk dalam kasus ini memiliko ordo 2 x 12. Matriks H transpose ditunjukkan pada Tabel 4.10.

Tabel 4.10 Tabel Matriks H *Transpose*

***Transpose* H**				
1	2	3	……..	12
0.273623	0.270856643	0.268915318	……..	0.39354
0.555008	0.549198295	0.555967024	……..	0.486566

Langkah 3.2: Pada langkah ini, dilakukan perkalian antara matriks H transpose dengan matriks H yang menghasilkan matriks dengan ordo 2 x 2 pada kasus ini. Contoh perhitungan akan ditunjukkan dibawah. Untuk hasil perkalian matriks ditunjukkan pada Tabel 4.11.

$$(H^T H)_{1,1} = \big((0.273623 * 0.273623) + (0.270856 * 0.270856) + (0.268915 * 0.268915) + \cdots + (0.39354 * 0.39354)\big) = 1.393062$$

Tabel 4.11 Tabel Hasil Perkalian Matriks H *Transpose* dengan Matriks H

Matriks H Transpose x Matriks H	
1.393062	2.08949692
2.089497	3.260602945

Langkah 3.3: Langkah selanjutnya adalah melakukan *inverse* terhadap matriks hasil perkalian matriks H transpose dengan matriks H yang memiliki ordo 2 x 2 menggunakan Operasi Baris Elementer (OBE). Berikut adalah langkah-langkah perhitungan untuk mendapatkan matriks *inverse* menggunakan OBE. Hasil perhitungan ditunjukkan pada Tabel 4.12.

1. Bentuk Matriks Identitas (*I*)

$$[H|I] = \begin{bmatrix} 1.393062 & 2.089497 \\ 2.089497 & 3.260602 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

2. Tukar baris 2 dengan baris 1

$$[H|I] = \begin{bmatrix} 2.089497 & 3.260602 \\ 1.393062 & 2.089497 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

3. $R_1 = R_1 \ / \ 2.089497$

$$[H|I] = \begin{bmatrix} 1 & 1.561 \\ 1.393062 & 2.089497 \end{bmatrix} \begin{bmatrix} 0 & 0.479 \\ 1 & 1 \end{bmatrix}$$

4. $R_2 = R_2 - (R_1 * 1.393062)$

$$[H|I] = \begin{bmatrix} 1 & 1.561 \\ 0 & -0.085 \end{bmatrix} \begin{bmatrix} 0 & 0.479 \\ 1 & -0.067 \end{bmatrix}$$

5. $R_2 = R_2 / -0.085$

$$[H|I] = \begin{bmatrix} 1 & 1.561 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 0.479 \\ -11.857 & 7.905 \end{bmatrix}$$

6. $R_1 = R_1 - (R_2 * 1.561)$

$$[H|I] = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 18.505 & -11.857 \\ -11.857 & 7.905 \end{bmatrix}$$

Tabel 4.12 Tabel *Inverse* Matriks

Matriks H Transpose x Matriks H	
18.50258	-11.857
-11.857	7.905053

Langkah 3.4: Langkah selanjutnya adalah menghitung matriks *Moore-Penrose Pseudo Inverse* sesuai Persamaan 2.7. Hasil dari perkalian ini adalah matriks dengan ordo 2 x 12. Contoh perhitungan akan ditunjukkan dibawah ini beserta hasil yang ditunjukkan pada Tabel 4.13.

$$H^{+}_{1,1} = \big((18.505 * 0.273623) + (-11.857 * 0.555008)\big) = -1.51802$$

Tabel 4.13 Tabel Matriks *Moore-Penrose Pseudo Inverse*

Matriks *Moore-Penrose Pseudo Inverse*				
1	2	3	.....	12
-1.51802	-1.50031678	-1.616493363	.....	1.512283
1.143011	1.129884765	1.206410285	.....	-0.81989

Langkah 4: Langkah selanjutnya adalah menghitung *output weight* berdasarkan Persamaan 2.8. Pada kasus ini, matriks yang terbentuk memiliki ordo 2 x 1. Contoh perhitungan ditunjukkan dibawah ini serta hasil *output weight* ditunjukkan pada Tabel 4.14.

$$\begin{aligned} \beta_{1,1} = \big((-1.51802 * 0.740198) + (-1.500316 * 0.744143) \\ + (-1.616493 * 0.565466) + \cdots + (1.512283 * 0.469798)\big) \\ = -0.77517 \end{aligned}$$

Tabel 4.14 Matriks *Output Weight*

$\boldsymbol{\beta}$
-0.77517
1.472541

Langkah 5: Langkah terakhir adalah menghitung nilai output layer sesuai dengan Persamaan 2.9. Hasil akhir adalah matriks dengan ordo 12 x 1. Untuk contoh perhitungan akan ditunjukkan dibawah, dan hasil perhitungan akan ditunjukkan pada Tabel 4.15.

$$Y_{1,1} = \big((0.273623 * -0.77517) + (0.555008 * 1.472541)\big) = 0.605168$$

Tabel 4.15 Tabel Nilai *Output Layer* Proses *Training*

Y
0.605168
0.598757
0.610229
0.597562
0.544347
0.534879
0.468427
0.42154
0.464284
0.383813
0.413788
0.411427

4.3.5 Perhitungan Proses *Testing*

Proses *testing* dilakukan untuk mendapatkan nilai *output layer* berdasarkan bobot masukan yang dilakukan saat inisialisasi bobot masukan dan *output weight* yang didapatkan saat proses *training*.

Langkah 1: Menghitung nilai H_{init} sesuai dengan Persamaan 2.5. Dibawah ini adalah contoh untuk melakukan perhitungannya. Hasil perhitungan ditunjukkan pada Tabel 4.16.

$$\begin{aligned} Hinit_{1,1} = \big((0.34929 * -0.21) + (0.476722 * -0.15) + (0.410041 * 0.25) \\ + (0.431606 * -0.75) + (0.412845 * 0.17) \\ + (0.478608 * -0.19) + (469798 * -0.93)\big) = -0.82372 \end{aligned}$$

Tabel 4.16 Tabel Matriks *Hinit* Proses *Testing*

Matriks Hinit	
-0.82372	0.379651
-0.81174	0.475058

-0.84526	0.437344

Langkah 2: Langkah selanjutnya sama dengan langkah 2 pada proses *training*, yaitu mendapatkan matriks H menggunakan fungsi aktivasi. Contoh perhitungan manual akan ditunjukkan dibawah ini. Untuk hasil perhitungan ditunjukkan pada Tabel 4.17.

$$H(x)_{1,1} = \frac{1}{1 + e^{(-(Hinit+bias))}} = \frac{1}{1 + e^{(-(-0.82372+0.35))}} = 0.383737$$

Tabel 4.17 Tabel Matriks H Proses *Testing*

Matriks H	
0.383737	0.46497
0.386574	0.488766
0.378656	0.479348

Langkah 3: Setelah mendapatkan matriks H, langkah selanjutnya adalah mendapatkan nilai *output layer*. Untuk kasus ini, matriks pada proses *testing* memiliki ordo 3 x 1. Contoh perhitungan akan ditunjukkan dibawah ini, dan hasil perhitungan ditunjukkan pada Tabel 4.18.

$$Y_{1,1} = \big((0.383737 * -0.77517) + (0.46497 * 1.472541)\big) = 0.387226$$

Tabel 4.18 Tabel Nilai *Output Layer* Proses *Testing*

Y
0.387226
0.420068
0.412336

4.3.6 Perhitungan Denormalisasi Data

Pada bagian ini, akan dijelaskan mengenai perhitungan manual denormalisasi data yang dilakukan baik pada proses *training* maupun proses *testing*. Perhitungan manual ini sesuai dengan Persamaan 2.10. Nilai *output layer* pada proses *testing* yang akan dijadikan contoh dalam perhitungan manual denormalisasi data yang akan ditunjukkan dibawah ini, lalu untuk hasil denormalisasi data ditunjukkan pada Tabel 4.19.

$$x_{1,1} = 0.387226 * (1117510000 - 77214000) + 77214000 = 92817665.7$$

Tabel 4.19 Tabel Denormalisasi Data

Y **Denormalisasi**
92817665.67

94141049.34
93829479.62

4.3.7 Perhitungan Nilai *Mean Absolute Percentage Error* (MAPE)

Pada bagian ini, akan dijelaskan mengenai perhitungan manual mendapatkan nilai *Mean Absolute Percentage Error* (MAPE) yang dilakukan baik pada proses *training* maupun proses *testing*. Perhitungan manual ini sesuai dengan Persamaan 2.11. Berikut ini adalah contoh perhitungan dan hasil pada proses *testing* dibawah ini.

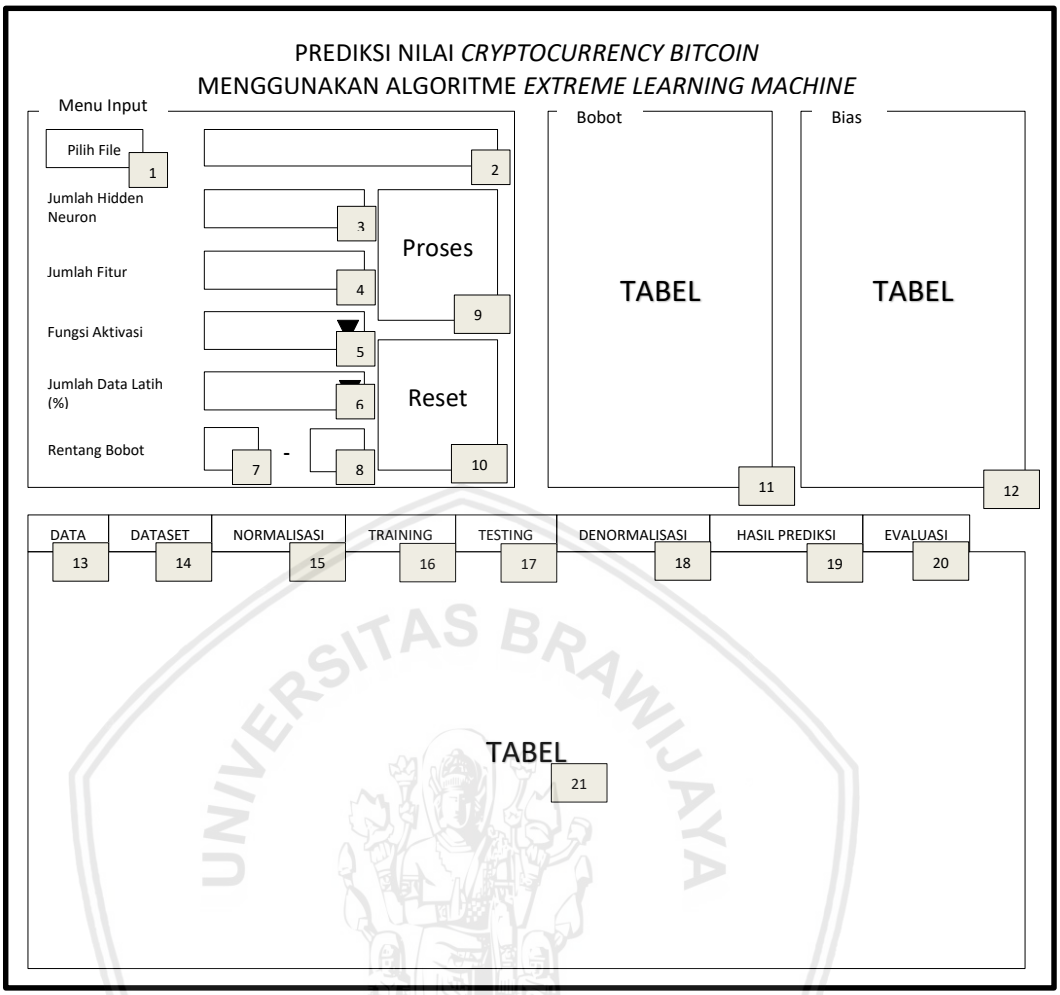
$$MAPE = \left(abs\left(\frac{92817665.7 - 96300000}{96300000}\right)x\ 100\right) + \left(abs\left(\frac{94141049.3\ - 95782000}{95782000}\right)x\ 100\right) + \left(abs\left(\frac{93829479.6\ - 87214000}{87214000}\right)x\ 100\right) = \frac{12.9147}{3} = 4.3049$$

4.4 Perancangan Antarmuka

Gambaran implementasi sistem akan dilakukan pada subbab ini agar dapat dijadikan acuan pada proses implementasi. Gambaran implementasi terdiri dari 8 subhalaman yang terdiri dari halaman data, dataset, normalisasi, *training*, *testing*, denormalisasi, hasil prediksi dan evaluasi.

4.4.1 Perancangan Halaman Utama

Antarmuka halaman utama digunakan untuk memasukkan parameter-paramater untuk melakukan proses prediksi nilai *cryptocurrency Bitcoin* menggunakan algoritme *Extreme Learning Machine* (ELM). Rancangan antarmuka halaman utama dapat dilihat di Gambar 4.20.



Gambar 4.20 Perancangan Antarmuka Halaman Utama

Keterangan rancangan antarmuka halaman utama pada Gambar 4.20 adalah sebagai berikut:

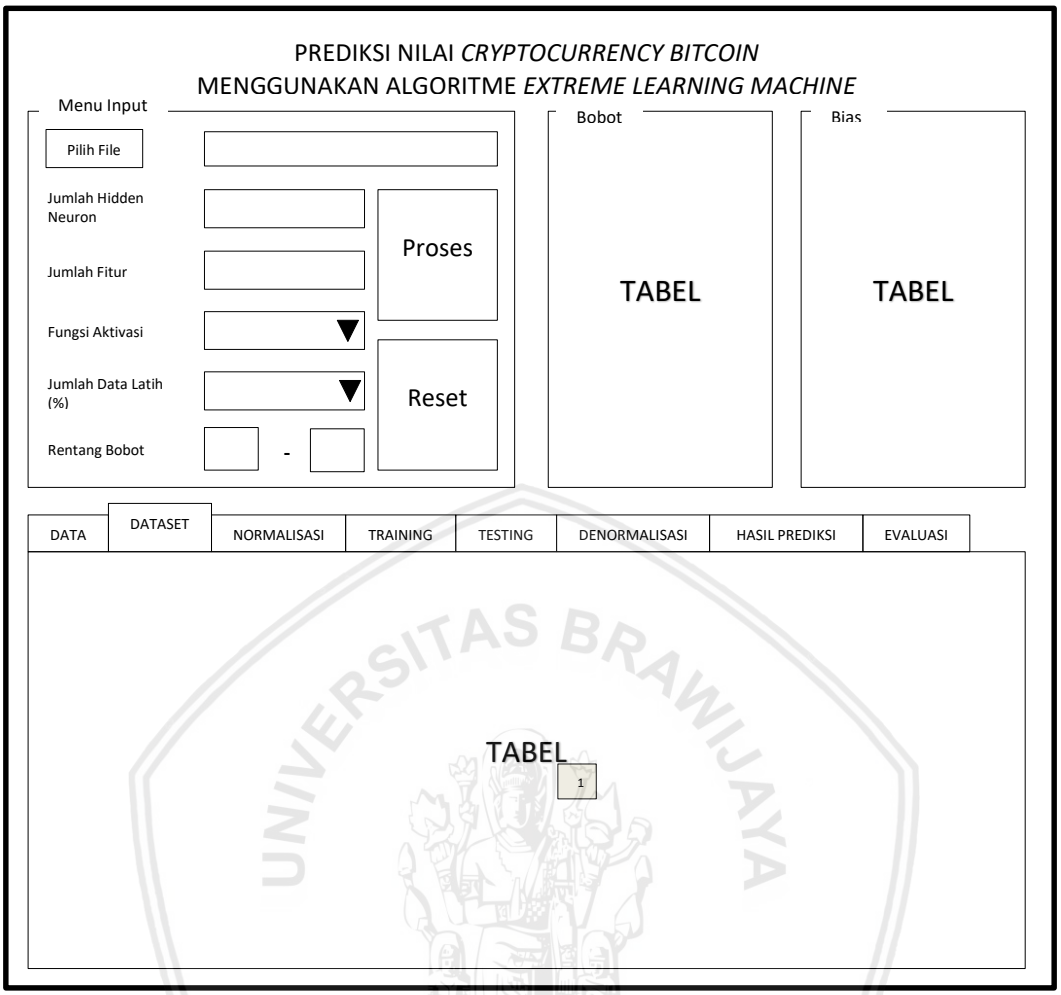
1. Tombol pilih file berformat *excel* untuk memilih file yang akan digunakan untuk proses prediksi.
2. *Textbox* nama file untuk menampilkan *path* file yang sudah dipilih.
3. *Textbox* jumlah *hidden neuron* untuk memasukkan jumlah hidden neuron yang diinginkan.
4. *Textbox* jumlah fitur unruk memasukkan jumlah fitur.
5. *Dropdownlist* fungsi aktivasi untuk memilih fungsi aktivasi yang diinginkan.
6. *Dropdownlist* jumlah data latih untuk memilih persentasi jumlah data latih yang diinginkan.
7. *Textbox* untuk memasukkan rentang bobot minimum.
8. *Textbox* untuk memasukkan rentang bobot maksimum.
9. Tombol proses untuk melakukan proses prediksi.

10. Tombol *reset* untuk menyetel ulang seluruh komponen yang ada di dalam sistem.
11. Tabel bobot untuk menampilkan nilai bobot.
12. Tabel bias untuk menampilkan nilai bias.
13. *Tab page* data untuk melihat data yang akan digunakan.
14. *Tab page dataset* untuk melihat dataset yang telah dibuat berdasarkan data sebelumnya.
15. *Tab page* normalisasi untuk melihat hasil normalisasi.
16. *Tab page training* untuk melihat hasil training.
17. *Tab page testing* untuk melihat hasil testing.
18. *Tab page denormalisasi* untuk melihat hasil denormalisasi.
19. *Tab page* hasil prediksi untuk melihat hasil prediksi.
20. *Tab page* evaluasi untuk melihat nilai kesalahan menggunakan MAPE.
21. Tabel untuk menampilkan data yang akan digunakan untuk prediksi.

4.4.2 Perancangan Sub Halaman *Dataset*

Antarmuka sub halaman *dataset* digunakan untuk menampilkan data yang sudah diubah menjadi *dataset* dengan jumlah kolom sesuai masukan dari *textbox* jumlah fitur. Rancangan sub halaman dataset ditunjukkan pada

Gambar 4.21.



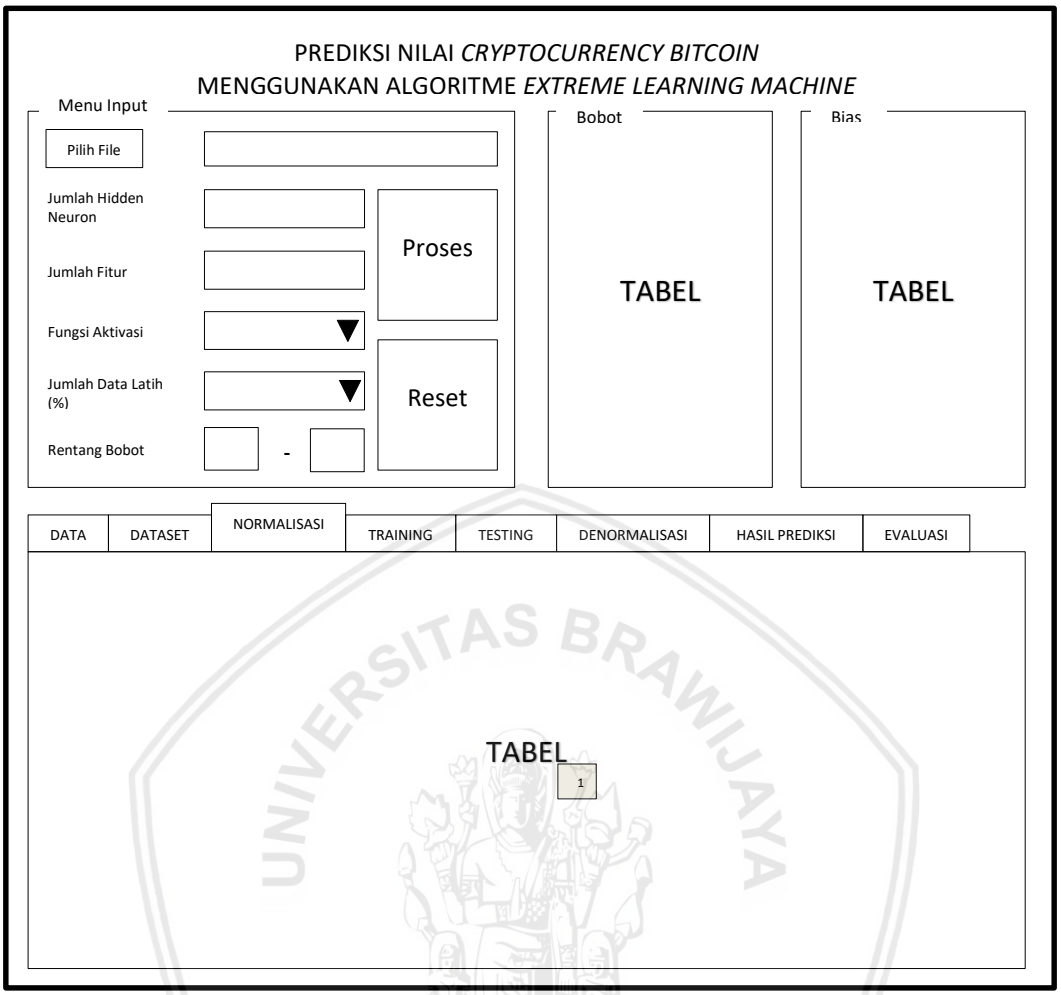
Gambar 4.21 Perancangan Antarmuka Sub Halaman *Dataset*

Keterangan rancangan antarmuka sub halaman *dataset* pada

Gambar 4.21 adalah sebuah tabel untuk menampilkan data yang sudah diubah menjadi dataset.

4.4.3 Perancangan Sub Halaman Normalisasi

Antarmuka sub halaman normalisasi digunakan untuk menampilkan *dataset* yang sudah dilakukan proses normalisasi. Rancangan antarmuka sub halaman normalisasi ditunjukkan pada Gambar 4.22.

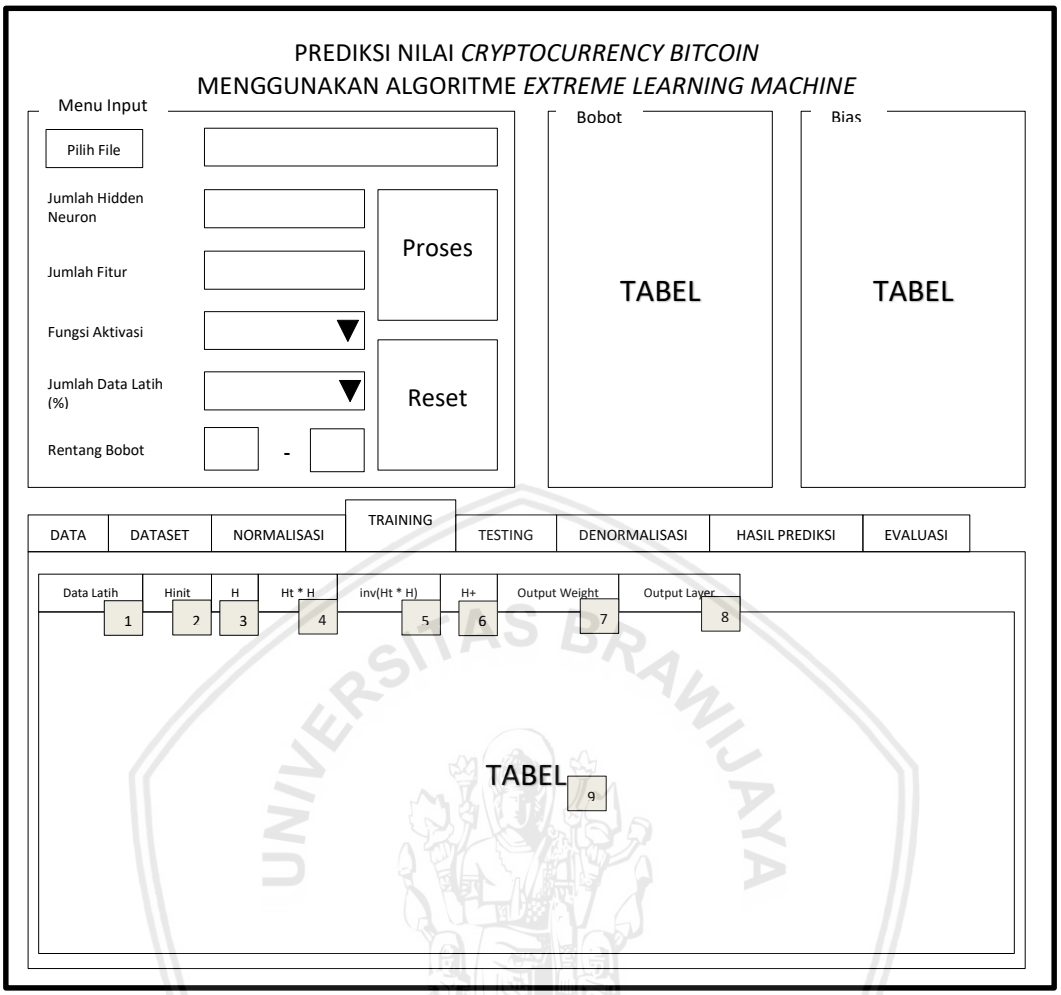


Gambar 4.22 Perancangan Antarmuka Sub Halaman Normalisasi

Keterangan rancangan antarmuka sub halaman normalisasi pada Gambar 4.22 adalah sebuah tabel untuk menampilkan *dataset* yang sudah dilakukan proses normalisasi.

4.4.4 Perancangan Sub Halaman *Training*

Antarmuka sub halaman *training* digunakan untuk menampilkan setiap hasil pada proses *training* yang terdiri dari data latih, H_{init}, H, Ht * H, inv(Ht * H), H+, *output weight*, dan *output layer*. Rancangan antarmuka sub halaman *training* ditunjukkan pada Gambar 4.23.



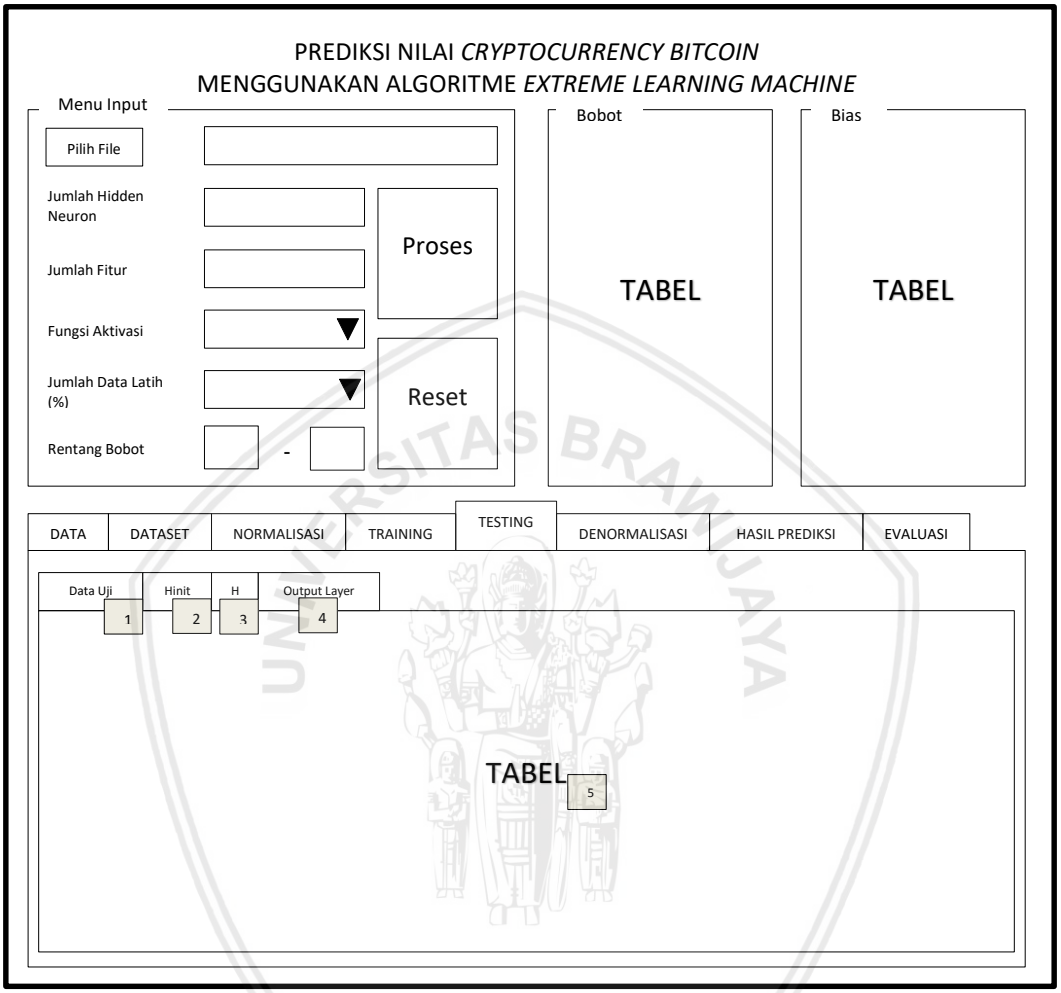
Gambar 4.23 Perancangan Antarmuka Sub Halaman *Training*

Keterangan rancangan antarmuka sub halaman *training* pada Gambar 4.23 adalah sebagai berikut:

1. *Tab page* data latih untuk menampilkan data latih yang digunakan pada proses *training.*
2. *Tab page* Hinit untuk menampilkan data hasil proses Hinit.
3. *Tab page* H untuk menampilkan data hasil proses H.
4. *Tab page* Ht * H untuk menampilkan data hasil proses Ht * H.
5. *Tab page* inv(Ht * H) untuk menampilkan data hasil proses inv (Ht * H).
6. *Tab page* H+ untuk menampilkan data hasil proses H+.
7. *Tab page output weight* untuk menampilkan data hasil proses *output weight*.
8. *Tab page output layer* untuk menampilkan data hasil proses *output layer*.
9. Tabel yang terdapat pada setiap tab page untuk menampilkan data.

4.4.5 Perancangan Sub Halaman *Testing*

Antarmuka sub halaman *testing* digunakan untuk menampilkan setiap hasil pada proses *testing* yang terdiri dari data *testing*, Hinit, H, dan *output layer*. Rancangan antarmuka sub halaman *testing* ditunjukkan pada Gambar 4.24.



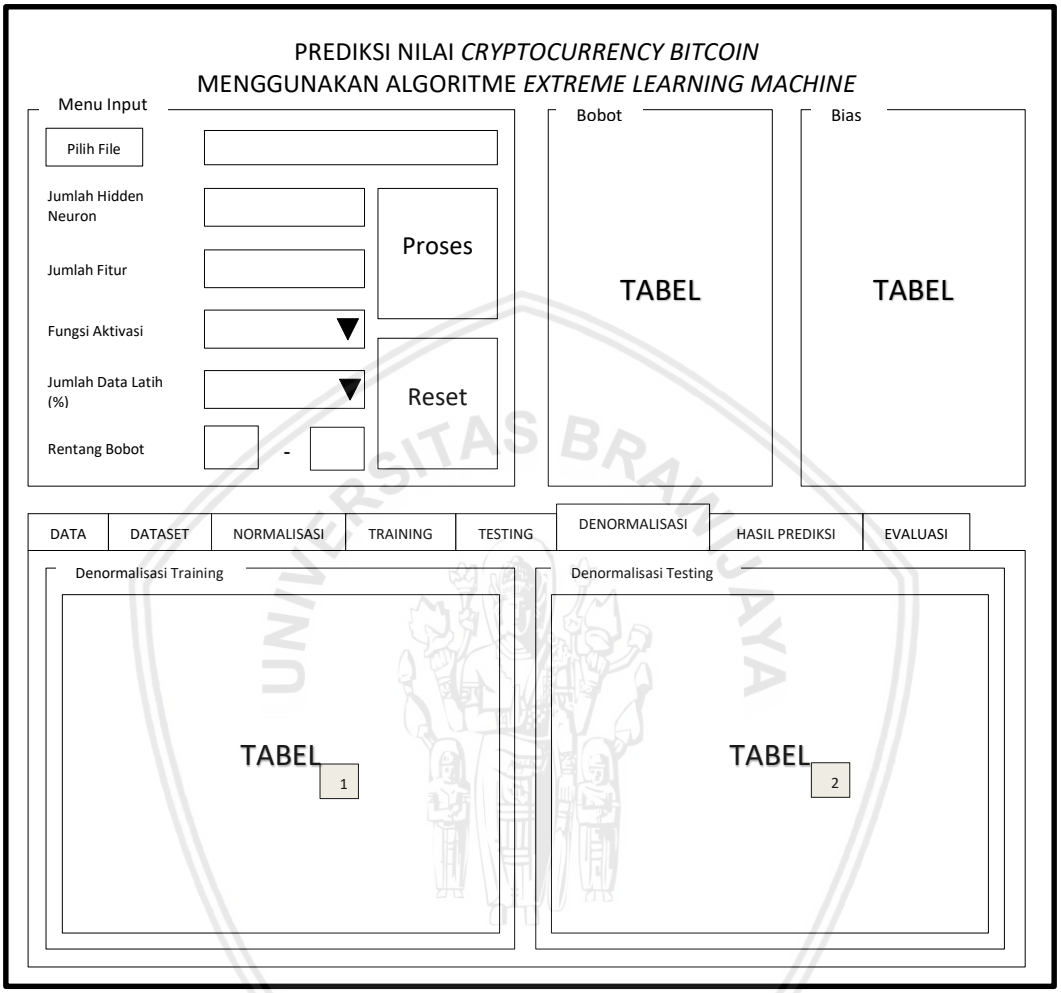
Gambar 4.24 Perancangan Antarmuka Sub Halaman *Testing*

Keterangan rancangan antarmuka sub halaman *training* pada Gambar 4.24 adalah sebagai berikut:

1. *Tab page* data *testing* untuk menampilkan data latih yang digunakan pada proses *testing.*
2. *Tab page* Hinit untuk menampilkan data hasil proses Hinit.
3. *Tab page* H untuk menampilkan data hasil proses H.
4. *Tab page output layer* untuk menampilkan data hasil proses *output layer.*
5. Tabel yang terdapat pada setiap *tab page* untuk menampilkan data.

4.4.6 Perancangan Antarmuka Sub Halaman Denormalisasi

Antarmuka sub halaman denormalisasi digunakan untuk menampilkan hasil denormalisasi baik pada proses *training* maupun proses *testing*. Rancangan antarmuka sub halaman denormalisasi ditunjukkan pada Gambar 4.25.



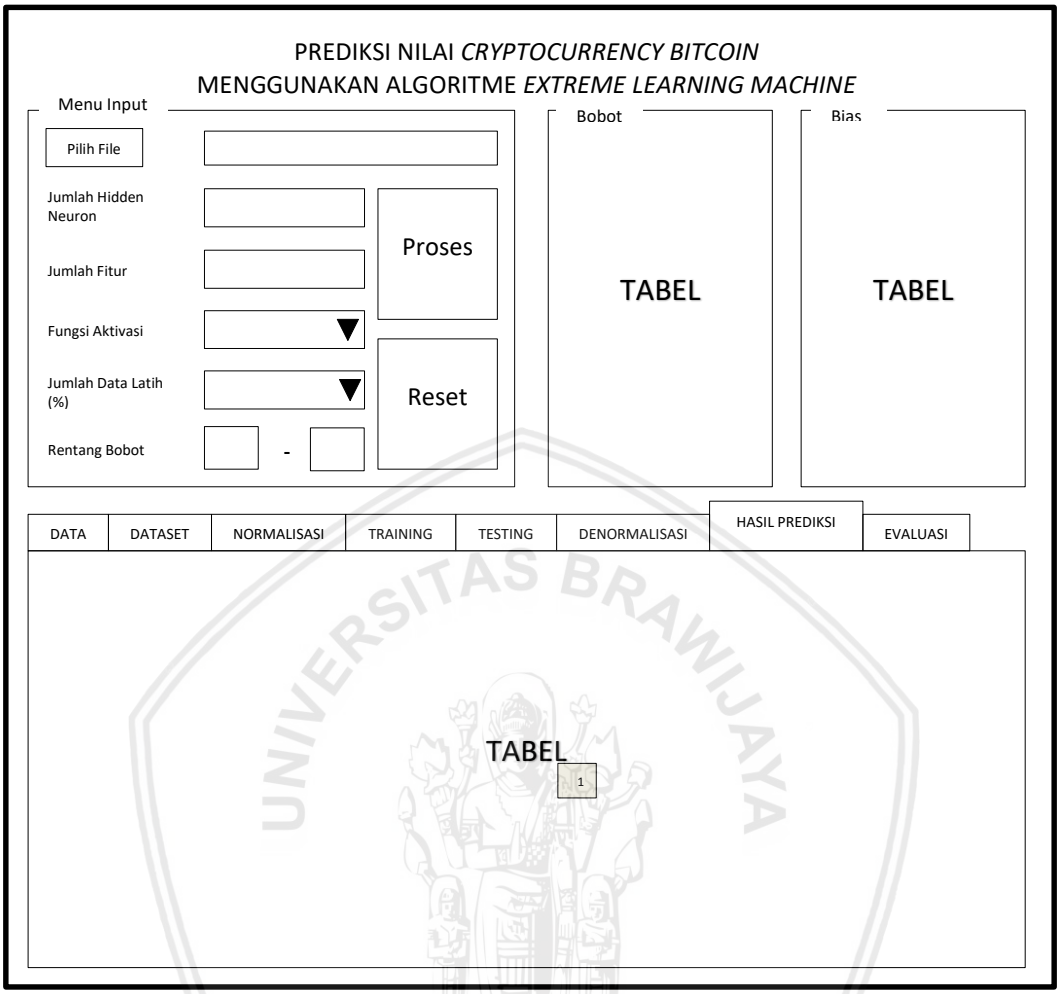
Gambar 4.25 Perancangan Antarmuka Sub Halaman Denormalisasi

Keterangan rancangan antarmuka sub halaman denormalisasi pada Gambar 4.25 adalah sebagai berikut:

1. Tabel denormalisasi *training* untuk menampilkan data pada proses *training* yang telah didenormalisasi.
2. Tabel denormalisasi *testing* untuk menampilkan data pada proses *testing* yang telah didenormalisasi.

4.4.7 Perancangan Antarmuka Sub Halaman Hasil Prediksi

Antarmuka sub halaman hasil prediksi digunakan untuk menampilkan hasil prediksi data baru yang sebelumnya belum diketahui hasilnya baik pada proses *training* maupun *testing*. Rancangan antarmuka sub halaman hasil prediksi ditunjukkan pada Gambar 4.26.

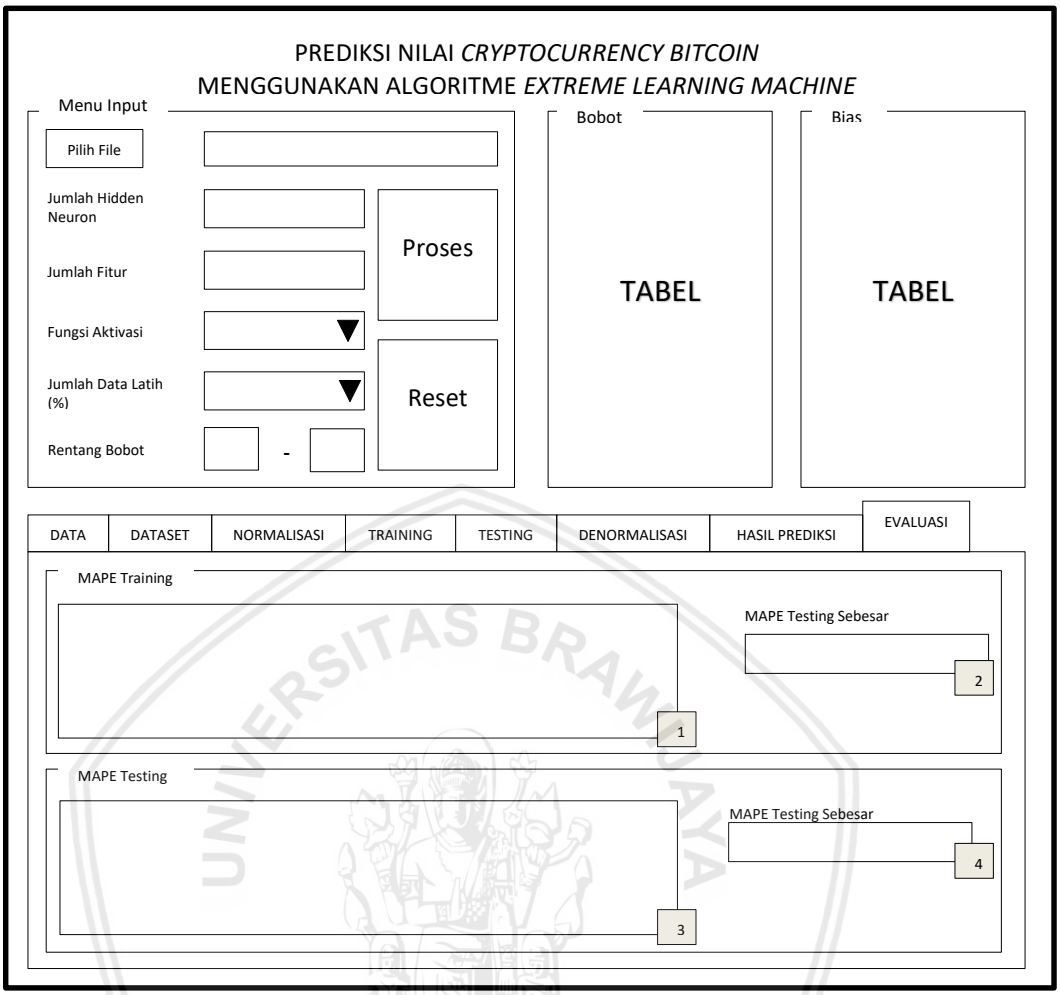


Gambar 4.26 Perancangan Antarmuka Sub Halaman Hasil Prediksi

Keterangan rancangan antarmuka sub halaman hasil prediksi pada Gambar 4.26 adalah sebuah tabel untuk menampilkan hasil prediksi data baru.

4.4.8 Perancangan Antarmuka Sub Halaman Evaluasi

Antarmuka sub halaman evaluasi digunakan untuk menampilkan hasil perhitungan evaluasi pada proses training dan testing menggunakan metode MAPE. Rancangan antarmuka sub halaman evaluasi ditunjukkan pada Gambar 4.27.



Gambar 4.27 Perancangan Antarmuka Sub Halaman Evaluasi

Keterangan rancangan antarmuka sub halaman evaluasi pada Gambar 4.27 adalah sebagai berikut:

1. Tabel untuk menampilkan hasil evaluasi pada proses *training*.
2. Tabel untuk menampilkan hasil evaluasi pada proses *testing*.
3. *Textbox* untuk menampilkan nilai tingkat kesalahan pada proses *training*.
4. *Textbox* untuk menampilkan nilai tingkat kesalahan pada proses *testing*.

4.5 Perancangan Pengujian

Pada perancangan pengujian ini, ada beberapa parameter pada algoritme Extreme Learning Machine (ELM) yang akan diuji pada penelitian ini. Parameter yang akan diuji pada penelitian ini adalah sebagai berikut:

1. Pengujian perbandingan jumlah fitur.
2. Pengujian perbandingan jumlah *hidden neuron*.
3. Pengujian rentang bobot masukan.
4. Pengujian perbandingan rasio data latih dan data uji.

4.5.1 Pengujian Pengaruh Jumlah Fitur

Pada pengujian ini, penulis ingin mengetahui pengaruh banyaknya jumlah fitur terhadap hasil prediksi serta nilai tingkat kesalahan yang dihasilkan. Pengujian dilakukan sebanyak 10 kali untuk setiap jumlah fitur. Jumlah fitur yang digunakan sebanyak 2 – 10 fitur. Lalu didapatkan rerata nilai kesalahan per jumlah fitu. Tabel perancangan pengujian perbandingan jumlah fitur ditunjukkan pada Tabel 4.20.

Tabel 4.20 Perancangan Pengujian Pengaruh Jumlah Fitur

Jumlah Fitur	Nilai MAPE							Rerata MAPE
	Percoban ke-							
	1	2	3	4	5	……..	10	
2								
3								
4								
5								
…….								
10								

4.5.2 Pengujian Pengaruh Jumlah *Hidden Neuron*

Pada pengujian ini, penulis ingin mengetahui pengaruh banyaknya jumlah *hidden neuron* terhadap hasil prediksi serta nilai tingkat kesalahan yang dihasilkan. Pada pengujian ini, jumlah hidden neuron yang diuji sebanyak 1-10 *hidden neuron* yang masing-masing diuji sebanyak 10 kali untuk mendapatkan rerata nilai kesalahan. Tabel perancangan pengujian perbandingan jumlah *hidden neuron* ditunjukkan pada Tabel 4.21.

Tabel 4.21 Perancangan Pengujian Pengaruh Jumlah *Hidden Neuron*

Jumlah *Hidden Neuron*	Nilai MAPE							Rerata MAPE
	Percoban ke-							
	1	2	3	4	5	……..	10	
2								
3								
4								
…….								
10								

4.5.3 Pengujian Pengaruh Rentang Bobot Masukan

Pada pengujian ini, peneliti ingin mengetahui rentang bobot masukan terbaik pada algortime ELM untuk memprediksi nilai cryptocurrency *Bitcoin*. Untuk nilai rentang bobot yang akan diuji harus diatas 0 (Dudek, 2016). Lalu untuk setiap rentang bobot akan didapatkan rerata nilai tingkat kesalahan. Tabel perancangan pengujian pengaruh rentang bobot masukan ditunjukkan pada Tabel 4.22.

Tabel 4.22 Perancangan Pengaruh Perbandingan Rentang Bobot Masukan

Rentang Bobot Masukan	Nilai MAPE							Rerata MAPE
	Percoban ke-							
	1	2	3	4	5	……..	10	
[-1,1]								
[-1.1,1.1]								
[-1.2,1.2]								
…								
[-2,2]								

4.5.4 Pengujian Pengaruh Jumlah Data Latih dengan Data Uji Konstan

Pada pengujian ini, peneliti ingin menguji apakah jumlah data latih berpengaruh terhadap nilai MAPE yang dihasilkan. Pengujian jumlah data latih ini menggunakan 5 persentase jumlah data latih, yaitu 50%, 60%, 70%, 80%, 90%. Sementara untuk jumlah data uji yang digunakan adalah konstan dengan persentasi 20% dari jumlah data keseluruhan. Tabel perancangan pengujian jumlah data latih dengan data uji konstan ditunjukkan pada .

Tabel 4.23 Perancangan Pengujian Pengaruh Jumlah Data Latih dengan Data Uji Konstan

Persentase Jumlah Data Latih	Nilai MAPE							Rerata MAPE
	Percoban ke-							
	1	2	3	4	5	……..	10	
50%								
60%								
70%								
80%								

BAB 5 IMPLEMENTASI

5.1 Implementasi Program

Melalui hasil perancangan yang telah dibuat dan dibahas pada Bab 4, tahap berikutnya adalah pembahasan mengenai implementasi program sesuai dengan perancangan yang telah dibuat sebelumnya. Proses implementasi ini dilakukan menggunakan bahasa pemrograman C# menggunakan editor Visual Studio 2017.

5.1.1 Implementasi Normalisasi Data

Pada proses normalisasi, standardidasi data dilakukan terhadap seluruh data yang diolah. Langkah-langkahnya adalah mencari nilai maksimum dan minimum seluruh data ada setiap fitur. Proses normalisasi menggunakan *min-max normalization*. Proses normalisasi data dapat dilihat pada Kode Program 5.1.

Kode Program 5.1 Implementasi Proses Normalisasi Data

```
Algoritme 1: Normalisasi Data
1  public static long findMin1d(long[] data)
2  {
3      long dataMin = 0;
4      long Min = data[0];
5      for (int i = 0; i < data.GetLength(0); i++)
6      {
7          if(data[i] < Min && data[i] !=0)
8          {
9              dataMin = data[i];
10             Min = dataMin;
11         }
12         else
13         {
14             dataMin = Min;
15         }
16     }
17     dataMin -= 10000000;
18     return dataMin;
19 }
20 public static long findMax1d(long[] data)
21 {
22     long hasil = 0;
23     long dataMax = 0;
24     long Max = data[0];
25     for (int i = 0; i < data.GetLength(0); i++)
26     {
27         if(data[i] > Max)
28         {
28             dataMax = data[i];
29             Max = dataMax;
30         }
31         else
32         {
33             dataMax = Max;
34         }
35     }
36     hasil= dataMax + 10000000;
37     return hasil;
38 }
39 public static double[,] normalization2(long[,] dataLatih, long min,
40 long max)
```

```
41  {
42        double[,]    norm    =    new    double[dataLatih.GetLength(0),
43        dataLatih.GetLength(1)];
44        for (int i = 0; i < dataLatih.GetLength(0); i++)
45        {
46            for (int j = 0; j < dataLatih.GetLength(1); j++)
47            {
48                if (dataLatih[i, j] != 0)
49                {
50                    norm[i,  j]  =  (((double)dataLatih[i,j]  -  min)  /
51                    (max - min));
52                }
53            }
54        }
55        return norm;
56  }
```

Penjelasan Kode Program 5.1 mengenai implementasi proses normalisasi data adalah sebagai berikut:

1. Baris 1 s.d 19 adalah proses mencari nilai minimum dari data.
2. Baris 20 s.d 38 adalah proses mencari nilai maksimum dari data.
3. Baris 39 s.d 56 adalah proses mencari nilai normalisasi dari data berdasarkan Persamaan 2.4.

5.1.2 Implementasi Proses Inisialisasi Bobot dan *Bias*

Pada proses inisialisasi bobot dan *bias*, pembangkitan dilakukan secara acak baik untuk pembangkitan nilai bobot maupun nilai *bias*. Hasil dari pembangkitan tersebut selanjutnya digunakan untuk menghitung keluaran *hidden layer*. Bentuk dari bobot dan bias ini adalah matriks dengan ordo sesuai dengan banyaknya *input neuron* dan *hidden neuron*. Implementasi proses inisialisasi bobot dan *bias* ditunjukkan pada Kode Program 5.2.

Kode Program 5.2 Implementasi Proses Inisialisasi Bobot dan *Bias*

```
Algoritme 2: Inisialisasi Bobot dan Bias
1   public static double[] getBias(int hidden, double min, double max)
2   {
3         double[] bias = new double[hidden];
4         Random ran = new Random();
5         for (int i = 0; i < bias.GetLength(0); i++)
6         {
7             bias[i] = min + (max - min) * ran.NextDouble();
8         }
9         return bias;
10  }
11  //membuat  bobot  sesuai  dengan  jumlah  hidden  layer  dan  jumlah
12  fitur
13  public static double[,] getBobot(int hidden, double min, double max,
14  int fitur)
15  {
16        double[,] weight = new double[hidden, fitur];
17        Random ran = new Random();
18        for (int i = 0; i < weight.GetLength(0); i++)
19        {
20            for (int j = 0; j < weight.GetLength(1); j++)
21            {
22                weight[i, j] = min + (max - min) * ran.NextDouble();
23            }
```

```
24          }
25          return weight;
26      }
```

Penjelasan Kode Program 5.2 mengenai implementasi proses inisialisasi bobot dan bias adalah sebagai berikut:

1. Baris 1 s.d 9 adalah proses mendapatkan nilai bias secara acak.
2. Baris 13 s.d 26 adalah proses mendapatkan nilai bobot secara acak.

5.1.3 Implementasi Proses Menghitung Keluaran *Hidden Layer* dengan Fungsi Aktivasi

Pada proses ini, dilakukan perhitungan nilai keluaran *hidden layer* dengan fungsi aktivasi. Proses ini dilakukan baik pada proses training maupun testing dengan urutan yang sama. Proses menghitung keluaran *hidden layer* dengan fungsi aktivasi dapat dilihat pada Kode Program 5.3.

Kode Program 5.3 Implementasi Proses Menghitung Keluaran *Hidden Layer* dengan Fungsi Aktivasi

```
Algoritme 3: Menghitung Keluaran Hidden Layer dengan Fungsi Aktivasi
1   public static double[,] transpose(double[,] data)
2   {
3       double[,] transpose = new double[data.GetLength(1),
4       data.GetLength(0)];
5       for (int i = 0; i < transpose.GetLength(0); i++)
6       {
7           for (int j = 0; j < transpose.GetLength(1); j++)
8           {
9               transpose[i, j] = data[j, i];
10          }
11      }
12      return transpose;
13  }
14  //melakukan perkalian matriks
15  public static double[,] getPerkalianMatrix(double[,] data, double[,]
16  data1)
17  {
18      double[,] Hasil = new double[data.GetLength(0),
19      data1.GetLength(1)];
20      for (int i = 0; i < Hasil.GetLength(0); i++)
21      {
22          for (int j = 0; j < Hasil.GetLength(1); j++)
23          {
24              for (int k = 0; k < data.GetLength(1); k++)
25              {
26                 Hasil[i, j] += data[i, k] * data1[k, j];
27              }
28          }
29      }
30      return Hasil;
31  }
32  //mendapatkan nilai H
33  public static double[,] getH2(double[,] data, double[] bias, int
34  fungsi)
35  {
36      double[,] H = new double[data.GetLength(0), data.GetLength(1)];
37      for (int i = 0; i < H.GetLength(0); i++)
38      {
39          for (int j = 0; j < H.GetLength(1); j++)
40          {
```

```
41                  if(fungsi == 1)
42                  {
43                      H[i, j] = 1 / (1 + Math.Exp(-(data[i, j] + bias[j])));
44                  }
45              }
46          }
47          return H;
48      }
```

Penjelasan Kode Program 5.3 mengenai implementasi proses inisialisasi bobot dan bias adalah sebagai berikut:

1. Baris 1 s.d 13 adalah proses untuk *transpose* pada bobot.
2. Baris 15 s.d 31 adalah proses perkalian matriks untuk mendapatkan nilai $Hinit$.
3. Baris 33 s.d 48 adalah proses untuk mendapatkan nilai H.

5.1.4 Implementasi Proses Perhitungan *Inverse* Matriks Menggunakan Operasi Baris Elementer (OBE)

Pada proses ini, langkah pertama yang dilakukan adalah membuat matriks identitas dan menggabungkannya dengan matriks yang ingin dilakukan proses *inverse*, lalu hitung *inverse* matriks tersebut menggunakan operasi baris elementer. Proses perhitungan *inverse* matriks menggunakan OBE ditunjukkan pada Kode Program 5.4.

Kode Program 5.4 Implementasi Proses Perhitungan Inverse Matriks Menggunakan Operasi Baris Elementer (OBE)

```
Algoritme 4: Inverse Matriks Menggunakan Operasi Baris Elementer (OBE)
1   public static double[,] inversOBE(double[,] matrix)
2   {
3       int length = matrix.GetLength(0);
4       int doubleLength = length * 2;
5       double[,]    inverse    =    new    double[matrix.GetLength(0),
6       matrix.GetLength(1)];
7       double[,] matriks = new double[length, doubleLength];
8       for (int i = 0; i < length; i++)
9       {
10          for (int j = length; j < doubleLength; j++)
11          {
12              if (i + length == j)
13              {
14                  matriks[i, j] = 1.0;
15              }
16              else
17              {
18                  matriks[i, j] = 0.0;
19              }
20              matriks[i, j - length] = matrix[i, j - length];
21          }
22      }
23
24      for (int j = 0; j < length - 1; j++)
25      {
26          for (int i = j + 1; i < length; i++)
27          {
28              if (matriks[i, j] != 0)
29              {
```

```
30                      double value = matriks[j, j] / matriks[i, j];
31                      for (int k = j; k < doubleLength; k++)
32                      {
33                          matriks[i, k] *= value;
34                          matriks[i, k] -= matriks[j, k];
35                      }
36                  }
37              }
38          }
39
40          for (int j = length - 1; j > 0; j--)
41          {
42              for (int i = j - 1; i >= 0; i--)
43              {
44                  if (matriks[i, j] != 0)
45                  {
46                      double value = matriks[j, j] / matriks[i, j];
47                      for (int k = i; k < doubleLength; k++)
48                      {
49                          matriks[i, k] *= value;
50                          matriks[i, k] -= matriks[j, k];
51                      }
52                  }
53              }
54          }
55
56          for (int j = 0; j < length; j++)
57          {
58              if (matriks[j, j] != 1)
59              {
60                  double value = 1 / matriks[j, j];
61                  for (int k = j; k < doubleLength; k++)
62                  {
63                      matriks[j, k] *= value;
64                  }
65              }
66          }
67
68          for (int i = 0; i < length; i++)
69          {
70              for (int j = length; j < doubleLength; j++)
71              {
72                  inverse[i, j - length] = matriks[i, j];
73              }
74          }
75          return inverse;
76      }
```

(Sumber: Mosabeth, Furqon dan Wihandika, 2018)

Penjelasan Kode Program 5.4 mengenai implementasi proses perhitungan *inverse* matriks menggunakan operasi baris elementer (OBE) adalah sebagai berikut:

1. Baris 3 adalah proses inisialisasi variabel panjang yang berasal dari panjang variabel matrix.
2. Baris 4 adalah proses inisialisasi variabel doubleLength.
3. Baris 5 s.d 6 adalah proses inisialisasi matriks *inverse* untuk menampung hasil *inverse* matriks.
4. Baris 7 adalah proses inialisasi variabel matriks.

5. Baris 8 s.d 22 adalah proses pembuatan identitas dan penggabungan matriks.
6. Baris 24 s.d 39 adalah proses perhitungan OBE pertama untuk menghitung *inverse* matriks.
7. Baris 40 s.d 55 adalah proses perhitungan OBE kedua untuk menghitung *inverse* matriks.
8. Baris 56 s.d 66 adalah proses perhitungan OBE ketiga untuk menghitung *inverse* matriks.
9. Baris 68 s.d 75 adalah proses memasukkan hasil *inverse* matriks ke matriks *inverse*.

5.1.5 Implementasi Proses Denormalisasi Data

Proses denormalisasi dilakukan untuk mengetahui nilai prediksi serta mengetahui perbandingan nilai aktual dengan nilai prediksi. Implementasi proses denormalisasi data ditunjukkan pada Kode Program 5.5.

Kode Program 5.5 Implementasi Proses Denormalisasi Data

Algoritme 5: Denormalisasi Data

```
1  public static long[] denormalization(double[,] Ydata, long min, long
2  max)
3  {
4        long[] denorm = new long[Ydata.GetLength(0)];
5        for (int i = 0; i < Ydata.GetLength(0); i++)
6        {
7            denorm[i] = (long)((Ydata[i, 0] * (max - min)) + min);
8        }
9        return denorm;
10 }
```

Penjelasan Kode Program 5.5 mengenai implementasi proses denormalisasi data adalah sebagai berikut:

1. Baris 4 adalah proses inisialisasi variabel denorm.
2. Baris 5 s.d 10 adalah proses denormalisasi data.

5.1.6 Implementasi Proses Perhitungan *Mean Average Percentage Error* (MAPE)

Pada proses ini, dilakukan perhitungan untuk mendapatkan nilai kesalahan *(error)* yang dilakukan pada proses *training* maupun *testing*. Implementasi proses perhitungan MAPE ditunjukkan pada Kode Program 5.6.

Kode Program 5.6 Implementasi Proses Perhitungan *Mean Average Percentage Error* (MAPE)

Algoritme 6: *Mean Average Percentage Error* (MAPE)

```
1  public static double getMape(long[] denormalisasi, long[] target)
2          {
3              double mape = 0.0;
4              double result = 0.0;
5              double[] data = new double[denormalisasi.GetLength(0)];
6              for (int i = 0; i < denormalisasi.GetLength(0); i++)
```

```
7               {
8                  data[i]    =    Math.Abs(((double)(denormalisasi[i]    -
9   target[i]) / target[i]) * 100);
10                 mape += data[i];
11              }
12              result = mape / denormalisasi.GetLength(0);
13              return result;
14          }
```

Penjelasan Kode Program 5.6 mengenai implementasi proses perhitungan *mean average percentage error* (MAPE) adalah sebagai berikut:

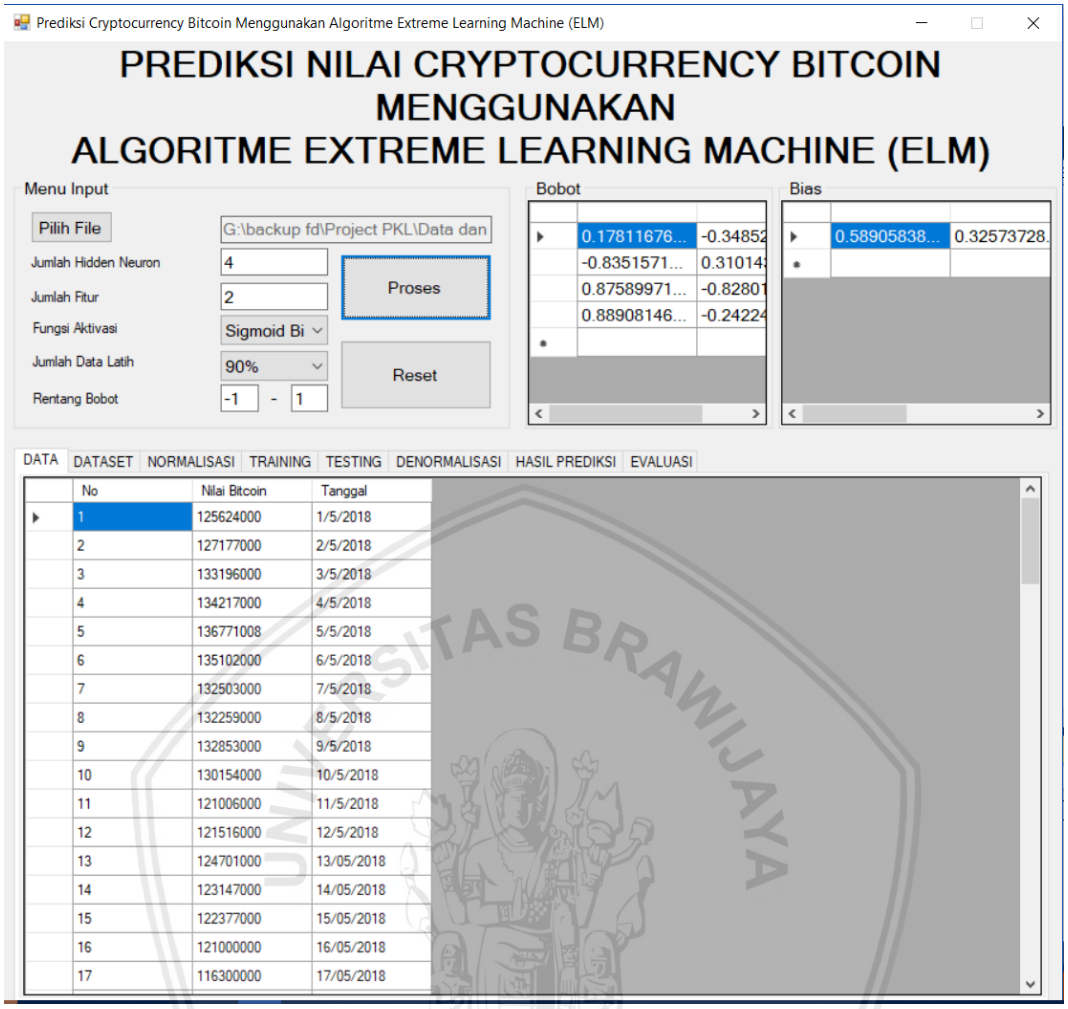
1. Baris 4 adalah proses inisialisasi variabel mape.
2. Baris 5 adalah proses inisialisasi variabel result.
3. Baris 6 s.d 7 adalah proses inisialisasi variabel data.
4. Baris 8 s.d 16 adalah proses perhitungan *mean absolute percentage error* (MAPE).

5.2 Implementasi Antarmuka

Hasil perancangan antarmuka yang telah dibuat sebelumnya akan dijelaskan pada subbab ini. Antarmuka sistem ini terdiri dari 8 halaman yang terdiri dari halaman utama, sub halaman *dataset*, sub halaman normalisasi, sub halaman *training*, sub halaman *testing,* sub halaman denormalisasi, sub halaman hasil prediksi, dan sub halaman evaluasi.

5.2.1 Implementasi Halaman Utama

Implementasi halaman utama merupakan halaman utama pada sistem prediksi nilai *cryptocurrency Bitcoin*. Tujuan dari halaman ini adalah untuk menampilkan keseluruhan komponen penyusun yang digunakan untuk melakukan proses prediksi. Pada halaman ini terdapat beberapa tabel-tabel serta kolom masukan yang digunakan untuk menampilkan data serta memasukkan nilai ke dalam sistem. Implementasi halaman utama ditunjukkan pada Gambar 5.1.

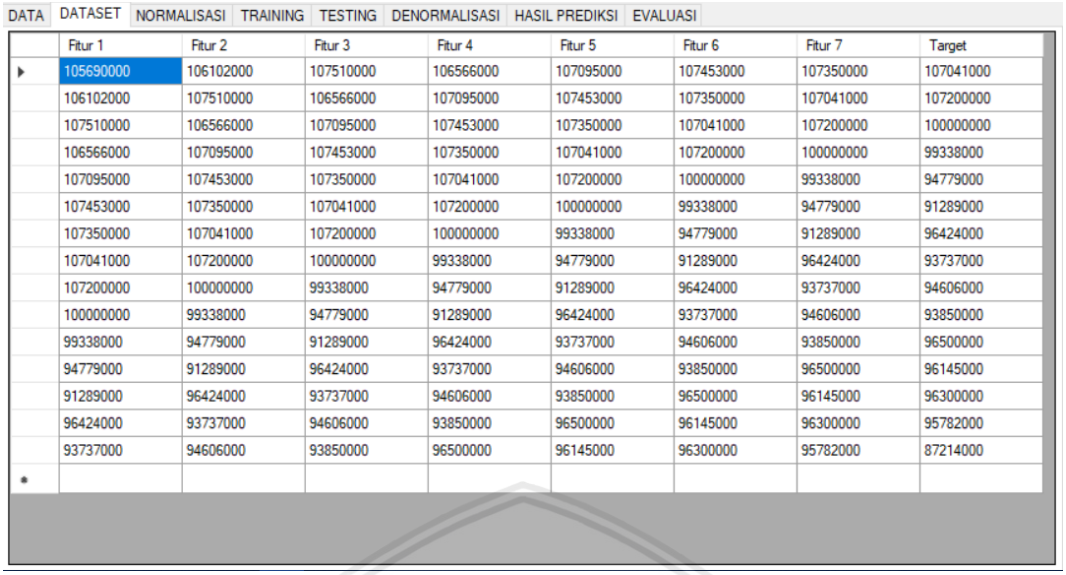


Gambar 5.1 Implementasi Halaman Utama

Pada Gambar 5.1, dijelaskan mengenai hasil implementasi halaman utama. Pada halaman ini terdapat Menu Input yang bertujuan untuk memasukkan data yang akan dilakukan proses prediksi dengan format .csv, memasukkan jumlah *hidden neuron*, jumlah fitur, memilih jenis fungsi aktivasi, Jumlah Data Latih, serta memasukkan rentang bobot yang diinginkan. Lalu, terdapat tombol proses dan reset yang akan melakukan aksi apabila ditekan. Disamping Menu input tedapat tabel yang akan menampilkan nilai bias dan bobot dari hasil aksi tombol proses.

5.2.2 Implementasi Sub Halaman *Dataset*

Pada implementasi sub halaman *dataset*, data yang akan dilakukan proses prediksi akan ditampilkan pada *datagridview*. Implementasi sub halaman *dataset* ditunjukkan pada Gambar 5.2.

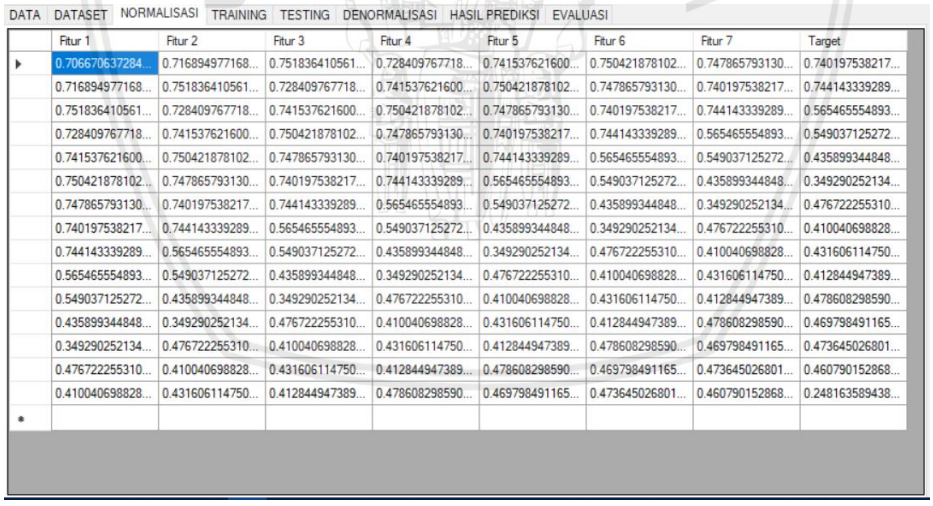


Fitur 1	Fitur 2	Fitur 3	Fitur 4	Fitur 5	Fitur 6	Fitur 7	Target
105690000	106102000	107510000	106566000	107095000	107453000	107350000	107041000
106102000	107510000	106566000	107095000	107453000	107350000	107041000	107200000
107510000	106566000	107095000	107453000	107350000	107041000	107200000	100000000
106566000	107095000	107453000	107350000	107041000	107200000	100000000	99338000
107095000	107453000	107350000	107041000	107200000	100000000	99338000	94779000
107453000	107350000	107041000	107200000	100000000	99338000	94779000	91289000
107350000	107041000	107200000	100000000	99338000	94779000	91289000	96424000
107041000	107200000	100000000	99338000	94779000	91289000	96424000	93737000
107200000	100000000	99338000	94779000	91289000	96424000	93737000	94606000
100000000	99338000	94779000	91289000	96424000	93737000	94606000	93850000
99338000	94779000	91289000	96424000	93737000	94606000	93850000	96500000
94779000	91289000	96424000	93737000	94606000	93850000	96500000	96145000
91289000	96424000	93737000	94606000	93850000	96500000	96145000	96300000
96424000	93737000	94606000	93850000	96500000	96145000	96300000	95782000
93737000	94606000	93850000	96500000	96145000	96300000	95782000	87214000

Gambar 5.2 Implementasi Sub Halaman *Dataset*

5.2.3 Implementasi Sub Halaman Normalisasi

Untuk implementasi sub halaman normalisasi, data yang sudah dilakukan normalisasi menggunakan *Min-Max Normalization* akan ditampilkan pada *datagridview*. Implementasi sub halaman normalisasi ditunjukkan pada Gambar 5.3.



Fitur 1	Fitur 2	Fitur 3	Fitur 4	Fitur 5	Fitur 6	Fitur 7	Target
0.706670637284...	0.716894977168...	0.751836410561...	0.728409767718...	0.741537621600...	0.750421878102...	0.747865793130...	0.740197538217...
0.716894977168...	0.751836410561...	0.728409767718...	0.741537621600...	0.750421878102...	0.747865793130...	0.740197538217...	0.744143339289...
0.751836410561...	0.728409767718...	0.741537621600...	0.750421878102...	0.747865793130...	0.740197538217...	0.744143339289...	0.565465554893...
0.728409767718...	0.741537621600...	0.750421878102...	0.747865793130...	0.740197538217...	0.744143339289...	0.565465554893...	0.549037125272...
0.741537621600...	0.750421878102...	0.747865793130...	0.740197538217...	0.744143339289...	0.565465554893...	0.549037125272...	0.435899344848...
0.750421878102...	0.747865793130...	0.740197538217...	0.744143339289...	0.565465554893...	0.549037125272...	0.435899344848...	0.349290252134...
0.747865793130...	0.740197538217...	0.744143339289...	0.565465554893...	0.549037125272...	0.435899344848...	0.349290252134...	0.476722255310...
0.740197538217...	0.744143339289...	0.565465554893...	0.549037125272...	0.435899344848...	0.349290252134...	0.476722255310...	0.410040698828...
0.744143339289...	0.565465554893...	0.549037125272...	0.435899344848...	0.349290252134...	0.476722255310...	0.410040698828...	0.431606114750...
0.565465554893...	0.549037125272...	0.435899344848...	0.349290252134...	0.476722255310...	0.410040698828...	0.431606114750...	0.412844947389...
0.549037125272...	0.435899344848...	0.349290252134...	0.476722255310...	0.410040698828...	0.431606114750...	0.412844947389...	0.478608298590...
0.435899344848...	0.349290252134...	0.476722255310...	0.410040698828...	0.431606114750...	0.412844947389...	0.478608298590...	0.469798491165...
0.349290252134...	0.476722255310...	0.410040698828...	0.431606114750...	0.412844947389...	0.478608298590...	0.469798491165...	0.473645026801...
0.476722255310...	0.410040698828...	0.431606114750...	0.412844947389...	0.478608298590...	0.469798491165...	0.473645026801...	0.460790152868...
0.410040698828...	0.431606114750...	0.412844947389...	0.478608298590...	0.469798491165...	0.473645026801...	0.460790152868...	0.248163589438...

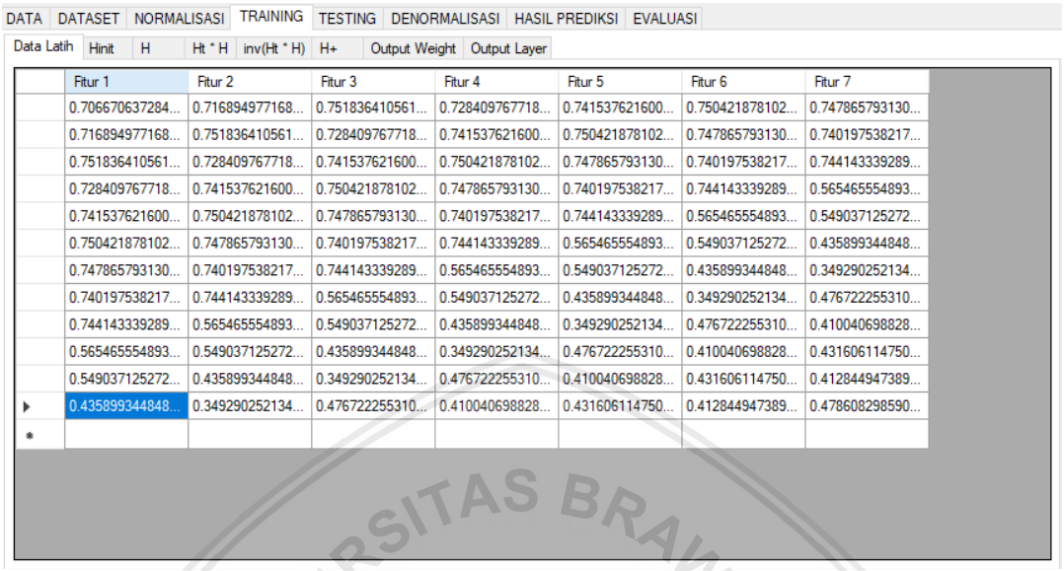
Gambar 5.3 Implementasi Halaman Normalisasi

Gambar 5.3 adalah hasil implementasi halaman normalisasi yang ditampilkan setelah pengguna menekan tombol proses. Setelah menekan tombol proses, maka data hasil normalisasi akan ditampilkan.

5.2.4 Implementasi Sub Halaman *Training*

Implementasi sub halaman *training* berfungsi untuk menampilkan hasil pembagian data latih, lalu hasil perhitungan H_{init}, hasil perhitungan *H*, hasil perhitungan perkalian *Ht* dengan *H*, hasil perhitungan *inverse* perkalian perkalian

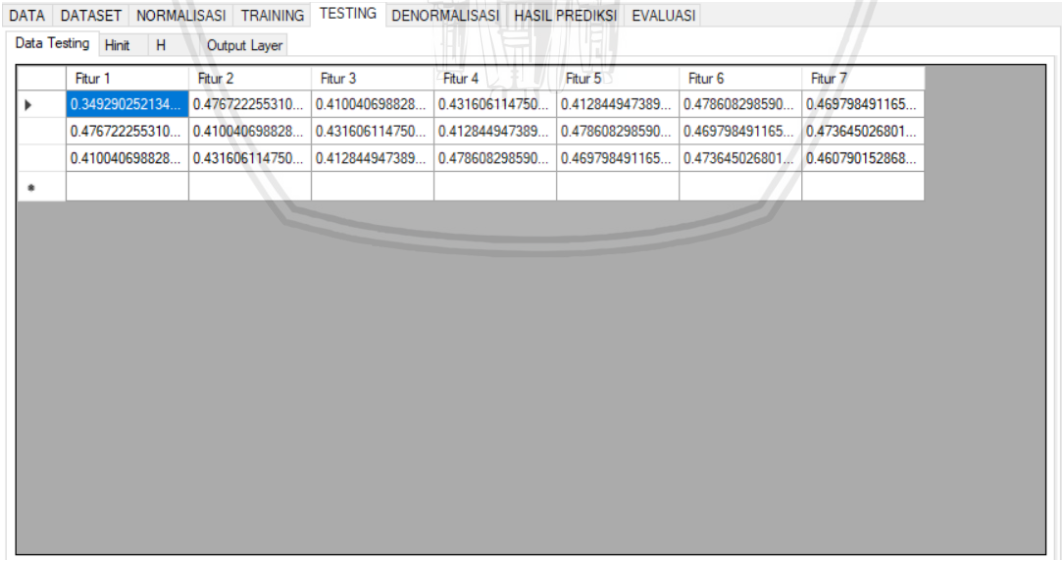
Ht dengan *H*, hasil perhitungan H^+, hasil perhitung dan untuk mendapatkan *output weight*, serta hasil perhitungan untuk mendapatkan nilai *output layer*. Implementasi sub halaman *training* ditunjukkan pada Gambar 5.4.



Gambar 5.4 Implementasi Sub Halaman *Training*

5.2.5 Implementasi Sub Halaman *Testing*

Implementasi sub halaman *testing* berfungsi untuk menampilkan hasil pembagian data uji, lalu hasil perhitungan H_{init}, hasil perhitungan *H*, serta hasil perhitungan untuk mendapatkan nilai *output layer*. Implementasi sub halaman *testing* ditunjukkan pada Gambar 5.5.

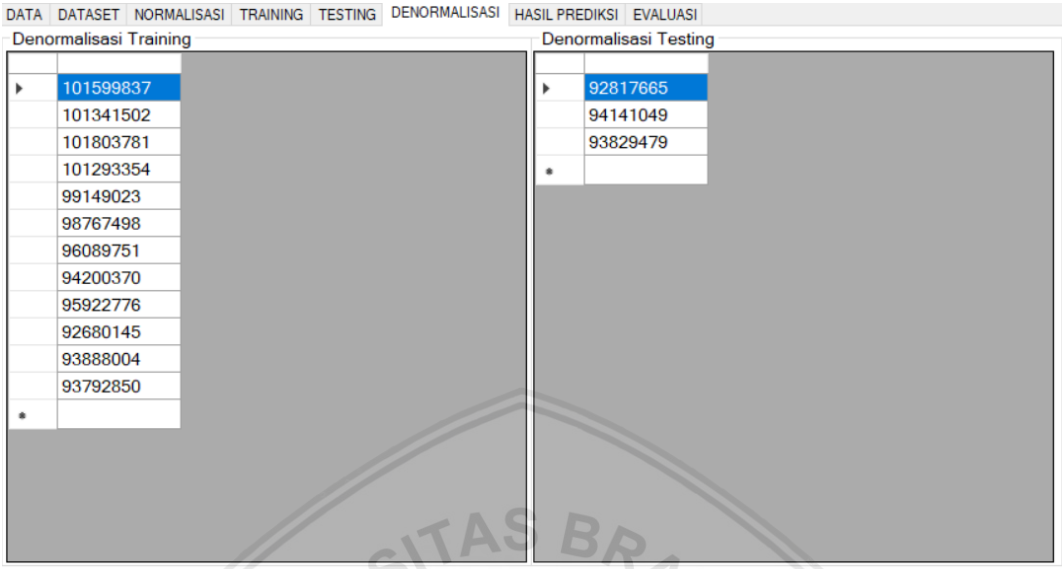


Gambar 5.5 Implementasi Sub Halaman *Testing*

5.2.6 Implementasi Sub Halaman Denormalisasi

Implementasi sub halaman denormalisasi berfungsi untuk menampilkan data hasil perhitungan denormalisasi pada *datagridview*. Baik hasil denormalisasi pada

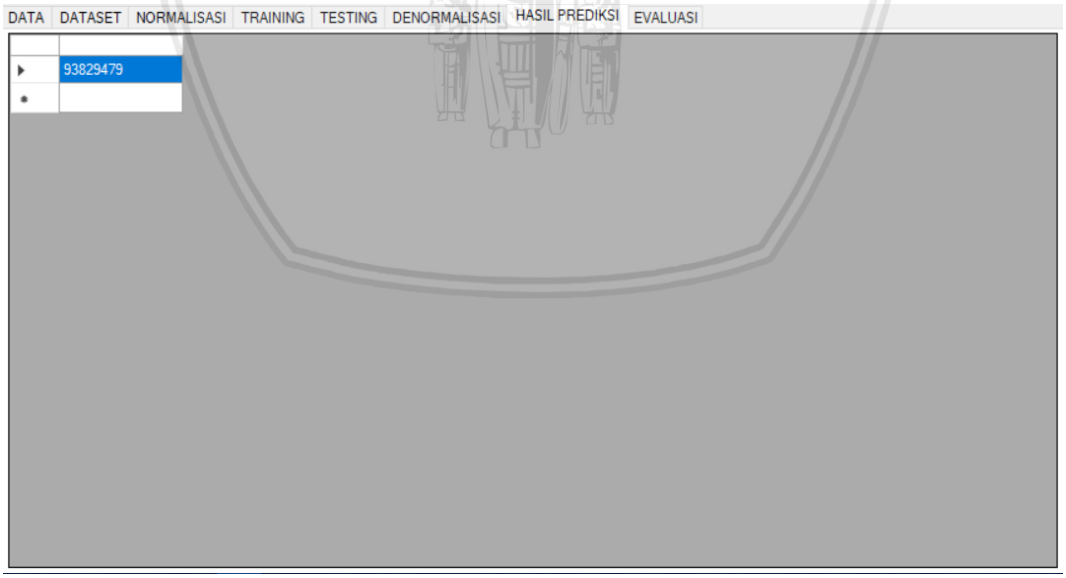
proses training dan testing akan ditampilkan pada sub halaman ini. Untuk lebih jelasnya akan ditunjukkan pada Gambar 5.6.



Gambar 5.6 Implementasi Sub Halaman Denormalisasi

5.2.7 Implementasi Halaman Hasil Prediksi

Implementasi sub halaman hasil prediksi berfungsi untuk melakukan prediksi terhadap data baru menggunakan algoritme *Extreme Learning Machine* (ELM). Hasil dari prediksi tersebut akan ditampilkan pada *datagridview*. Untuk lebih jelasnya akan ditunjukkan pada Gambar 5.7.

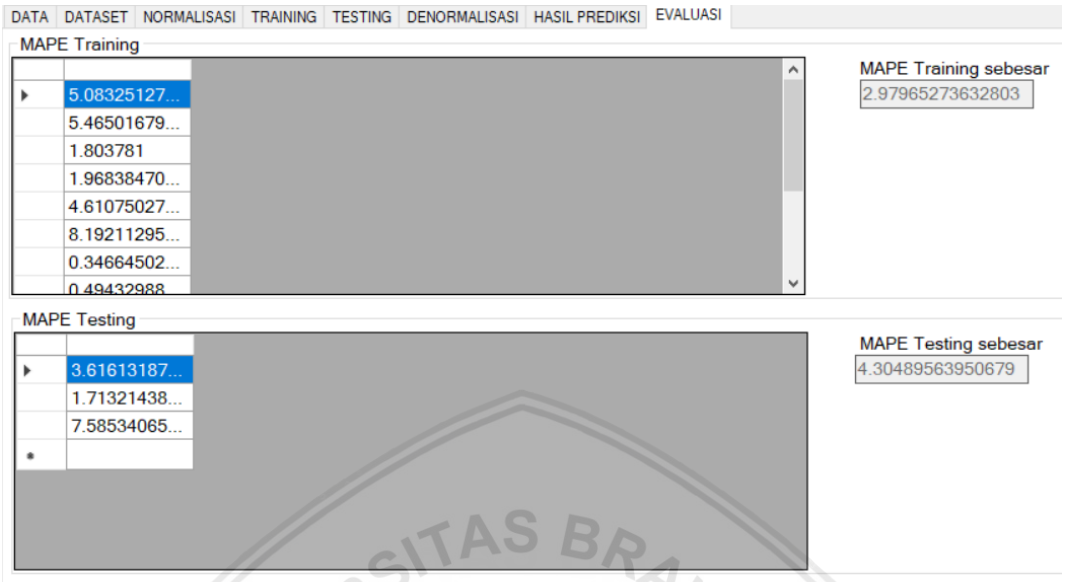


Gambar 5.7 Implementasi Sub Halaman Hasil Prediksi

5.2.8 Implementasi Sub Halaman Evaluasi

Pada sub halaman evaluasi, hasil dari perhitungan MAPE per data baik pada proses *training* maupun proses *testing* akan ditampikan pada *datagridview*. Lalu

nilai MAPE pada tiap proses akan ditampilkan pada kolom nilai MAPE. Untuk lebih jelasnya akan ditunjukkan pada Gambar 5.8.



Gambar 5.8 Implementasi Sub Halaman Evaluasi

BAB 6 PENGUJIAN DAN ANALISIS

6.1Pengujian Pengaruh Jumlah Fitur

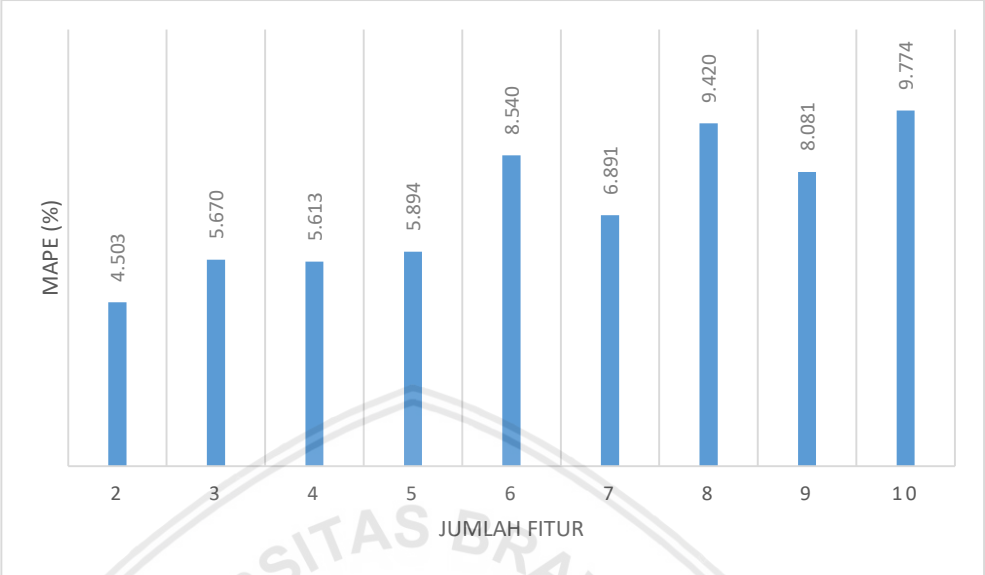
Pengujian pengaruh jumlah fitur dilakukan untuk mengetahui pengaruh jumlah fitur terhadap nilai kesalahan (*error*) menggunakan MAPE. Untuk pengujiannya sendiri, jumlah fitur yang digunakan mulai dari 2 sampai dengan 10 fitur. Pengujian dilakukan menggunakan fungsi aktivasi *sigmoid biner*, lalu jumlah *hidden neuron* 2, jumlah data latih 80% dari keseluruhan data dan sisanya untuk data uji, serta rentang bobot [-1,1]. Lalu, tiap fitur dilakukan percobaan sebanyak 10 kali yang mana akan dicari rata-rata nilai kesalahannya. Untuk lebih jelasnya akan ditampilkan pada Tabel 6.1.

Tabel 6.1 Pengujian Pengaruh Jumlah Fitur

Jumlah Fitur	Nilai MAPE							Rerata MAPE
	Percoban ke-							
	1	2	3	4	5	.....	10	
2	4.714	4.827	7.411	2.945	2.852	...	3.069	4.503
3	2.945	3.268	5.973	3.259	3.125	...	2.873	5.670
4	8.600	7.357	4.067	6.290	4.944	...	3.091	5.613
5	4.812	4.357	6.386	5.492	7.274	...	5.503	5.894
6	12.646	8.268	5.988	11.306	7.580	...	5.141	8.540
7	3.575	6.020	14.126	4.831	4.585	...	6.112	6.891
8	8.128	12.130	9.254	6.195	10.293	...	6.566	9.420
9	4.781	8.787	4.567	18.203	4.703	...	7.968	8.081
10	6.776	10.761	9.346	19.070	14.359	...	7.372	9.774

Berdasarkan pengujian yang telah dilakukan terhadap masing-masing jumlah fitur yang ditunjukkan pada Tabel 6.1, menunjukkan rerata MAPE pada jumlah fitur 2 sebesar 4.503, lalu pada jumlah fitur 3 sebesar 5.670, jumlah fitur 4 sebesar 5.613, jumlah fitur 5 sebesar 5.894, jumlah fitur 6 sebesar 8.540, jumlah fitur 7 sebesar 6.891, jumlah fitur 8 sebesar 9.420, jumlah fitur 9 sebesar 8.081, dan jumlah fitur 10 sebesar 9.744. Dari hasil tersebut, nilai kesalahan terkecil pada pengujian jumlah fitur dimiliki oleh pengujian pada jumlah fitur 2. Hal ini disebabkan karena jumlah data yang dimiliki saat jumlah fitur 2 lebih banyak daripada yang lain. Hal ini tentu saja berpengaruh terhadap proses pelatihan karena semakin banyak data latih dapat memperkecil nilai kesalahan. Selain itu, fitur yang digunakan adalah jumlah hari sebelum target yang mengindikasikan bahwa untuk memprediksi nilai *Bitcoin* dibutuhkan membutuhkan rentang hari yang tidak boleh terlalu jauh untuk menghasilkan

prediksi yang baik. Untuk visualisasi perbandingan jumlah fitur ditunjukkan pada Gambar 6.1.



Gambar 6.1 Grafik Perbandingan Jumlah Fitur

6.2 Pengujian Pengaruh Jumlah *Hidden Neuron*

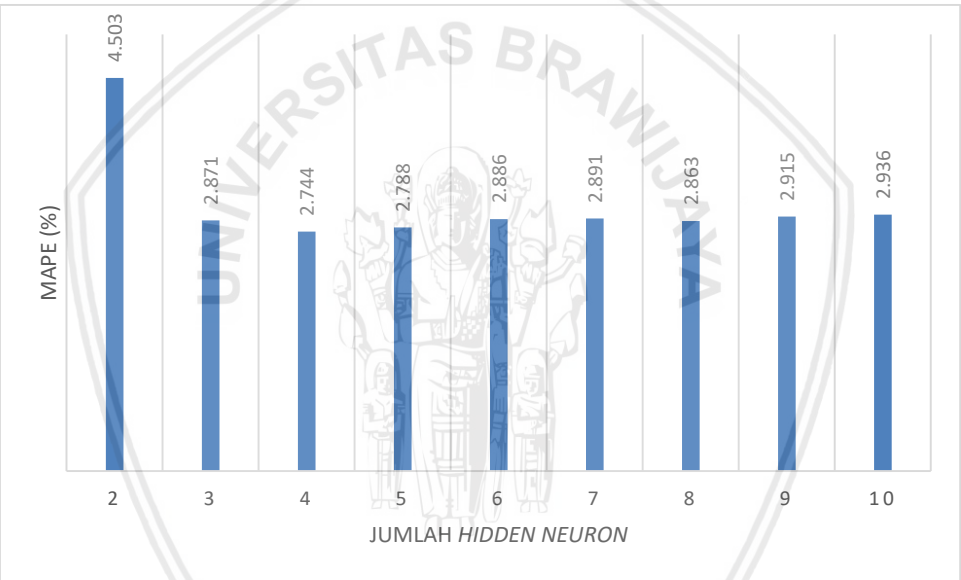
Pengujian pengaruh jumlah hidden neuron dilakukan untuk mencari berapa jumlah *hidden neuron* terbaik yang dapat memberikan nilai kesalahan terkecil pada prediksi nilai *cryptocurrency Bitcoin* menggunakan algoritme ELM ini. Untuk melakukannya, kita harus menentukan berapa saja jumlah *hidden neuron* yang akan diuji. Untuk pengujian ini, jumlah *hidden neuron* yang akan diuji berjumlah mulai dari 2 sampai dengan 10. Serta pengujian ini dilakukan menggunakan fungsi aktivasi *sigmoid biner*, lalu jumlah fitur 2, jumlah data latih 80% dari keseluruhan data dan sisanya untuk data uji, serta rentang bobot [-1,1]. Untuk pengujiannya sendiri, per jumlah hidden neuron akan diuji sebanyak 10 kali. Lalu akan dicari rerata nilai kesalahannya per jumlah hidden neuron. Untuk lebih jelasnya akan ditampilkan pada Tabel 6.2.

Tabel 6.2 Pengujian Pengaruh Jumlah *Hidden Neuron*

Jumlah *Hidden Neuron*	Nilai MAPE							Rerata MAPE
	Percoban ke-							
	1	2	3	4	5	…	10	
2	4.714	4.827	7.411	2.945	2.852	…	3.069	4.503
3	2.539	2.575	2.519	2.651	2.615	…	5.124	2.871
4	2.748	2.724	2.620	2.750	2.792	…	2.781	2.744
5	2.782	2.779	2.715	2.905	2.741	…	2.780	2.788

6	2.874	2.892	2.971	2.835	2.835	...	2.924	2.886
7	2.826	2.853	2.960	2.839	2.982	...	2.951	2.891
8	2.896	2.927	2.972	2.870	2.698	...	2.945	2.863
9	2.943	2.955	2.792	2.941	2.953	...	2.900	2.915
10	3.252	2.909	2.982	2.920	2.947	...	2.746	2.936

Berdasarkan hasil pengujian yang ditunjukkan pada Tabel 6.2, Rerata MAPE terkecil untuk pengujian jumlah *hidden neuron* terdapat pada jumlah hidden neuron sebanyak 4 dengan nilai rerata MAPE sebesar 2.744. Hal ini menunjukkan bahwa tidak selalu jumlah *hidden neuron* yang banyak dapat memberikan hasil prediksi yang baik. Untuk visualisasi perbandingan jumlah *hidden neuron* ditunjukkan pada Gambar 6.2.



Gambar 6.2 Grafik Perbandingan Jumlah *Hidden Neuron*

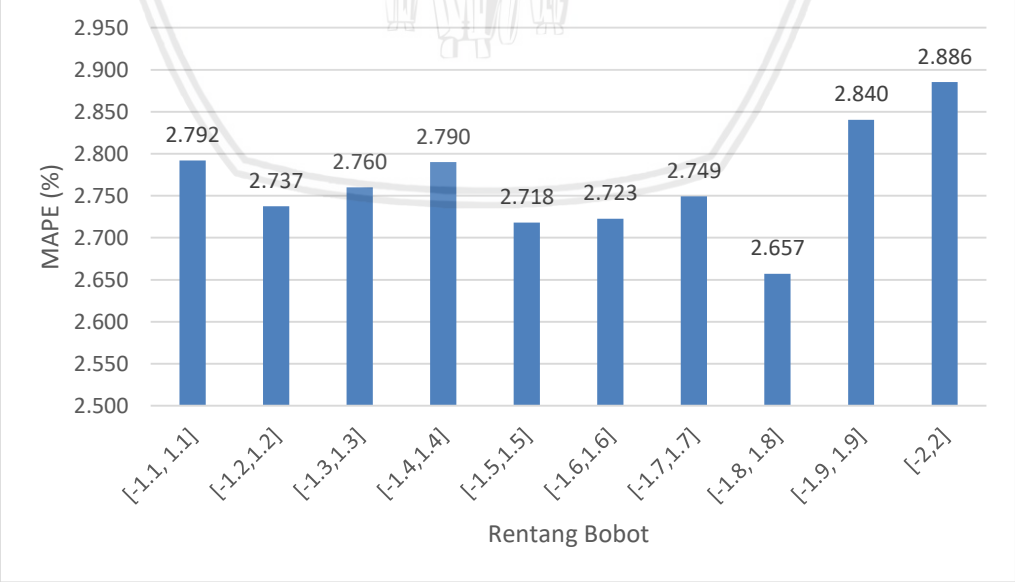
6.3 Pengujian Pengaruh Rentang Bobot Masukan

Pengujian pengaruh rentang bobot masukan untuk mencari rentang bobot terbaik pada fungsi aktivasi sigmoid biner yang dapat memberikan nilai kesalahan terkecil prediksi nilai *cryptocurrency Bitcoin* menggunakan algoritme ELM ini. Untuk melakukannya, kita harus menentukan jumlah serta rentang bobot yang akan diuji. Pada pengujian ini, jumlah rentang bobot adalah 10 dengan rentang antara -2 sampai dengan 2. Pengujian ini dilakukan menggunakan fungsi aktivasi *sigmoid biner*, dengan jumlah fitur sebanyak 2, jumlah *hidden neuron* sebanyak 4. Setiap rentang bobot akan diuji sebanyak 10 kali yang kemudian akan dicari rerata nilai kesalahannya. Untuk lebih jelasnya akan ditampilkan pada Tabel 6.3

Tabel 6.3 Pengujian Pengaruh Rentang Bobot Masukan

Rentang Bobot Masukan	Nilai MAPE							Rerata MAPE
	Percoban ke-							
	1	2	3	4	5	...	10	
[-1, 1]	2.748	2.724	2.620	2.750	2.792	...	2.781	2.744
[-1.1, 1.1]	3.517	2.710	2.630	2.744	2.635	...	2.763	2.792
[-1.2, 1.2]	2.849	2.827	2.691	2.663	2.683	...	2.742	2.737
[-1.3, 1.3]	2.839	2.577	2.850	2.776	2.705	...	2.772	2.760
[-1.4, 1.4]	2.801	2.764	2.742	2.788	2.826	...	2.955	2.790
[-1.5, 1.5]	2.731	2.601	2.770	2.671	2.893	...	2.535	2.718
[-1.6, 1.6]	2.805	2.673	2.598	2.753	2.810	...	2.532	2.723
[-1.7, 1.7]	2.784	2.666	2.724	2.807	2.774	...	2.710	2.749
[-1.8, 1.8]	2.584	2.707	2.616	2.644	2.659	...	2.662	2.657
[-1.9, 1.9]	2.812	2.794	2.720	2.626	2.663	...	2.742	2.840
[-2, 2]	4.515	2.740	2.840	2.753	2.651	...	2.692	2.886

Berdasarkan hasil pengujian yang ditunjukkan pada Tabel 6.3, Rerata MAPE terkecil untuk pengujian ini terdapat pada rentang bobot [-1.8, 1.8] dengan nilai 2.657. Hal ini menunjukkan bahwa rentang bobot berpengaruh terhadap nilai kesalahan yang dihasilkan. Untuk visualisasi perbandingan rentang bobot ditunjukkan pada Gambar 6.3.



Gambar 6.3 Grafik Perbandingan Rentang Bobot

6.4 Pengujian Pengaruh Jumlah Data Latih dengan Data Konstan

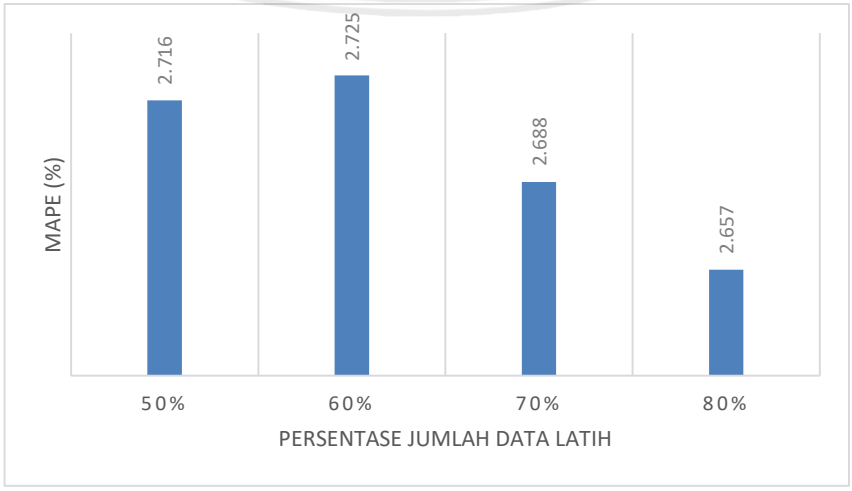
Pengujian jumlah data latih dengan data konstan dilakukan untuk mencari berapa persentase data latih terbaik yang dapat menghasilkan nilai MAPE terkecil untuk memprediksi nilai *cryptocurrency Bitcoin* menggunakan algoritme ELM. Pengujian ini dilakukan dengan melakukan 10 kali percobaan untuk setiap nilai persentase yaitu 50%. 60%. 70%, 80%, dan 90%. Sementara persentase jumlah data uji dibuat konstan yaitu sebesar 20%. Lalu untuk setiap nilai persentase akan didapatkan nilai rerata MAPE. Untuk lebih jelasnya akan ditampilkan pada Tabel 6.4.

Tabel 6.4 Pengujian Jumlah Data Latih dengan Data Konstan

Persentase Jumlah Data Latih	Nilai MAPE							Rerata MAPE
	Percoban ke-							
	1	2	3	4	5	...	10	
50%	2.590	2.671	2.887	2.637	2.806	...	3.003	2.716
60%	2.699	2.694	2.445	2.796	2.641	...	2.785	2.725
70%	2.856	2.720	2.677	2.685	2.738	...	2.612	2.688
80%	2.584	2.707	2.616	2.644	2.659	...	2.662	2.657

Berdasarkan hasil pengujian yang ditunjukkan pada Tabel 6.4, persentase jumlah data latih yang menghasilkan rerata nilai MAPE terkecil didapat pada persentase jumlah data latih sebesar 90%. Hal ini menunjukkan bahwa semakin besar jumlah data latih yang digunakan dapat menghasilkan nilai MAPE yang semakin kecil yang menandakan bahwa hasil prediksi mendekati nilai asli yang mana jika nilai MAPE tersebut dikonversikan ke dalam rupiah, maka selisih yang didapatkan berkisar antara Rp. 2.000.000 – Rp. 3.000.000. Untuk visualisasi perbandingan nilai MAPE ditunjukkan pada Gambar 6.4.

Gambar 6.4 Grafik Perbandingan Nilai MAPE Jumlah Data Latih dengan Data Uji Konstan



BAB 7 PENUTUP

7.1 Kesimpulan

Berdasarkan hasil pengujian dan analisis terhadap sistem prediksi *cryptocurrency Bitcoin* menggunakan algoritme *Extreme Learning Machine* (ELM), maka dapat ditarik kesimpulan sebagai berikut:

1. Algoritme *Extreme Learning Machine* (ELM) dapat digunakan untuk memprediksi nilai *cryptocurrency Bitcoin* dengan jumlah *hidden neuron* terbaik sebanyak 4.
2. Jumlah fitur terbaik untuk memprediksi nilai *cryptocurrency* bitcin menggunakan algoritme *Extreme Learning Machine* (ELM) sebanyak 2 fitur.
3. Rentang bobot fungsi aktivasi sigmoid biner terbaik untuk memprediksi nilai *cryptocurrency Bitcoin* menggunakan algoritme *Extreme Learning Machine* (ELM) adalah [-1.8, 1.8] dengan persentase data latih 80% dan nilai rerata MAPE terkecil sebesar **2.657%**.

7.2 Saran

Penelitian selanjutnya mengenai penggunaan algoritme *Extreme Learning Machine* (ELM) untuk prediksi khususnya regresi dapat dikembangkan dengan beberapa saran sebagai berikut:

1. Untuk menghasilkan nilai MAPE yang lebih kecil, dapat dilakukan optimasi bobot menggunakan algorite evolusi atau *Particle Swarm Optimization* (PSO).
2. Untuk penelitian selanjutnya, diharapkan peneliti dapat menampilkan prediksi untuk beberapa hari ke depan, tidak hanya satu hari saja.

DAFTAR REFERENSI

Admin, 2017. *Faktor-faktor Yang Mempengaruhi Harga Bitcoin*. [online] Available at: <https://forexindonesia.org/*Bitcoin*/faktor-faktor-yang-mempengaruhi-harga-*Bitcoin*.html> [Accessed 13 Mar. 2019].

Barata, J.C.A. and Hussein, M.S., 2012. The Moore-Penrose Pseudoinverse: A Tutorial Review of the Theory. *Brazilian Journal of Physics*, 42(1–2), pp.146–165.

Dewi, S.N., Cholissodin, I. and Santoso, E., 2018. Prediksi Jumlah Kriminalitas Menggunakan Metode Extreme Learning Machine (Studi Kasus Di Kabupaten Probolinggo). 2(11), pp.4687–4693.

Dudek, G., 2016. Extreme learning machine as a function approximator: Initialization of input weights and biases. *Advances in Intelligent Systems and Computing*, 403, pp.59–69.

Fitriyani, I.R., Setiawan, B.D. and Perdana, R.S., 2018. Prediksi Indeks Harga Konsumen (IHK) Kelompok Perumahan , Air , Listrik , Gas , dan Bahan Bakar Menggunakan Metode Extreme Learning Machine. 2(11).

Huang, G. Bin, Zhu, Q.Y. and Siew, C.K., 2006. Extreme learning machine: Theory and applications. *Neurocomputing*, 70(1–3), pp.489–501.

Huixuan, F., Yuchao, W. and Hongmei, Z., 2015. Ship rolling motion prediction based on extreme learning machine. *2015 34th Chinese Control Conference (CCC)*, [online] pp.3468–3472. Available at: <http://ieeexplore.ieee.org/document/7260174/>.

Jauhari, D., Cholissodin, I. and Dewi, C., 2017. Prediksi Nilai Tukar Rupiah Indonesia Terhadap Dolar Amerika Serikat Menggunakan Metode Recurrent Extreme Learning Machine Neural Network. 1(11), pp.1188–1197.

Kim, S. and Kim, H., 2016. A new metric of absolute percentage error for intermittent demand forecasts. *International Journal of Forecasting*, [online] 32(3), pp.669–679. Available at: <http://dx.doi.org/10.1016/j.ijforecast.2015.12.003>.

Liang, N.-Y., Huang, G.-B., Saratchandran, P. and Sundararajan, N., 2006. A Fast and Accurate Online Sequential Learning Algorithm for Feedforward Networks. *IEEE Transactions on Neural Networks*, [online] 17(6), pp.1411–1423. Available at: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4012031%5Cnpapers3://publication/doi/10.1109/TNN.2006.880583>.

Luno, 2017. *Apa yang Menyebabkan Harga Bitcoin Naik Turun?*

Mosabeth, C., Furqon, M.T. and Wihandika, R.C., 2018. Prediksi Harga Pasar Daging Sapi Di Kota Malang Dengan Menggunakan Metode Extreme Learning Machine (ELM). 2(12), pp.6362–6369.

Permatasari, A.R. and Rahayudi, B., 2018. Estimasi Hasil Produksi Benih Tanaman Kenaf (Hibiscus Cannabinus L .) Menggunakan Metode Extreme Learning Machine (ELM) Pada Balai Penelitian Tanaman Pemanis dan Serat (Balittas). 2(11), pp.5475–5483.

Pratama, A.H., 2017. *[Update] Bitcoin 101 untuk Para Pemula*. [online] Available at: <https://id.techinasia.com/fakta-penting-tentang-*Bitcoin*> [Accessed 10 Aug. 2018].

Radityo, A., Munajat, Q. and Budi, I., 2017. Prediction of *Bitcoin* Ex change Rate to Americ an Dollar Using Artificial Neural Network Methods. *ICACSIS*.

Setiawan, S.R.D., 2017. *Nilai Terus Menguat, Pengguna Bitcoin Indonesia Meningkat*.

Sun, Z.L., Choi, T.M., Au, K.F. and Yu, Y., 2008. Sales forecasting using extreme learning machine with applications in fashion retailing. *Decision Support Systems*, [online] 46(1), pp.411–419. Available at: <http://dx.doi.org/10.1016/j.dss.2008.07.009>.

Velankar, S., Valecha, S., Maji, S. and *Bitcoin*, A., 2018. *Bitcoin* Price Prediction using Machine Learning. pp.144–147.

Yekkin, T., Aratari, D. and Pagliery, J., 2018. *What is Bitcoin?*