PREDIKSI HARGA CABAI RAWIT DI KOTA MALANG MENGGUNAKAN ALGORITME *EXTREME LEARNING MACHINE* (ELM)

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:
Galih Ariwanda
NIM: 155150200111029

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

PENGESAHAN

PREDIKSI HARGA CABAI RAWIT DI KOTA MALANG MENGGUNAKAN ALGORITME *EXTREME LEARNING MACHINE* (ELM)

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh :
Galih Ariwanda
NIM: 155150200111029

Skripsi ini telah diuji dan dinyatakan lulus pada
17 Juni 2019
Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I	Dosen Pembimbing II
Imam Cholissodin, S.Si, M.Kom	Tibyani, S.T, M.T
NIK: 201201 850719 1 001	NIP: 19691101 199512 1 002

Mengetahui
Ketua Jurusan Teknik Informatika

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

PERNYATAAN ORISINALITAS

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

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

Malang, 17 Juni 2019

Galih Ariwanda

NIM: 155150200111029

PRAKATA

Puji syukur penulis panjatkan atas ke hadirat Allah SWT yang telah memberikan rahmat-Nya sehingga penulis dapat menyelesaikan penelitian yang berjudul "Prediksi Harga Cabai Rawit Di Kota Malang Menggunakan Algoritme *Extreme Learning Machine* (ELM)". Dalam kesempatan ini penulis ingin menyampaikan rasa terima kasih kepada pihak-pihak yang telah membimbing dan membantu penulis dalam peyusunan laporan penelitian ini, antara lain:

1. Bapak Imam Cholissodin, S.Si, M.Kom, selaku Dosen Pembimbing I yang telah baik hati dan sabar membimbing penulis dalam menyelesaikan penelitian ini.
2. Bapak Tibyani, S.T, M.T, selaku Dosen Pembimbing II yang dengan ikhlas membimbing dan mengarahkan penulis dalam menyelesaikan penyusunan penelitian ini.
3. Bapak Agus Wahyu Widodo, S.T, M.Cs, selaku Ketua Program Studi Teknik Informatika, Fakultas Ilmu Komputer Universitas Brawijaya.
4. Bapak Tri Astoto Kurniawan, S.T, M.T, Ph.D, selaku Ketua Jurusan Teknik Informatika, Fakultas Ilmu Komputer Universitas Brawijaya.
5. Bapak Achmad Basuki, S.T, M.MG, Ph.D, selaku Dosen Pembimbing Akademik yang telah membimbing penulis selama menempuh perkuliahan.
6. Keluarga penulis, terutama kedua orang tua, adik, kakek, dan nenek yang selalu memberikan do'a, motivasi, dan kesabaran dalam segala hal bagi penulis dalam menyelesaikan penelitian ini.
7. Uke Rahma Hidayah, selaku orang terdekat yang paling saya percaya dan telah banyak memberikan do'a dan bantuan kepada penulis dalam menyelesaikan penelitian ini.
8. Angga Yanuar Pratama, Fadhlillah Ikhsan, Yoga Pratama, Rahmat Faizal, M. Rafi Farhan, dan M. Zuliono D.R.P, selaku teman penulis yang telah memberikan banyak bantuan dalam proses perkuliahan dan pengerjaan penelitian.
9. Seluruh pihak yang telah mendukung dan membantu penulis dalam menyeselesaikan penelitian ini yang belum dicantumkan namanya.

Penulis menyadari penelitian ini masih memiliki kekurangan, untuk itu penulis sangat mengharapkan kritik dan saran yang bersifat membangun. Semoga penelitian ini dapat digunakan dengan baik dan bermanfaat dalam membantu penelitian selanjutnya.

Malang, 17 Juni 2019

Penulis

galihariwanda01@gmail.com

ABSTRAK

Galih Ariwanda, Prediksi Harga Cabai Rawit Di Kota Malang Menggunakan Algoritme *Extreme Learning Machine* (ELM)

Pembimbing: Imam Cholissodin, S.Si, M.Kom dan Tibyani, S.T, M.T

Tanaman cabai rawit merupakan tanaman komoditas untuk bahan pangan yang tidak bisa terlepas dari kebutuhan sehari-hari masyarakat di Indonesia. Salah satu kota di Indonesia yang menjadikan cabai rawit sebagai bahan olahan makanan adalah Kota Malang. Cabai rawit bagi masyarakat di Kota Malang dikonsumsi untuk menjaga metabolisme dan suhu tubuh agar tetap hangat serta vitamin C yang dapat membantu dalam menjaga kesehatan tubuh manusia. Harga cabai rawit di Kota Malang selalu terjadi perubahan yang fluktuatif setiap harinya. Perubahan yang terjadi secara fluktuatif membuat harga cabai rawit sulit diprediksi dengan baik. Selain itu, harga yang diberikan pedagang selalu bervariasi, cabai rawit juga termasuk salah satu komoditas penyumbang inflasi dan mencegah terjadinya perbedaan harga yang didapatkan oleh konsumen dan petani agar tidak saling dirugikan. Maka dari itu, diperlukan prediksi harga cabai rawit di Kota Malang agar konsumen dan pemerintah dapat melakukan tindakan pencegahan terhadap masalah yang ada. Pada penelitian ini, proses prediksi dilakukan dalam beberapa proses yang terdiri dari *pre-processing*, normalisasi data, prediksi menggunakan algoritme *Extreme Learning Machine*, dan hasil *error* menggunakan MAPE. Berdasarkan hasil pengujian yang telah dilakukan menggunakan data harga cabai rawit dari tanggal 1 Januari 2017 sampai 31 Desember 2018 di Kota Malang diperoleh nilai MAPE terkecil sebesar 2,087% dengan banyak fitur sebanyak 2, banyak *neuron* pada *hidden layer* sebanyak 5, persentase data *training* dan data *testing* 90%:10%, dan fungsi aktivasi yang digunakan *Sigmoid Biner*.

Kata kunci: prediksi, harga cabai rawit, *extreme learning machine,* MAPE

ABSTRACT

Galih Ariwanda, ***Prediction of Prices of Cayenne in Malang City Using Extreme Learning Machine (ELM) Algorithm***

***Supervisors*: Imam Cholissodin, S.Si, M.Kom *and* Tibyani, S.T, M.T**

Cayenne is a commodity for food that cannot be separated from the daily needs of people in Indonesia. One of the cities in Indonesia that makes cayenne as a processed food ingredient is Malang City. Cayenne for people in Malang City is consumed to keep metabolism and body temperature warm and vitamin C which can help maintain the health of the human body. Prices of cayenne in Malang City always fluctuate changes every day. Fluctuation changes that make the price of cayenne are difficult to predict well. In addition, the prices given by traders are always varied, cayenne pepper is also one of the contributing commodities of inflation and prevents the difference in prices obtained by consumers and farmers so that they are not harmed by each other. Therefore, it is necessary to predict the price of cayenne in Malang so that consumers and the government can take preventive measures against the existing problems. In this study, the prediction process was carried out in several processes consisting of pre-processing, normalization of data, predictions using the Extreme Learning Machine algorithm, and the error results using MAPE. Based on the results of testing that has been carried out using cayenne price data from January 1, 2017 to December 31, 2018 in Malang City the smallest MAPE value was 2.087% with 2 features, 5 neurons in the hidden layer, percentage training data and testing data 90%: 10%, and the activation function is Binary Sigmoid.

Keywords: prediction, price of cayenne, extreme learning machine, MAPE

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR LAMPIRAN

BAB 1 PENDAHULUAN

1.1 Latar Belakang

Cabai rawit merupakan bagian dari tanaman komoditas kebutuhan pangan yang tidak bisa terlepas dari kehidupan sehari-hari masyarakat di Indonesia. Banyak masyarakat Indonesia yang mengonsumsi cabai rawit sebagai bahan olahan untuk kebutuhan setiap hari. Cabai rawit umumnya dikenal masyarakat luas karena memiliki ciri khas yang unik dibandingkan tanaman yang lain. Ciri khas tersebut adalah rasa dari tanaman yang sangat pedas saat dimakan serta kandungan vitamin C yang dapat membantu dalam menjaga kesehatan tubuh manusia. Vitamin C dalam cabai rawit memiliki kandungan yang disebut antioksidan yang mana berfungsi untuk meningkatkan daya tahan tubuh pada saat proses penyerapan kalsium di dalam tubuh manusia (Rosmainar et al., 2018). Banyak masyarakat yang menjadikan cabai rawit sebagai bahan utama pada masakan untuk menghasilkan rasa pedas. Salah satu kota di wilayah Indonesia yang mengkonsumsi cabai rawit adalah Kota Malang. Kota Malang sendiri terletak di wilayah Jawa Timur dengan luas kota tersebut sebesar 110,06 km^2. Luas wilayah kota Malang dibagi menjadi lima kecamatan antara lain Kecamatan Kedungkandang, Kecamatan Sukun, Kecamatan Klojen, Kecamatan Blimbing, dan Kecamatan Lowokwaru dengan ketinggian wilayah sebesar 445-526 m di atas permukaan air laut dan jumlah penduduk di proyeksi tahun 2017 sebanyak 861.414 jiwa (BPS, 2018). Dengan ketinggian wilayah yang cukup tinggi membuat suhu di Kota Malang mencapai rata-rata sebesar 15,8^oC-24,9^oC. Dengan suhu wilayah yang dingin, cabai rawit sangat cocok untuk dikonsumsi agar metabolisme dan suhu tubuh agar tetap hangat bagi masyarakat di Kota Malang.

Harga jual cabai rawit di Kota Malang selalu mengalami perubahan yang selalu fluktuatif setiap harinya. Perubahan harga yang fluktuatif membuat harga cabai rawit sulit diprediksi dengan baik. Dari data yang telah diambil pada tahun 2017-2018 dari *website* Dinas Perindustrian dan Perdagangan Jawa Timur (Disperindag Jatim), Kota Malang pernah mandapatkan harga cabai yang tidak wajar mencapai Rp136.000,00 per kilogram (Disperindag Jatim, 2018). Sehingga tercatat bahwa setiap hari harga untuk cabai rawit di Kota Malang selalu mengalami perubahan. Perubahan harga cabai rawit tertinggi terjadi pada tanggal 22-23 Desember 2018 sebesar Rp23.800,00 dan Rp35.800,00 yang mana dari harga tersebut terjadi kenaikan harga sebesar Rp12.000,00 (Disperindag Jatim, 2018). Selain itu, perbedaan harga yang sangat bervariasi dari setiap pasar dan pada pedagang dari pasar yang sama membuat konsumen merasa kebingungan dalam menentukan harga yang dapat dipercaya. Pada tanggal 23 Desember 2018 harga cabai rawit untuk pasar Blimbing sebesar Rp35.000,00 sedangkan di pasar Oro-Oro Dowo sebesar Rp37.000,00 (Disperindag Jatim, 2018). Karena perubahan yang sangat tinggi pada harga membuat cabai rawit termasuk ke dalam salah satu 10 komoditas penyumbang inflasi di Kota Malang. Hal tersebut membuat khawatir terhadap pihak pemerintah dalam menjaga

stabilisasi harga cabai agar tetap bisa terjangkau sehingga nilai inflasi di Kota Malang akan baik. Dari penelitian jangka panjang terhadap harga cabai rawit tahun 2011-2016 di Kota Malang menghasilkan bahwa untuk setiap harga cabai rawit yang mengalami kenaikan 1% maka dapat menyebabkan kenaikan nilai inflasi sebesar 0,0000286% (Rizaldy, 2017). Harga cabai rawit yang selalu mengalami kenaikan dan penurunan secara drastis berdampak pada pihak konsumen dan petani. Saat harga cabai rawit terlalu tinggi membuat konsumen merasa terbebani dengan harga yang terlalu tinggi, tetapi saat harga terlalu tinggi membuat petani mengalami kerugian dikarenakan pembelian hasil panen cabai rawit tidak bisa mengembalikan modal yang dikeluarkan saat proses penanaman. Data yang diambil di Kota Malang menyebutkan bahwa harga cabai rawit pernah menyentuh angka lebih dari Rp100.000,00 dari bulan Januari sampai Maret 2017 dengan kenaikan tertinggi sebesar Rp136.000,00 tanggal 23 Februari 2017 dan harga terendah sebesar Rp12.200,00 pada tanggal 13 Oktober 2017 (Disperindag Jatim, 2018).

Berdasarkan permasalahan tersebut dibutuhkan adanya penelitian tentang prediksi harga cabai rawit. Diharapkan dengan adanya penelitian ini harga cabai rawit dapat diprediksi sehingga pihak pemerintah Kota Malang dapat melakukan tindakan pencegahan terhadap harga yang fluktuatif, perbedaan harga pada setiap pasar, dan mampu menjaga stabilisasi harga di setiap pasar agar inflasi serta harga yang didapatkan konsumen dan petani tidak saling merugikan salah satu pihak. Maka pada penelitian ini dilakukan prediksi menggunakan algoritme *Extreme Learning Machine* (ELM).

Algoritme ELM sudah banyak dipakai dalam penelitian karena algoritme tersebut mampu menyeselesaikan permasalahan yang kompleks. Algoritme ELM juga memiliki hasil prediksi dengan akurasi yang baik, serta memiliki kinerja yang sangat cepat dalam pembelajaran dengan mampu belajar ribuan kali lebih cepat dari algoritme konvensional lainnya untuk menyelesaikan kasus prediksi dan klasifikasi yang besar dan kompleks (Huang, Zhu & Siew, 2004). Algoritme ELM dipilih karena memiliki hasil untuk kasus prediksi yang lebih baik dengan nilai nilai *error* yang dihasilkan lebih kecil dan waktu eksekusi yang cepat jika dibandingkan dengan algoritme konvensional lainnya seperti *Backpropagation*. Perbandingan hasil nilai MSE untuk algortime ELM dan *Backpropagation* sebesar 5,4743e-005 dan 0,0178 sedangkan untuk waktu eksekusi antara ELM dan *Backpropagation* sebesar 0,0673 *seconds* dan 0,35198 *seconds* (Huixuan, Yuchao & Zhang, 2015). Penelitian sebelumnya tentang *Extreme Learning Machine* untuk prediksi cairan pemodelan *Magnetorheological.* Dengan menggunakan perhitungan nilai *error* menggunakan *Root Mean Square Error* (RMSE) menghasilkan nilai dari fungsi aktivasi *hard limit* dan *triangular basis function* dengan nilai *error* terkecil pada proses *training* sebesar 0,126 dan 0,323. Sedangkan pada proses *prediction* fungsi aktivasi *sigmoid* dan *Radial Basis Function* (RBF) menghasilkan nilai *error* terkecil sebesar 1,427 dan 1,807 (Bahiuddin et al., 2018). Setelah itu, penelitian tentang prediksi menggunakan algoritme ELM dengan kasus pada harga untuk daging sapi di Kota Malang diperoleh hasil nilai dari perhitungan *error* antara data aktual dan data hasil

prediksi menghasilkan nilai evaluasi MAPE terkecil sebesar 0,344%. Sehingga dari hasil tersebut algoritme ELM mampu menyelesaikan kasus yang berkaitan permasalahan prediksi dengan optimal dengan menghasilkan tingkat *error* terkecil (Mosabeth, Furqon, & Wihandika, 2018). Pada penelitian sebelumnya untuk penyelesaian kasus prediksi pada harga cabai rawit nasional. Dilakukan prediksi harga dengan menerapakan sebuah metode *High Order Fuzzy Times Series Multifactors* dan terbukti dapat memprediksi dengan baik dengan hasil perhitungan nilai MSE sebesar 20374,19 (Gumelar, Setiawan & Adikara, 2019).

Dari permasalahan dan referensi penelitian sebelumnya, maka penelitian ini akan melakukan prediksi pada harga cabai rawit di Kota Malang agar dapat menghasilkan nilai prediksi yang mendekati harga sebenarnya sehingga harga yang fluktuatif bisa prediksi dengan baik. Berdasarkan latar belakang permasalahan tersebut, maka peneliti membuat judul "Prediksi Harga Cabai Rawit di Kota Malang menggunakan Algoritme *Extreme Learning Machine* (ELM)".

1.2 Rumusan Masalah

Berdasarkan pemaparan permasalahan pada bagian latar belakang tersebut, adapun rumusan masalah sebagai berikut.

1. Bagaimana mengetahui nilai parameter yang optimal dari setiap hasil pengujian prediksi harga cabai rawit di Kota Malang menggunakan algoritme *Extreme Learning Machine* (ELM)?
2. Bagaimana nilai *error rate* dengan *Mean Absolute Percentage Error* pada hasil prediksi harga cabai rawit di Kota Malang menggunakan algoritme *Extreme Learning Machine* (ELM)?

1.3 Tujuan

Berdasarkan pemaparan rumusan masalah di atas, adapun tujuan yang ingin diperoleh dari penelitian ini sebagai berikut.

1. Dapat mengetahui nilai parameter yang optimal dari setiap hasil pengujian prediksi harga cabai rawit di Kota Malang menggunakan algoritme *Extreme Learning Machine* (ELM).
2. Dapat mengetahui nilai *error rate* dengan *Mean Absolute Percentage Error* pada hasil prediksi harga cabai rawit di Kota Malang menggunakan algoritme *Extreme Learning Machine* (ELM).

1.4 Manfaat

Berdasarkan penelitian ini, maka manfaat yang diperoleh dari penelitian sebagai berikut.

1. Mampu memperoleh pengetahuan dalam penerapan Algoritme *Extreme Learning Machine*.
2. Mampu menyelesaikan masalah dalam prediksi harga cabai rawit di Kota Malang dengan tepat melalui pengujian tingkat *error rate*.

1.5 Batasan Masalah

Berdasarkan penelitian ini, batasan masalah dalam penelitian ini sebagai berikut.

1. Data yang digunakan diambil dari *website* Sistem Informasi Ketersediaan dan Perkembangan Harga Bahan Pokok di Jawa Timur (SISKAPERBAPO) yang bersumber dari Dinas Perindustrian dan Perdagangan Jawa Timur dengan *link* http://siskaperbapo.com/harga/tabel diakses pada tanggal 7 Januari 2019.
2. Jumlah data yang digunakan sebanyak 730 data berdasarkan harga rata-rata cabai rawit di lima pasar yaitu Pasar Dinoyo, Pasar Blimbing, Pasar Tawangmangu, Pasar Oro-Oro Dowo, dan Pasar Klojen setap hari dari tanggal 1 Januari 2017 sampai 31 Desember 2018 di Kota Malang.
3. Pengujian yang dilakukan untuk mencari nilai hasil perhitungan *error rate* yang optimal pada *Extreme Learning Machine* menggunakan *Mean Absolute Percentage Error* (MAPE).

1.6 Sistematika Pembahasan

Pada bagian ini akan menjelaskan dari pelaksanaan penelitian yang digunakan sebagai gambaran serta uraian dari laporan penelitian ini secara umum dibagi menjadi tujuh bab. Penjelasan singkat uraian dari beberapa bagian bab antara lain:

BAB 1 PENDAHULUAN

Pada bagian ini akan membahas mengenai ide dasar dilakukan sebuah penelitian yang dijelaskan dalam subbab dari latar belakang masalah, rumusan masalah, tujuan, manfaat, batasan masalah, dan sistematika pembahasan dalam penelitian.

BAB 2 LANDASAN KEPUSTAKAAN

Pada bagian ini akan membahas mengenai dasar teori dari penelitian ini dan kajian pustaka yang digunakan dari penelitian ini yang telah dilakukan sebelumnya sebagai bahan referensi untuk mendukung penelitian yang berhubungan pada penelitian ini dan menjelaskan dasar teori yang digunakan mengenai prediksi, cabai rawit, Jaringan Saraf Tiruan, fungsi aktivasi, normalisasi data, *denormalisasi* data, *Extreme Learning Machine*, evaluasi MAPE.

BAB 3 METODOLOGI PENELITIAN

Pada bagian ini akan membahas mengenai alur penelitian yang dijelaskan pada subbab dari tipe penelitian, strategi penelitian, lokasi dilakukan penelitian, teknik dari pengumpulan data, sumber data penelitian, teknik analisis data, implementasi dari algoritme, peralatan pendukung yang digunakan, pengujian dan analisisnya, dan kesimpulan dan saran.

BAB 4 PERANCANGAN

Pada bagian ini akan membahas mengenai perancangan dari hasil ini akan dipaparkan mengenai formulasi permasalahan, desain arsitektur sistem,

perancangan algoritme dari mulai diagram alir dari proses *training*, *testing*, *evaluasi*, perhitungan manualisasi dan skenario pengujian yang digunakan.

BAB 5 IMPLEMENTASI

Pada bagian ini akan membahas mengenai subbab penerapan dari proses pengerjaan implementasi yang dipaparkan mengenai spesifikasi perangkan yang digunakan dan implementasi algoritme ELM dengan penjelasan dari *source code* sesuai bahasa pemrograman.

BAB 6 HASIL DAN PEMBAHASAN

Pada bagian ini akan dilakukan pembahasan tentang subbab dari hasil dan pembahasan yang mana akan menjelaskan tentang hasil dari pengujian. Setelah hasil pengujian telah dilakukan akan ditampilkan dalam bentuk grafik untuk dilakukan pembahasan agar mengetahui sejauh mana algoritme dapat bekerja dengan optimal dalam menjawab pertanyaan yang di permasalahkan.

BAB 7 PENUTUP

Pada bagian ini akan membahas mengenai subbab untuk bagian dari penutup yang berisi bagian kesimpulan dan saran, yang mana pada bagian kesimpulan akan menjelasakan kesimpulan hasil akhir yang didapatkan melalui penelitian yang dilakukan dan saran sebagai pengembangan dari kekurangan penelitian saat ini untuk dilanjutkan pada penelitian selanjutnya.

BAB 2 LANDASAN KEPUSTAKAAN

2.1 Kajian Pustaka

Dari hasil kajian yang dilakukan pada penelitian sebelumnya, penelitian sebelumnya yang telah dilakukan oleh Bahiuddin, Mazlan, Shapiai, & Ubaidillah (2018) melakukan prediksi cairan untuk pemodelan *Magnetorheological* dengan menggunakan algoritme ELM. Pada skema pemodelan pengujian menggunakan fungsi aktivasi yaitu *hard limit, sigmoid, sine, triangular basis function,* dan *Radial Basis Function* (RBF). Dengan menggunakan perhitungan nilai *error* menggunakan *Root Mean Square Error* (RMSE) menghasilkan nilai dari fungsi aktivasi *hard limit* dan *triangular basis function* dengan nilai *error* terkecil pada proses *training* sebesar 0,126 dan 0,323. Pada proses *prediction* fungsi aktivasi *sigmoid* dan *Radial Basis Function* (RBF) menghasilkan nilai *error* terkecil sebesar 1,427 dan 1,807. Hasil nilai fungsi aktivasi terbaik yang dapat digunakan dengan selisih nilai RMSE *training* dan RMSE *prediction* terkecil adalah *sigmoid dan sine*.

Pada pustaka kedua penelitian dilakukan oleh Fachrony, Cholissodin, & Santoso (2018) tentang prediksi tentang beban pemanasan atau *Heating Load* (HL) dan beban pendinginan atau *Cooling Load* (CL) menggunakan algoritme ELM. Data dalam penelitian menggunakan data tentang bangunan yang memiliki 3 fitur seperti *Relative Compactness* (RC), *Surface Area* (SA), dan *Wall Area* (WA). Hasil nilai *error* pada perhitungan MAPE sebesar 24,73% dengan waktu eksekusi 0,0176 *seconds.* Parameter fungsi aktivasi yaitu *sigmoid biner,* dengan 3 nilai *input,* 1 *hidden neuron,* jumlah *output target* sebanyak 2, dan jumlah data sebanyak 130 data. Pada pustaka ketiga penelitian yang telah dilakukan oleh Anggraini (2017) tentang prediksi nilai tukar mata uang asing dengan ELM. Data yang digunakan berasal dari Bank Indonesia. Penelitian ini mendapatkan hasil yang cukup baik pada proses pelatihan menggunakan perhitungan *error* dengan hasil *Mean Square Error* (MSE) pada nilai kurs jual sebesar 0,001296 dan kurs beli sebesar 0,001099, sedangkan pada proses prediksi nilai kurs jual sebesar 0,000368 dan nilai kurs beli sebesar 0,001596. Hasil tersebut diperoleh dari perhitungan prediksi untuk bulan Februari dengan parameter pada *hidden node* yang paling optimal sejumlah 25 *hidden node.*

Pada pustaka keempat dilakukan penelitian oleh Giusti, Widodo, & Adinugroho (2018) tentang prediksi penjualan Mi dengan algoritme ELM. Data pada penelitian diperoleh dari data penjualan mie harian di Kober Mie Setan Cabang Soekarno Hatta. Dari hasil yang didapatkan melalui nilai *Mean Square Error* (MSE) diperoleh tingkat *error* terkecil sebesar 0,0176 untuk pengujian jumlah *neuron* sebanyak 7, tingkat *error* sebesar 0,0171 untuk hasil dari pengujian jumlah data *training* sebesar 80% dan data *testing* sebesar 20%, tingkat *error* sebesar 0,0171 untuk pengujian jumlah fitur sebanyak 7. Hasil rata-rata nilai *error* yang dihasilkan melalui pengujian tersebut sebesar 0,0171. Pada pustaka kelima dilakukan penelitian oleh Mosabeth, Furqon, & Wihandika (2018) tentang prediksi harga daging sapi di Kota Malang. Hasil penelitian diperoleh

hasil nilai dari perhitungan *error* antara data aktual dan data hasil prediksi menghasilkan nilai evaluasi MAPE terkecil sebesar 0,344%. Nilai tersebut diperoleh dengan variasi nilai parameter pada algoritme ELM berupa perbandingan antara jumlah dari data *training* yang akan dipakai dan data *testing* sebagai data uji sebesar 90%:10%, pada nilai *input weight* menggunakan rentang nilai antara -1 dan 1, fungsi aktivasi dengan rumus *sigmoid biner,* dan jumlah penggunaan fitur data sebanyak 3 fitur. Dari nilai parameter tersebut algoritme *Extreme Learning Machine* mampu berjalan dengan optimal dengan tingkat *error* terkecil.

Pustaka terakhir dilakukan penelitian oleh Gumelar, Setiawan, & Adikara (2019) yang mana dilakukan prediksi menggunakan metode *High Order Fuzzy Times Series Multifactors*. Sumber dari data pada penelitian diperoleh melalui Badan Pusat Statistik (BPS) pada tahun 2010-2015. Hasil penelitian menghasilkan nilai nilai MSE terkecil sebesar 20374,19 dengan nilai *order* sebesar 10 dan nilai konstanta sebesar 30. Berdasarkan hasil kajian dalam penelitian sebelumnya dalam mendukung penelitian saat ini dapat ditampilkan pada bagian Tabel 2.1.

Tabel 2.1 Daftar Kajian Pustaka

No.	Penelitian	Objek	Metode	Hasil
1.	(Bahiuddin et al., 2018)	Data Pemodelan cairan *Magnetorheological* (MR).	*Extreme Learning Machine* dengan skema pengujian berdasarkan *activation function* dan nilai error menggunakan *Root Mean Square Error* (RMSE)	Hasil pemodelan skema pengujian menggunakan *Extreme Learning Machine* bahwa *sine* dan *sigmoid* adalah *activation function* terbaik dengan hasil nilai RMSE terkecil.
2.	(Fachrony, Cholissodin & Santoso, 2018)	Data Bangunan dengan 3 fitur yaitu *Relative Compactness* (RC), *Surface Area* (SA), *Wall Area* (WA) dan 2 target yaitu *Heating Load* (HL) serta *Cooling Load* (CL).	*Extreme Learning Machine* dengan fungsi aktivasi *sigmoid biner* dan nilai evaluasi menggunakan *Mean Absolute Error Percentage* (MAPE).	Hasil pengujian menggunakan *Extreme Learning Machine* didapatkan hasil nilai evaluasi terbaik MAPE sebesar 24,73% dengan waktu eksekusi sebesar 0,0176 detik.
3.	(Anggraini, 2017)	Data fluktuatif untuk mata uang Dollar Amerika (USD) dengan mata	*Extreme Learning Machine* dengan perhitungan akurasi *error* melalui perhitungan *Mean*	Algoritme seperti *Extreme Learning Machine* pada pelatihan dapat memperoleh hasil

Tabel 2.1 Daftar Kajian Pustaka (Lanjutan)

		uang Rupiah (IDR) pada nilai bagian kurs beli dan nilai kurs jual.	*Square Error* (MSE).	nilai MSE pada bagian kurs jual sebesar 0,001296 dan kurs beli sebesar 0,001099, sedangkan pada proses prediksi nilai *error* kurs jual sebesar 0,000368 dan kurs beli sebesar 0,001596.
4.	(Giusti, Widodo & Adinugroho, 2018)	Data penjualan restoran Kober Mie Setan cabang Soekarno Hatta.	*Extreme Learning Machine* untuk kasus dalam hal prediksi melalui perhitungan nilai *error Mean Square Error* (MSE).	Algoritme *Extreme Learning Machine* mampu menghasilkan hasil pengujian dengan rata-rata untuk nilai MSE yang terkecil sebesar 0,0171.
5.	(Mosabeth, Furqon & Wihandika, 2018)	Data harga untuk daging sapi di Indonesia.	*Extreme Learning Machine* dengan perhitungan *error* menggunakan *Mean Absolute Percentage Error* (MAPE).	Hasil pengujian algoritme *Extreme Learning Machine* mampu perhitungan nilai menggunakan MAPE sebesar 0,344%.
6.	(Gumelar, Setiawan & Adikara, 2019)	Data Harga Bahan Cabai Nasional.	*High Order Fuzzy Times Series Multifactors* dengan perhitungan *error* menggunakan *Mean Square Error* (MSE).	Hasil yang diperoleh penelitian ini pada harga cabai sebesar 20374,19.

Dari referensi yang digunakan oleh penelitian sebelumnya menghasilkan bahwa hasil dari algoritme *Extreme Learning Machine* mampu untuk diterapkan ke dalam proses perhitungan permasalahan prediksi dengan hasil evaluasi yang rendah. Oleh karena itu, pada penelitian ini digunakan sebagai prediksi dalam menentukan harga cabai rawit di Kota Malang.

2.2 Prediksi

Prediksi merupakan suatu model pendekatan yang diharapkan dapat menghasilkan ramalan atau prakiraan terhadap nilai sehingga dapat menjelaskan keadaan mendatang melalui hasil data dari waktu sebelumnya secara proses perhitungan matematis. Prediksi mempunyai peran yang sangat penting dalam proses penentuan hasil terkait sebuah peristiwa yang akan terjadi sehingga dapat

dipersiapkan dengan baik yang akan dibutuhkan (Mosabeth, Furqon & Wihandika, 2018). Hasil prediksi diharapkan dapat sekecil mungkin memiliki nilai *error* dan berusaha memberikan jawaban semirip mungkin dengan data aktual. Dalam prediksi memiliki jenis prediksi berdasarkan sifat yang telah disusun dan jangka waktu (Rusdiana, 2014). Berdasarkan sifat yang digunakan dalam prediksi memiliki 2 jenis sebagai berikut.

1. Prediksi kualitatif adalah prediksi yang digunakan berdasarkan data *non-numerik* yang berasal dari masa lalu dan umumnya berupa pendapat, pengetahuan, intuisi, dan pengalaman.
2. Prediksi kuantitatif adalah prediksi yang menggunakan metode berdasarkan data *numerik* dari beberapa waktu yang lalu.

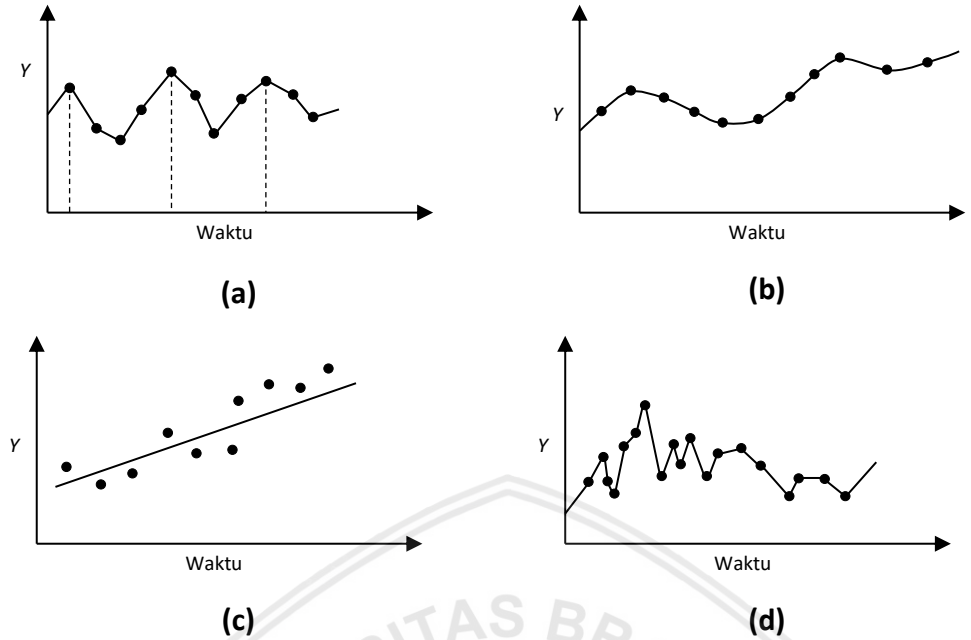
Berdasarkan jangka waktu yang diambil pada proses prediksi terbagi menjadi 3 jenis sebagai berikut.

1. Prediksi jangka pendek adalah prediksi yang penyusunan dalam hasil prediksi jangka waktu yang digunakan penelitian selama satu tahun atau kurang.
2. Prediksi jangka menengah adalah prediksi yang penyusunan dalam hasil prediksi jangka waktu yang digunakan penelitian selama satu hingga lima tahun.
3. Prediksi jangka panjang adalah prediksi yang penyusunan dalam hasil prediksi jangka waktu yang digunakan penelitian selama lebih dari lima tahun.

Dalam pendekatan menggunakan model prediksi perlu diketahui jenis pola data. Analisis jenis pola data memiliki empat jenis pola sebagai berikut (Ukhra, 2014).

1. Pola Musiman adalah pola data yang bergerak secara berulang-ulang dan fluktuatif yang terjadi secara periodik dalam rentang waktu 1 tahun, pola yang berulang bisa terjadi setiap triwulan, bulanan, mingguan, dan harian.
2. Pola Siklis adalah pola data yang bergerak dari pengaruh siklus yang fluktuatif yang terjadi dalam rentang waktu yang panjang.
3. Pola *Trend* adalah pola data yang bergerak dalam rentang waktu yang panjang dengan pergerakan data cenderung mengalami kenaikan atau penurunan.
4. Pola *Irregular* adalah pola data yang bergerak secara acak dan tidak beraturan sehingga dapat menimbulkan pola yang fluktuatif.

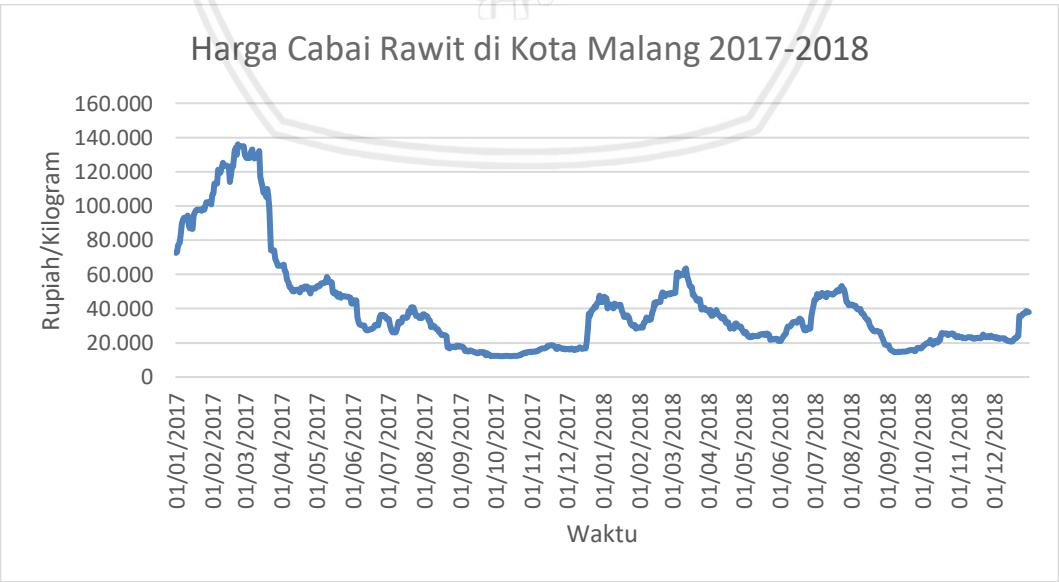
Berikut gambar jenis pola data yang dapat dilihat pada Gambar 2.1.



Gambar 2.1 Jenis Pola Data, (a) Musiman, (b) Siklis, (c) *Trend*, (d) *Irregular*

Sumber: (Hanke & Wichern, 2009)

Prediksi yang dilakukan termasuk jenis prediksi kuantitatif dengan nilai setiap data berupa *numerik* dengan satuan harga rupiah/kilogram. Berdasarkan jangka waktu yang diambil termasuk dalam prediksi dengan jangka waktu pendek karena data yang diprediksi dalam penelitian menghasilkan data dalam waktu harian. Dari analisis data yang digunakan data memiliki pola data berupa pola *Irregular* karena pola pergerakan yang dihasilkan tidak dapat diketahui dengan pasti dengan bergerak tidak beraturan dan acak. Berikut grafik harga cabai rawit pada Gambar 2.2.



Gambar 2.2 Grafik Harga Cabai Rawit

Sumber: (Disperindag Jatim, 2018)

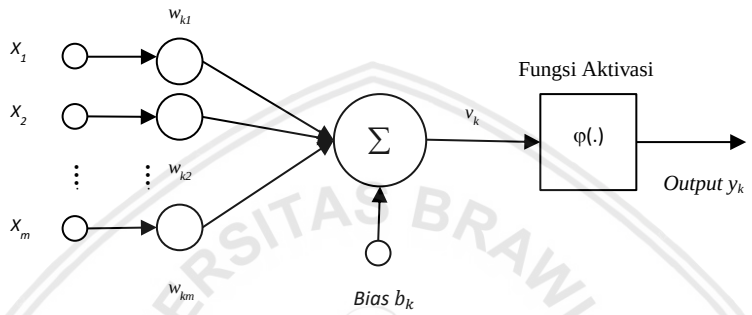
2.3 Cabai Rawit

Cabai rawit (*Capsicum frustescens L*.) merupakan salah satu tanaman dari sekitar 20-30 spesies dalam genus *Capsicum* yang telah banyak dibudidayakan (Kartikasari, Purnamaningsih, & Soetopo, 2016). Cabai rawit biasa dikenal dengan sebutan cabai kecil yang memiliki tingkat rasa pedas yang cukup tinggi, sehingga membuat penikmat rasa pedas selalu menggunakan bahan cabai rawit sebagai bahan tambahan untuk menambah cita rasa makanan. Cabai rawit termasuk hasil dari komoditas pertanian yang ada di Indonesia yang selalu dibutuhkan sebagai bahan kebutuhan sehari-hari. Selain rasa pedas yang dihasilkan, tanaman tersebut juga memiliki tingkat kandungan vitamin C yang tinggi. Kandungan vitamin C dalam cabai rawit dapat berfungsi sebagai aktioksidan di dalam tubuh sehingga mampu untuk meningkatkan daya tahan tubuh yang diserap melalui kalsium dalam tubuh (Rosmainar et al., 2018). Dengan vitamin yang ada dalam cabai rawit dan ciri khas rasa yang dihasilkan membuat cabai rawit dikenal banyak orang dan mudah untuk ditemukan. Beberapa tempat yang menjual cabai rawit dapat ditemukan di pasar tradisional dan pasar modern. Tentunya dengan harga yang bervariasi, konsumen bisa mendapatkan yang sesuai dengan ekonomi. Begitu banyak makanan yang hampir semua menggunakan bahan dasar cabai rawit membuat harga cabai memiliki nilai ekonomi yang tinggi. Hampir bisa dilihat harga cabai rawit setiap hari selalu mengalami fluktuatif harga. Cabai rawit awalnya hanya dijadikan sebagai bahan kebutuhan pelengkap makanan akan tetapi saat ini sudah dianggap sebagai bahan kebutuhan pokok oleh masyarakat mengingat tidak lengkap jika hasil makanan tidak disajikan tanpa menggunakan bahan cabai rawit. Dengan harga yang sangat tinggi tersebut membuat masyarakat mengalami kebingungan menghadapi permasalahan harga cabai rawit. Harga cabai rawit dijual dalam satuan rupiah/kilogram.

2.4 Jaringan Saraf Tiruan

Jaringan Saraf Tiruan (JST) merupakan sebuah cabang model dari *machine learning* yang dapat menyelesaikan sebuah masalah melalui perhitungan komputasi yang diadaptasi dengan melihat cara kerja jaringan dari saraf otak manusia. Penyelesaian masalah pada JST jika data bernilai kontinu perlu dilakukan sebuah proses *pre-processing* untuk mengubah bentuk data kontinu menjadi bentuk fitur, umumnya masalah yang bernilai kontinu adalah data *time series* seperti kasus prediksi. Pada kasus JST dengan data bernilai kontinu, maka perlu didefinisikan terlebih dahulu pola masukan dan target dari data *time series* dengan menggunakan analisis teknikal (Cholissodin & Sutrisno, 2018). Analisis teknikal didefinisikan sebagai pola masukan berdasarkan *n* data hari sebelumnya dengan minimal menggunakan dua data hari dan target pada data hari berikutnya pada data bernilai kontinu (Suyanto, 2014). Pola masukan dan target akan dilakukan proses pembelajaran JST yaitu dengan meniru cara kerja saraf otak pada manusia yang memiliki *neuron* sebagai jaringan penghubung dengan *neuron* yang lain. *Neuron* memiliki informasi berupa angka yang bisa diberikan ke

neuron lainnya. Dalam penerapan JST nilai x_1, x_2, sampai x_m, akan berfungsi sebagai *neuron* yang memiliki informasi. Untuk menghubungkan antar *neuron* dibutuhkan input *weight* (w_k). Dalam JST diperlukan proses *training* yang berfungsi untuk melatih jaringan pada setiap input *neuron* agar pada saat dilakukan prediksi jaringan mampu menjawab dengan baik. Pada bagian *neuron* pada jaringan sel saraf memiliki tiga bagian yaitu bagian fungsi penjumlah atau bisa disebut *summing function*, bagian fungsi aktivasi atau bisa disebut *activation function*, dan bagian keluaran atau *output* (Fikriya, Irawan & Soetrisno, 2017). Berikut gambar arsitektur untuk bagian jaringan saraf tiruan terlihat di Gambar 2.3.



Gambar 2.3 Arsitektur Jaringan Saraf Tiruan

Sumber: (Fikriya, Irawan & Soetrisno, 2017)

2.5 Fungsi Aktivasi

Dalam pengertian fungsi aktivasi diartikan sebagai suatu fungsi yang terdapat di dalam Jaringan Saraf Tiruan yang mampu melakukan proses pembelajaran jaringan dalam memahami permasalahan yang kompleks. Dari permasalahan dapat dilakukan pemetaan dengan cara mengubah nilai melalui fungsi aktivasi antara sinyal input menuju sinyal *output*. Penerapan fungsi aktivasi setelah nilai input (x) dikali dengan nilai bobot (w). Dalam fungsi aktivasi terdapat 2 jenis yaitu fungsi aktivasi *linear* dan fungsi *non-linear*. Fungsi aktivasi *linear* atau identitas adalah sebuah fungsi yang akan menghasilkan garis lurus atau *linear* yang menggambarkan identitas asli. Fungsi aktivasi *linear* tidak mampu beradaptasi dengan baik terhadap variasi bentuk data karena hanya bisa mengenal bentuk data yang bersifat *linear*. Fungsi *non-linear* adalah fungsi aktivasi yang berbentuk bentuk kurva. Beberapa fungsi aktivasi non-linear menurut Kusumadewi (2003) dan Bahiuddin (2018).

1. Fungsi *Sigmoid Biner*

Rumus fungsi aktivasi menggunakan *sigmoid* terdapat di Persamaan 2.1.

$$g(x) = \frac{1}{1+\exp(-x)} \quad (2.1)$$

2. Fungsi *Sin*

Rumus fungsi aktivasi menggunakan *sine* terdapat di Persamaan 2.2.

$$g(x) = \sin(x) \quad (2.2)$$

3. Fungsi *Sigmoid Bipolar*

Rumus fungsi aktivasi menggunakan *sigmoid bipolar* terdapat di Persamaan 2.3.

$$g(x) = \frac{1-\exp(-x)}{1+\exp(-x)} \quad (2.3)$$

4. Fungsi *Radial Basis*

Rumus fungsi aktivasi menggunakan *radial basis* terdapat pada Persamaan 2.4.

$$g(x) = \exp(-x^2) \quad (2.4)$$

Keterangan:

$g(x)$ = Fungsi *g* dengan nilai input *x*

x = Nilai input pada fungsi aktivasi

exp = Basis untuk bilangan logaritma alami atau dapat disebut bilangan *Euler* dengan nilai pembulatan 2,71828183

sin = Fungsi *sin*

2.6 Normalisasi Data

Normalisasi adalah teknik yang dapat dilakukan pada data dalam proses untuk memetakan data ke dalam rentang tertentu. Proses normalisasi umumnya dilakukan untuk menyeimbangkan nilai jika rentang dari nilai data memiliki selisih yang jauh dengan yang lain. Proses tersebut dilakukan sebelum proses *training*, data di proses menggunakan perhitungan normalisasi menggunakan *Min-Max Normalization*. *Min-Max Normalization* mampu menghasilkan nilai terbaik jika dibandingkan dengan *Z-Score Normalization* dengan nilai MSE rata-rata sebesar 0,0187 dan akurasi sebesar 82% (Mustaffa & Yusof, 2011). Nilai hasil *Min-Max Normalization* memiliki rentang nilai antara 0 dan 1. Pengambilan nilai X_{min} ditentukan berdasarkan nilai data terkecil yang tidak pernah dijangkau dari semua data dan nilai X_{max} berdasarkan nilai terbesar yang tidak pernah dijangkau dari semua data (Suyanto, 2014). Berdasarkan data harga cabai rawit dan hasil pakar, nilai minimum sebesar Rp5.000,00 dan nilai maksimum sebesar Rp150.000,00. Rumus normalisasi dapat dilihat di Persamaan 2.5.

$$X_n = \frac{(X_o - X_{min})}{(X_{max} - X_{min})} \quad (2.5)$$

Keterangan:

X_n = Nilai baru dari hasil normalisasi

X_o = Nilai sebelum normalisasi

X_{min} = Nilai minimum yang didapatkan melalui *dataset*

X_{max} = Nilai maksimum yang didapatkan melalui *dataset*

2.7 *Denormalisasi* Data

Denormalisasi data adalah suatu bentuk teknik pada data dengan proses yang dilakukan dalam tahap mengembalikan nilai dari data menjadi nilai aktual atau nilai sebenarnya. Hasil rumus untuk menghitung *denormalisasi* dapat dilihat pada Persamaan 2.6.

$$X_o = X_n \,.\, (X_{max} - X_{min}) + X_{min} \tag{2.6}$$

Keterangan:

X_n = Nilai hasil prediksi yang sudah normalisasi

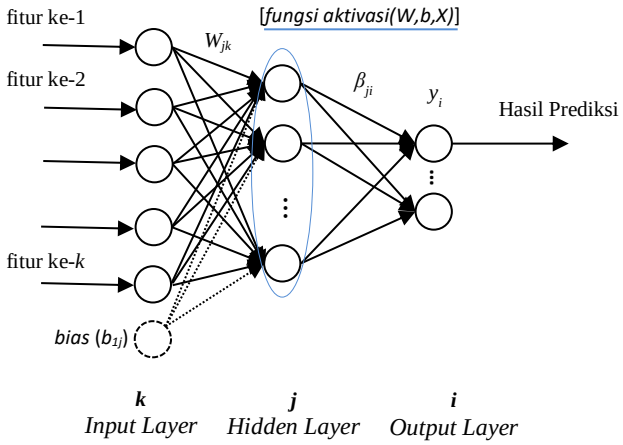
X_o = Nilai hasil *denormalisasi* data

X_{min} = Nilai minimum yang didapatkan melalui *dataset*

X_{max} = Nilai maksimum yang didapatkan melalui *dataset*

2.8 *Extreme Learning Machine* (ELM)

Algoritme *Extreme Learning Machine* adalah jenis dari algoritme yang terdapat di Jaringan Saraf Tiruan yang dipakai untuk meningkatkan perubahan dari segi kecepatan. Algoritme ELM memiliki kecepatan belajar jauh lebih cepat dibandingkan dengan algoritme *feedforward neural networks* lainnya. Pada jaringan *feedforward neural networks* menggunakan *slow gradient-based learning* hal itu membuat proses pembelajaran algoritme menjadi lambat dan semua parameter nilai yang digunakan ditentukan berdasarkan setiap iterasi (Huang, Zhu & Siew, 2004). Algoritme ELM termasuk ke dalam jenis *feedforward neural networks* yang dikenal hanya memiliki *single hidden layer* dengan demikian algortime ELM termasuk sebagai *single hidden layer feedforward neural networks* (SLFNs). Pada arsitektur algoritme terdapat tiga bagian yaitu bagian input *layer*, *hidden layer,* dan *output layer*. Pada parameter nilai input *weight* dan *hidden bias* dilakukan dengan acak dari hal tersebut algoritme ELM dapat memiliki kelebihan dalam *learning speed* yang sangat tinggi dan menghasilkan kinerja generalisasi yang baik karena dapat menghasilkan *output* dengan hasil benar pada input yang belum pernah dimasukkan proses pelatihan (Huang, Zhu & Siew, 2006). Pada algoritme ELM terdapat 2 tahapan proses yaitu tahapan proses *training* data dan tahapan proses *testing* data. Dalam proses menuju tahapan tersebut dapat dilakukan sebuah proses untuk analisis data menggunakan analisis teknikal dan setelah itu dilakukan pembagian data untuk data *training* dan data *testing*. Data *training* berfungsi untuk membangun model pelatihan agar algoritme dapat melakukan prediksi dan data *testing* berfungsi sebagai data uji untuk mengukur seberapa baik algoritme ELM bekerja (Hanifa, Adiwijaya & Al-faraby, 2017). Proses pembagian data dilakukan secara variasi sesuai dengan skenario pengujian untuk persentase dari pembagian data tersebut. Untuk mengetahui gambaran arsitektur algoritme ELM dapat diamati pada bagian Gambar 2.4.



Gambar 2.4 Arsitektur Algoritme *Extreme Learning Machine*

Sumber: (Cholissodin & Sutrisno, 2018)

Pada tahapan algoritme ELM dijelaskan bahwa sebelum menuju tahapan dari ELM diperlukan terlebih dahulu proses *pre-processing* data dan normalisasi data, setelah proses tersebut kemudian melakukan tahapan algoritme ELM, selanjutnya setelah tahapan algoritme ELM akan dilakukan proses *denormalisasi* data dan menghitung hasil *error* menggunakan MAPE. Berikut tahapan dari langkah-langkah algoritme ELM pada proses *training* dan *testing* dengan menggunakan nilai *bias* (Fachrony, Cholissodin & Santoso, 2018).

2.8.1 Proses *Training*

Pada bagian *training* di dalam algoritme ELM bertujuan untuk memberikan pembelajaran pada sistem dengan menggunakan data *training* sebagai data untuk pelatihan algoritme agar pada saat proses *testing* sistem mampu menangani berbagai bentuk data baru. Berikut adalah tahapan dari algoritme ELM dalam proses *training* dengan menggunakan nilai *bias*.

1. Inisialisasi bobot *input* dan *bias* yang dilakukan secara *random*. Inisialisasi nilai yang diberikan pada bobot *input* dengan rentang [-1,1] (Cao et al., 2017) dan ukuran matriks sebesar *j* (banyak *hidden neuron*) x *k* (banyak *input neuron*). Lalu pada nilai *bias* untuk rentang [0,1] dengan matriks sebesar 1 x banyak *hidden neuron* (Ertuğrul & Kaya, 2014).
2. Menghitung nilai H_{init}

   Sebelum proses menghitung nilai keluaran *hidden layer* terlebih dahulu dilakukan perkalian antara nilai input *weight* dengan nilai pada fitur dan dijumlahkan dengan nilai *bias*. Proses perhitungan tersebut dapat dilihat pada proses menghitung H_{init} pada Persamaan 2.7.

   $$H_{init} = X_{training}.W^T + bias \quad (2.7)$$

   Keterangan:

   H_{init} = Matriks H_{init}

$X_{training}$ = Matriks nilai dari data *training*

W^T = Matriks transposisi dari input *weight*

3. Menghitung keluaran *hidden layer* (*H*)

   Proses perhitungan ini dilakukan dengan memakai fungsi aktivasi dari nilai H_{init} pada Persamaan 2.7. Fungsi *sigmoid biner* digunakan sebagai bentuk proses dari hasi pada proses *H*. Rumus menghitung untuk fungsi aktivasi sesuai rumus *sigmoid biner* dapat dilihat di Persamaan 2.8.

   $$H = \frac{1}{1+\exp(-H_{init})} \quad (2.8)$$

   Keterangan:

   H = Matriks *output hidden layer* atau *H*

   exp = Basis untuk bilangan logaritma alami atau dapat disebut bilangan *Euler* dengan nilai pembulatan 2,71828183

4. Menghitung matriks *Moore-Penrose Generalized Inverse* (H^+)

   Proses menghitung nilai yang dihasilkan *H plus* dibutuhkan proses inversi matriks. Pada inversi matriks pada umumnya berukuran *n* x *n*, sedangkan pada perhitungan ini ukuran matriks berukuran *n* x *m*. Sehingga diperlukan perhitungan menggunakan *Moore-Penrose generalized inverse.* Berikut persamaan H^+ dapat diamati di Persamaan 2.9.

   $$H^+ = (H^T.\ H)^{-1}.H^T \quad (2.9)$$

   Keterangan:

   H^+ = Matriks *Moore-Penrose Generalized Inverse*

   H^T = Matriks transposisi nilai *H*

   $(H^T.\ H)^{-1}$ = Inversi dari perkalian matriks transposisi *H* dengan *H*

5. Menghitung *output weight* ($\hat{\beta}$)

   Proses menghitung nilai *output weight* dapat diamati di Persamaan 2.10.

   $$\hat{\beta} = H^+.Y \quad (2.10)$$

   Keterangan:

   $\hat{\beta}$ = Matriks *output weight* pada *hidden layer*

   H^+ = Matriks *Moore-Penrose Generalized* Inversi

   Y = Matriks target

2.8.2 Proses *Testing*

Pada bagian proses *testing* ELM bertujuan menguji seberapa baik algoritme mampu memberikan hasil nilai yang dijadikan sebagai data prediksi dengan hasil *error* terkecil. Tahapan pada algoritme proses *testing* tidak memiliki perbedaan

yang jauh dengan proses *training*. Perbedaan tahapan terjadi ketika nilai input bobot yang digunakan dari hasil proses *training* dan sedangkan untuk data yang akan digunakan adalah data *testing*. Berikut tahapan *testing* pada algoritme ELM.

1. Menyimpan nilai input *weight*, *bias*, dan $\hat{\beta}$ dari hasil proses *training* data.
2. Menghitung nilai H_{init}
   Pada perhitungan H_{init} untuk proses *testing* menggunakan rumus pada Persamaan 2.11.

   $$H_{init} = X_{testing}.W^T + bias \quad (2.11)$$

   Keterangan:

   H_{init} = Matriks H_{init}

   $X_{testing}$ = Matriks nilai dari data *testing*

   W^T = Matriks transposisi dari input *weight*
3. Menghitung keluaran *hidden layer* (*H*)
   Pada perhitungan ini nilai *H testing* yang dihasilkan dari perhitungan rumus fungsi aktivasi yaitu *sigmoid biner* diambil dari Persamaan 2.8 dengan nilai input menggunakan hasil dari perhitungan H_{init} pada Persamaan 2.11.
4. Menghitung hasil prediksi ($\hat{Y}$)
   Proses perhitungan hasil prediksi dapat diamati di Persamaan 2.12.

   $$\hat{Y} = H.\hat{\beta} \quad (2.12)$$

   Keterangan:

   $\hat{Y}$ = Matriks dari hasil prediksi

   H = Matriks dari keluaran *hidden layer* dengan fungsi aktivasi

   $\hat{\beta}$ = Matriks dari *output weight*

2.9 *Mean Absolute Percentage Error* (MAPE)

MAPE merupakan proses menghitung nilai evaluasi yang bertujuan agar dapat diketahui nilai *error* dari proses pengujian atau *testing*. Menghitung nilai MAPE dilakukan untuk mencari kesalahan absolut pada setiap periode dengan selisih antara nilai hasil aktual dengan nilai hasil prediksi. Rumus menghitung nilai MAPE dapat dilihat pada Persamaan 2.13 (Darmayanti, Setiawan & Bachtiar, 2018).

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\ x\ 100\right| \quad (2.13)$$

Keterangan:

$\hat{y}_i$ = Nilai data hasil prediksi ke-*i*

y_i = Nilai data aktual ke-*i*

n = Banyak data *testing*

i = Indeks dari data

BAB 3 METODOLOGI PENELITIAN

Pada bagian untuk bab metodologi penelitian dijelaskan tentang pengerjaan dari langkah dalam menyelesaikan masalah penelitian secara sistematik untuk implementasi dari algoritme *Extreme Learning Machine* untuk prediksi harga cabai rawit di Kota Malang.

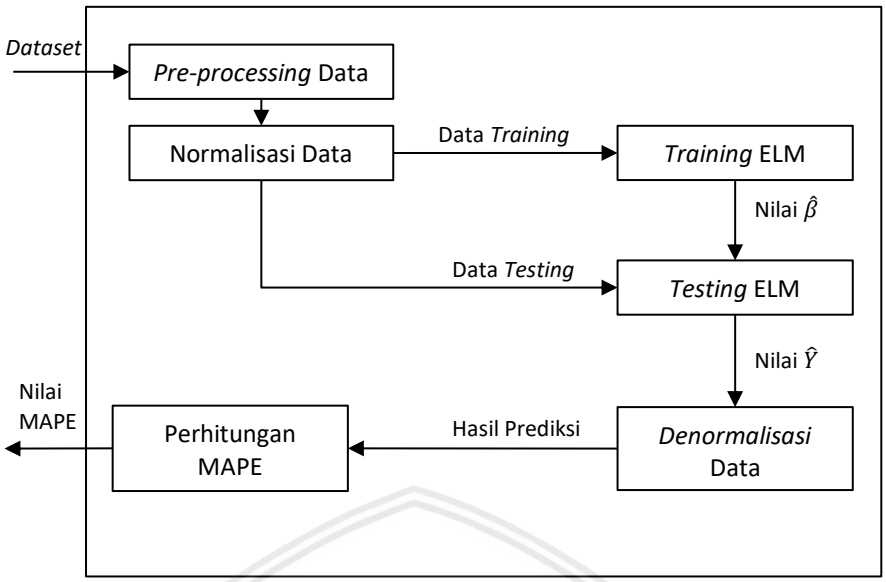
3.1 Tipe Penelitian

Tipe penelitian yang digunakan termasuk ke dalam jenis penelitian non-implementatif analitik. Tipe penelitian ini akan melakukan pembahasan terkait fenomena atau kejadian yang ada di sekitar dengan cara mengamati kondisi sekitar kemudian di peroleh hasil analisis dari data yang diteliti. Pada penelitian ini data yang digunakan berupa data kuantitatif *time series* yang menerapkan pemodelan matematis pada fenomena yang ada. Data ini didasarkan dari data harga cabai rawit di Kota Malang pada tahun 2017-2018.

3.2 Strategi Penelitian

Pada strategi penelitian untuk implementasi dari algoritme *Extreme Learning Machine* untuk prediksi harga cabai rawit di Kota Malang. Data prediksi digunakan adalah data harga cabai rawit di Kota Malang dalam bentuk data *time series* atau data didapatkan berdasarkan waktu yaitu perhari. Algoritme *Extreme Learning Machine* dipilih karena kinerja yang dihasilkan oleh algoritme ELM mampu melebihi hasil dari algoritme lain seperti *Backpropagation Neural Network,* terbukti algoritme ELM memiliki nilai MSE terkecil dan waktu eksekusi tercepat. Algoritme ELM dapat digunakan dalam penyelesaian beberapa masalah seperti prediksi atau peramalan. Pada kasus prediksi algoritme ELM dapat di implementasi dengan membagi data pada bagian *training* sebagai proses pelatihan terhadap data latih dan *testing* sebagai proses pengujian untuk menguji hasil yang optimal sesuai dengan skenario pengujian yang dibuat bervariasi.

Data dalam penelitian tahapan proses dari algoritme ELM dari fitur dan target akan dilakukan proses *pre-processing* untuk mengubah data menjadi fitur berdasarkan hari menggunakan analisis teknikal, setelah itu dilakukan normalisasi data, setelah itu dilakukan pembagian data untuk bagian data pada proses *training* dan data pada proses *testing.* Selanjutnya menjalankan proses dari *training* data yang diharapkan dapat melatih algoritme ELM agar dapat memprediksi dan hasil dari proses *training* berupa nilai ($\hat{\beta}$). Dari nilai ($\hat{\beta}$) diproses menuju *testing* yang menghasilkan nilai ($\hat{Y}$). Hasil dari proses *testing* akan dilakukan proses *denormalisasi data* untuk diubah menjadi data awal sebelum dilakukan normalisasi. Setelah diketahui nilai data hasil prediksi dan data aktual maka dilakukan proses perhitungan MAPE untuk mengetahui seberapa baik algoritme bekerja dengan menghitung nilai selisih *error* dari data. Berikut diagram blok untuk algoritme ELM dapat dilihat pada bagian Gambar 3.1.



Gambar 3.1 Diagram Blok Prediksi Harga Cabai Rawit dengan Algoritme ELM

3.3 Lokasi Penelitian

Dalam penelitian ini berlokasi di Laboratorium Riset Fakultas Ilmu Komputer Universitas Brawijaya dan menghasilkan nilai berupa informasi tentang prediksi harga cabai rawit di Kota Malang yang dibutuhkan dalam proses penelitian. Data yang digunakan berasal dari *website* Sistem Informasi Ketersediaan dan Perkembangan Harga Bahan Pokok di Jawa Timur berupa data harga cabai rawit di Kota Malang setiap hari.

3.4 Teknik Pengumpulan Data

Teknik dalam proses pengumpulan data pada penelitian dilakukan dengan menggunakan data *time series* pada harga cabai rawit di Kota Malang. Sehingga di dapatkan harga cabai rawit di Kota Malang setiap hari selama 2 tahun. Dengan analisis teknikal dari data harga cabai rawit di Kota Malang dapat dilakukan perhitungan berdasarkan data dari waktu sebelumnya yang dijadikan sebagai faktor penentu pada harga cabai. Dari data yang diperoleh sebanyak 730 data rata-rata harga cabai rawit lima pasar di Kota Malang yaitu Pasar Dinoyo, Pasar Blimbing, Pasar Tawangmangu, Pasar Oro-Oro Dowo, dan Pasar Klojen dari tanggal 1 Januari 2017 sampai 31 Desember 2018 yang didapatkan melalui website Sistem Informasi Ketersediaan dan Perkembangan Harga Bahan Pokok di Jawa Timur. Data diunduh pada tanggal 7 Januari 2019 dari *link* http://siskaperbapo.com/harga/tabel.

3.5 Data Penelitian

Data yang digunakan dalam penelitian ini adalah data harga cabai rawit di Kota Malang. Data harga cabai rawit dikumpulkan hingga berjumlah sebanyak 730 data. Pada penelitian ini untuk melakukan validasi data didapatkan melalui wawancara pakar dengan narasumber Ir. Asih Siswanti dari Dinas Perdagangan

Kota Malang yang menjabat sebagai Kepala Seksi Stabilisasi Harga. Wawancara dilakukan di Dinas Perdagangan Kota Malang yang beralamat di Jalan Simpang Terusan Danau Sentani No. 3, Kota Malang, Jawa Timur, 65138 pada tanggal 14 Februari 2019. Dari data tersebut akan diproses menggunakan prediksi pada algoritme ELM dengan dilakukan pembagian untuk data pada proses *training* dan data pada proses *testing*. Banyak data yang digunakan pada proses *training* dan *testing* dibuat secara bervariasi sesuai dengan skenario pengujian yang dibuat.

3.6 Teknik Analisis Data

Teknik analisis data yang digunakan pada penelitian ini menggunakan implementasi dari algoritme ELM untuk prediksi harga cabai rawit di Kota Malang adalah teknik analisis teknikal. Teknik analisis data yang digunakan menggunakan analisis teknikal yang mana data yang bersifat kontinu akan digunakan berdasarkan data dari hari sebelumnya sebagai fitur sebanyak *n* hari, sedangkan hari berikutnya sebagai target. Setalah proses analisis teknikal maka didapatkan hasil data dalam bentuk fitur yang data tersebut dapat dipakai ke dalam proses algoritme ELM. Analisis yang digunakan pada saat hasil prediksi sudah didapatkan yaitu dengan cara menggunakan perhitungan nilai MAPE yang melakukan perhitungan selisih antara data aktual dengan data hasil prediksi. Dilakukan perhitungan MAPE karena mampu menghasilkan nilai *error* dari hasil algoritme ELM dengan memperhitungkan selisih dan skala dari data.

3.7 Implementasi Algoritme

Pada implementasi algoritme akan dibuat menjadi sebuah sistem yang akan diuji tingkat hasil prediksi dengan menghitung nilai *error*. Agar hasil prediksi dapat sesuai dengan yang diinginkan maka perlu dilakukan evaluasi terhadap hasil prediksi. Perhitungan evaluasi menggunakan nilai MAPE yang menghasilkan nilai *error* dari data sebenarnya (aktual) terhadap data hasil prediksi. Dalam implementasi algoritme *Extreme Learning Machine* untuk mengukur seberapa baik algoritme maka dilakukan proses pengujian nilai MAPE.

3.8 Peralatan Pendukung

Dalam penelitian digunakan alat pendukung agar penelitian dapat berjalan dengan baik. Alat pendukung bisa berupa dalam bentuk perangkat lunak (*software*) dan perangkat keras (*hardware*).

1. Perangkat Keras

   Perangkat keras atau *hardware* yang digunakan dalam mendukung penelitian ini sebagai berikut:

   - Laptop Asus X442UQ
   - Layar monitor 14”
   - Memori RAM 8074MB

2. Perangkat Lunak

   Perangkat lunak atau *software* yang digunakan dalam mendukung penelitian ini sebagai berikut:

   - Sistem Operasi Windows 10 Home Single Language 64-bit
   - Visual Studio Code versi 1.25.1
   - Google Chrome versi 71.0.3578.98 64-bit
   - Microsoft Office 2016
   - Python 3.6.3
   - Spyder 3.2.4
   - Django 2.1.1
   - Bootstrap v4.1.3
   - Highcharts JS v7.1.1

3.9 Pengujian dan Analisis

Pengujian dilakukan agar mendapatkan hasil pembanding antara data aktual dengan data prediksi pada harga cabai rawit di Kota Malang dan untuk mengetahui parameter pada algoritme ELM yang paling optimal untuk kasus prediksi harga cabai rawit. Pada penelitian ini, akan dilakukan empat skenario pengujian dan mendapatkan hasil evaluasi *error* terkecil sebagai nilai parameter yang optimal untuk selanjutnya digunakan pada masing-masing hasil pengujian. Parameter pengujian yang akan dipakai dalam pengujian ini yaitu pengujian pertama adalah banyak fitur, pengujian kedua adalah persentase perbandingan banyak data *training* terhadap data *testing*, pengujian ketiga adalah banyak *neuron* pada *hidden layer*, dan pengujian keempat adalah fungsi aktivasi.

3.10 Kesimpulan dan Saran

Bagian kesimpulan pada penelitian ini akan memberikan penjelasan terkait dari hasil dari pengujian dan analisis pada bab sebelumnya dan menjawab mengenai rumusan masalah penelitian sebagai bentuk penyelesaian masalah yang ditanyakan. Saran bertujuan agar dalam penelitian ini masih terdapat kekurangan maka dapat dijadikan masukan yang lebih baik pada penelitian selanjutnya.

BAB 4 PERANCANGAN

Pada bagian bab perancangan akan dijelaskan mengenai perancangan algoritme, perancangan pengujian, perancangan *user interface* dan langkah-langkah perhitungan manual dalam menyelesaikan masalah penelitian secara sistematik dalam implementasi dari algoritme *Extreme Learning Machine* (ELM) pada prediksi harga cabai rawit di Kota Malang.

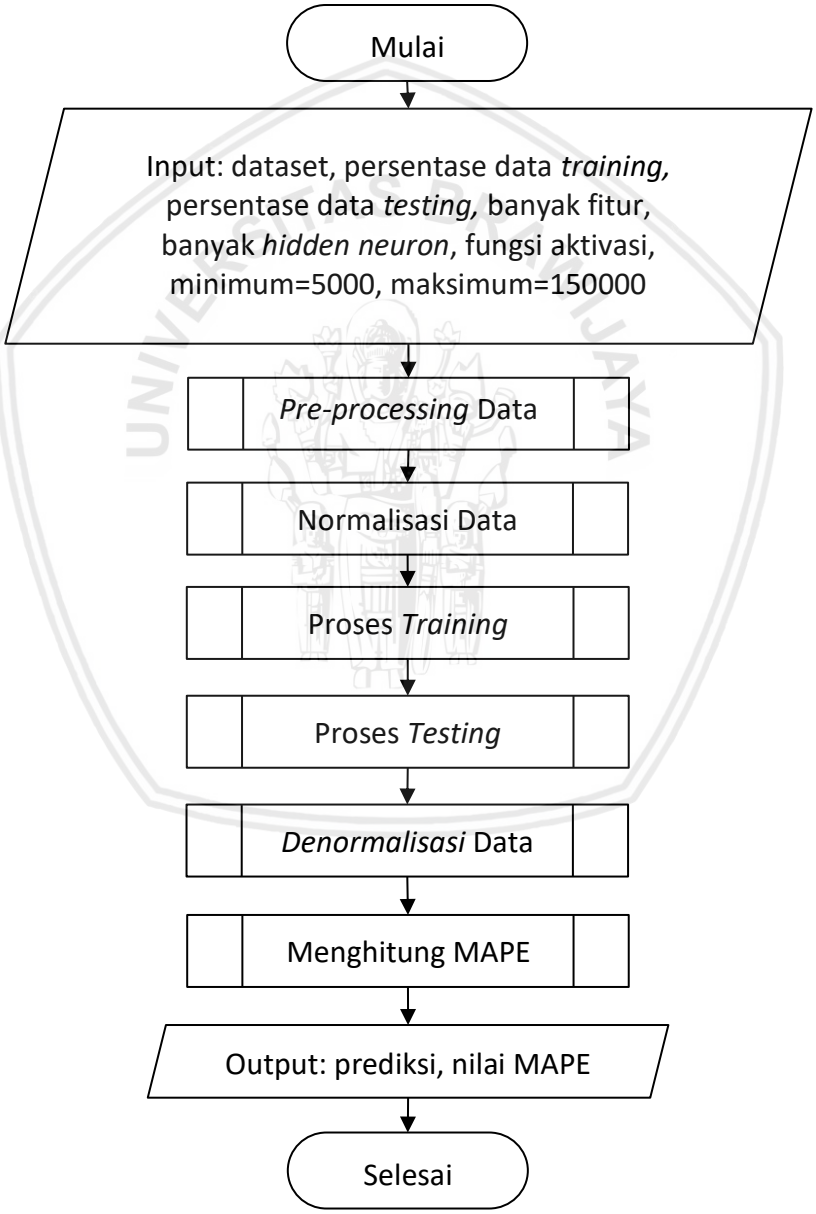
4.1 Formulasi Permasalahan

Pada kondisi sekarang banyak masyarakat di Kota Malang dalam kehidupan sehari-hari selalu mengkonsumsi cabai rawit. Hal tersebut membuat cabai rawit sudah menjadi bahan kebutuhan pokok oleh masyarakat sehingga membuat harga cabai rawit selalu mengalami perubahan yang fluktuatif, karena perubahan harga yang terlalu tinggi menyebabkan meningkatnya nilai inflasi di Kota Malang. Selain itu, harga yang ditawarkan oleh beberapa pasar sangat bervariasi dan saat penentuan harga terlalu tinggi di pasar dapat membuat konsumen terbebani tetapi jika harga terlalu rendah membuat petani cabai rawit mengalami kerugian akibat rendahnya pendapatan yang diperoleh. Dalam menyelesaikan permasalahan tersebut perlu dilakukan prediksi tentang harga cabai rawit agar harga cabai rawit stabil dan mengurangi perbedaan harga yang terlalu tinggi.

Dari permasalahan tersebut diperlukan proses perhitungan prediksi menggunakan algoritme ELM. Dalam penerapan algoritme ELM terlebih dahulu dibutuhkan data prediksi harga cabai rawit di Kota Malang dalam bentuk data sekunder yang diperoleh dari *website* SISKAPERBAPO dengan *link* http://siskaperbapo.com/harga/tabel dari tanggal 1 Januari 2017 sampai 31 Desember 2018. Dalam proses algoritme ELM dibutuhkan langkah input nilai harga cabai rawit di Kota Malang, persentase data *training*, persentase data *testing*, banyak fitur, jenis fungsi aktivasi, masukan untuk nilai bobot diperoleh melalui cara *random* dalam rentang nilai [-1,1], dan masukan nilai *bias* yang didapatkan melalui cara *random* dalam rentang nilai [0,1]. Proses selanjutnya dilakukan proses *pre-processing* untuk mengubah data menjadi bentuk fitur berdasarkan data harga sebelumnya, selanjutnya normalisasi data dari data yang sudah diinputkan dengan rentang nilai hasil normalisasi adalah [0,1]. Langkah selanjutnya menuju proses *training* menggunakan nilai dari data *training* yang hasil akhir berupa nilai *output weight*. Hasil yang didapatkan dari nilai *output weight* akan menuju proses *testing* untuk melakukan proses yang menghasilkan nilai prediksi. Sebelum nilai hasil prediksi tersebut dihitung maka nilai prediksi perlu terlebih dahulu dilakukan sebuah proses *denormalisasi* data, dan hasil dari *denormalisasi* data merupakan nilai prediksi sebenarnya. Proses terakhir dilakukan perhitungan evaluasi menggunakan MAPE untuk menghitung nilai *error* antara data aktual dan data prediksi dan menghasilkan keluaran nilai prediksi dan MAPE.

4.2 Alir Perancangan Algoritme

Pada bagian perancangan penelitian ini akan menggunakan penerapan dari algoritme *Extreme Learning Machine* (ELM) dalam prediksi harga cabai rawit di Kota Malang. Dalam perancangan algoritme, ELM memiliki beberapa tahapan proses dalam penyelesaian masalah prediksi yaitu bagian input *dataset*, persentase data *training*, persentase data *testing*, jenis fungsi aktivasi, banyak fitur, banyak *neuron*, nilai minimum, nilai maksimum, bagian *pre-processing* data, bagian proses normalisasi data, proses *training,* proses *testing*, proses *denormalisasi* data, dan proses MAPE. Berikut diagram alir yang digunakan sebagai gambaran proses dari algoritme ELM terdapat pada bagian Gambar 4.1.

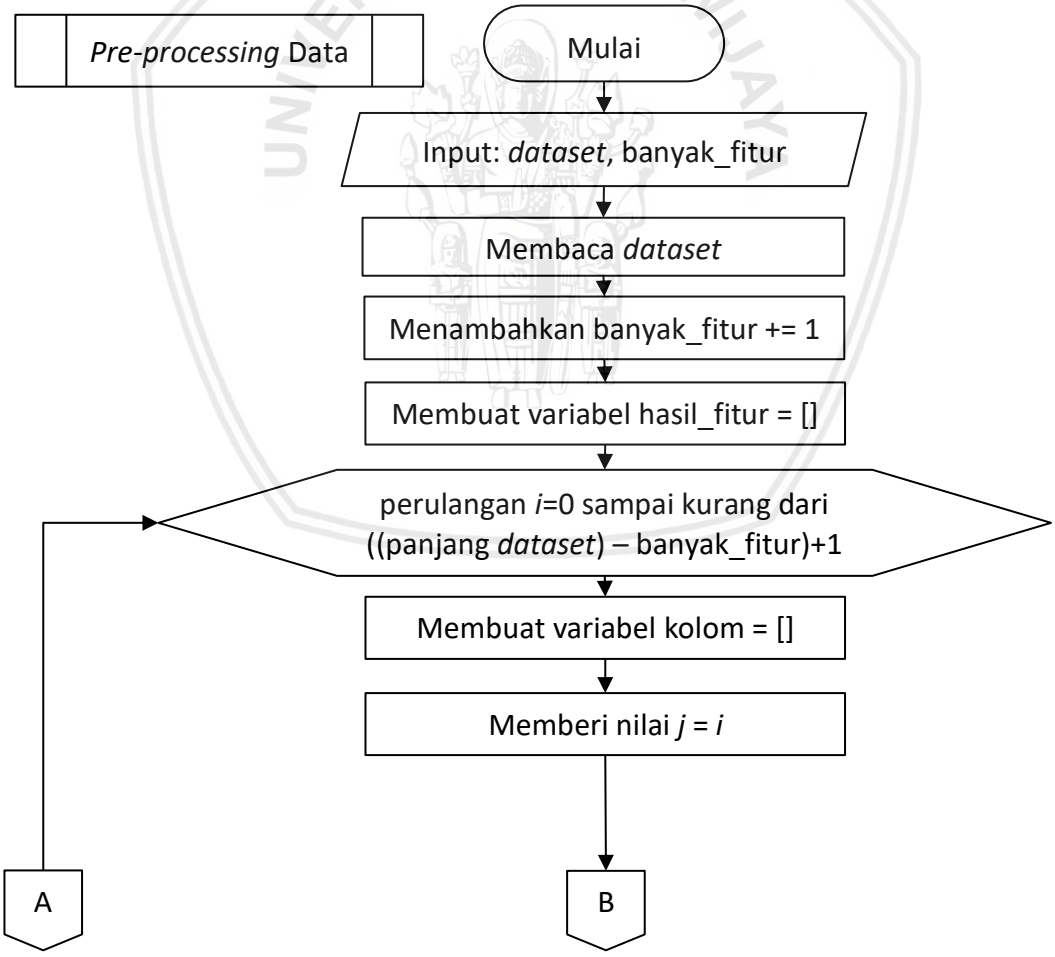


Gambar 4.1 Diagram Alir *Extreme Learning Machine*

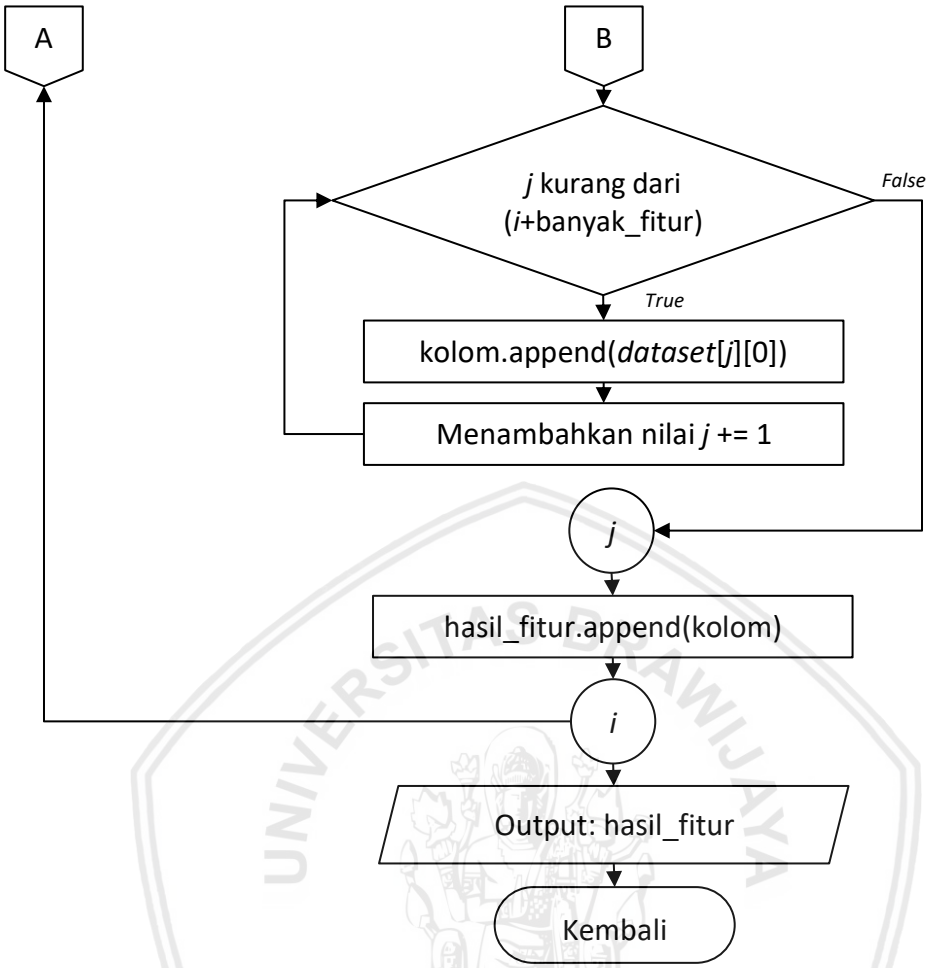
Dari diagram alir pada Gambar 4.1 dapat dijelaskan setiap langkah dari algoritme ELM sebagai berikut.

1. Memasukkan input dari sistem dalam bentuk *user interface* berupa nilai *dataset* untuk harga cabai rawit, persentase dari data *training* dan *testing*, banyak fitur, banyak *hidden neuron* fungsi aktivasi, nilai minimum dan maksimum.
2. Melakukan proses *pre-processing* data dengan metode analisis teknikal.
3. Melakukan proses normalisasi data dari yang sudah dimasukkan sebelumnya dengan rumus *Min-Max Normalization* sesuai dengan Persamaan 2.5.
4. Melakukan tahapan proses *training* untuk mencari nilai *output weight*.
5. Melakukan tahapan proses *testing* untuk menghasilkan nilai data prediksi dari hasil nilai *output weight* yang sudah diperoleh sebelumnya.
6. Melakukan proses *denormalisasi* data untuk mengubah nilai prediksi ke dalam nilai yang sebenarnya.
7. Melakukan proses perhitungan nilai MAPE dari hasil prediksi yang sudah di *denormalisasi* dengan mencari nilai *error* data prediksi dan data aktual.
8. Menghasilkan keluaran *user interface* berupa hasil prediksi dari hasil nilai *error* MAPE.

4.2.1 Alir *Pre-processing* Data



Gambar 4.2 Diagram Alir *Pre-processing* Data

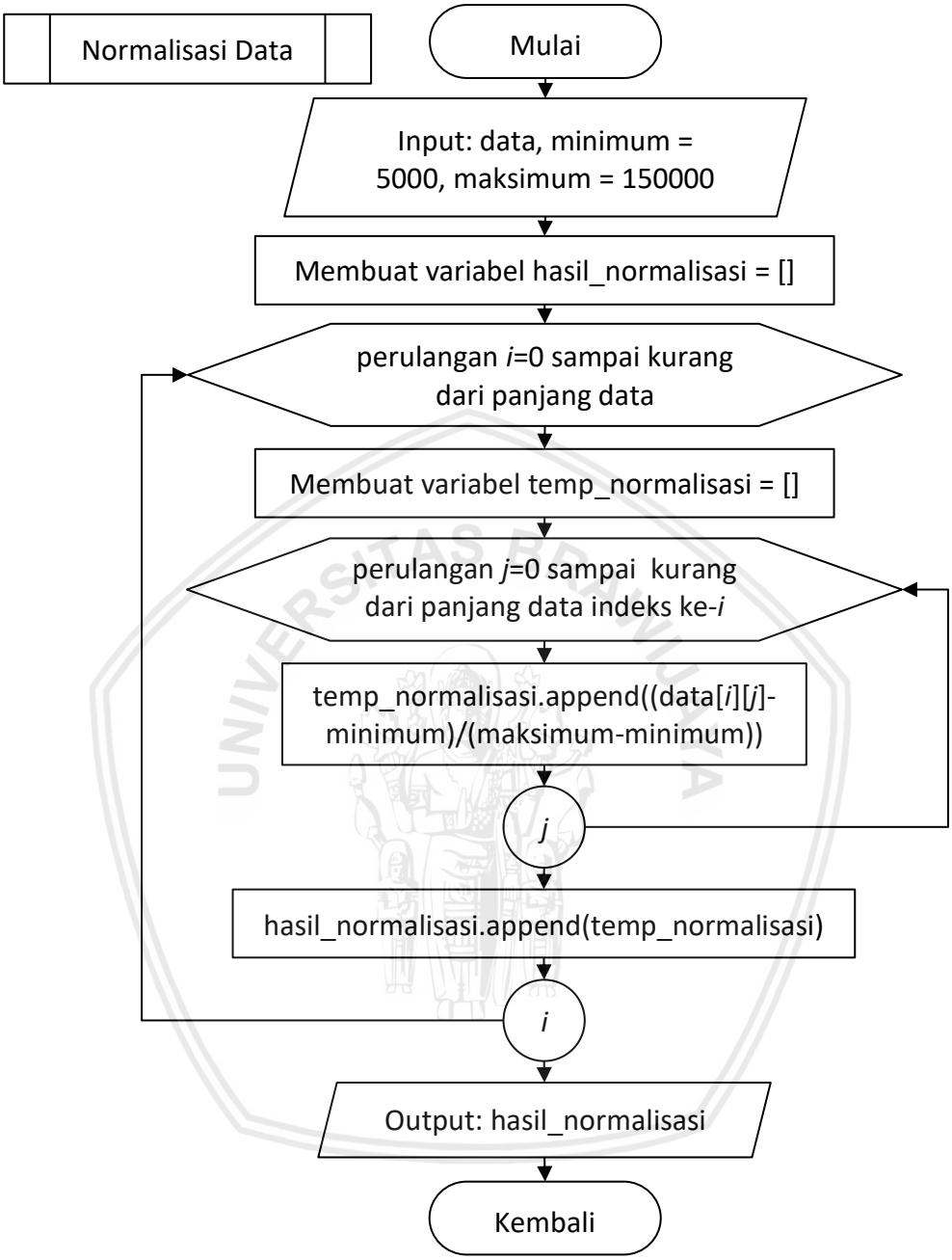


Gambar 4.2 Diagram Alir *Pre-processing* Data (Lanjutan)

Dari diagram alir pada bagian Gambar 4.2 dapat dijelaskan setiap langkah dari proses *pre-processing* data sebagai berikut.

1. Memasukkan input *file dataset* untuk harga cabai rawit dan banyak_fitur.
2. Melakukan proses membaca *dataset* dengan menggunakan library pandas untuk membaca *file* dengan format *Comma Separated Values* (CSV).
3. Melakukan penambahan nilai banyak_fitur dengan nilai 1.
4. Membuat variabel list kosong dengan nama hasil_fitur.
5. Melakukan perulangan variabel *i*=0 sampai kurang dari ((panjang *dataset*)-banyak_fitur)+1.
6. Membuat variabel list kosong dengan nama kolom.
7. Memberikan nilai inisialisasi untuk nilai *j* sama dengan nilai *i*.
8. Melakukan perulangan sampai nilai *j* kurang dari jumlah (*i*+banyak fitur).
9. Memasukkan nilai *dataset* baris ke-*j* dan kolom ke-0 ke dalam variabel *kolom*.
10. Melakukan penambahan pada nilai *j* dengan nilai 1.
11. Memasukkan nilai hasil yang telah disimpan pada variabel kolom ke dalam variabel hasil_fitur.
12. Menghasilkan keluaran berupa nilai hasil_fitur.

4.2.2 Alir Normalisasi Data



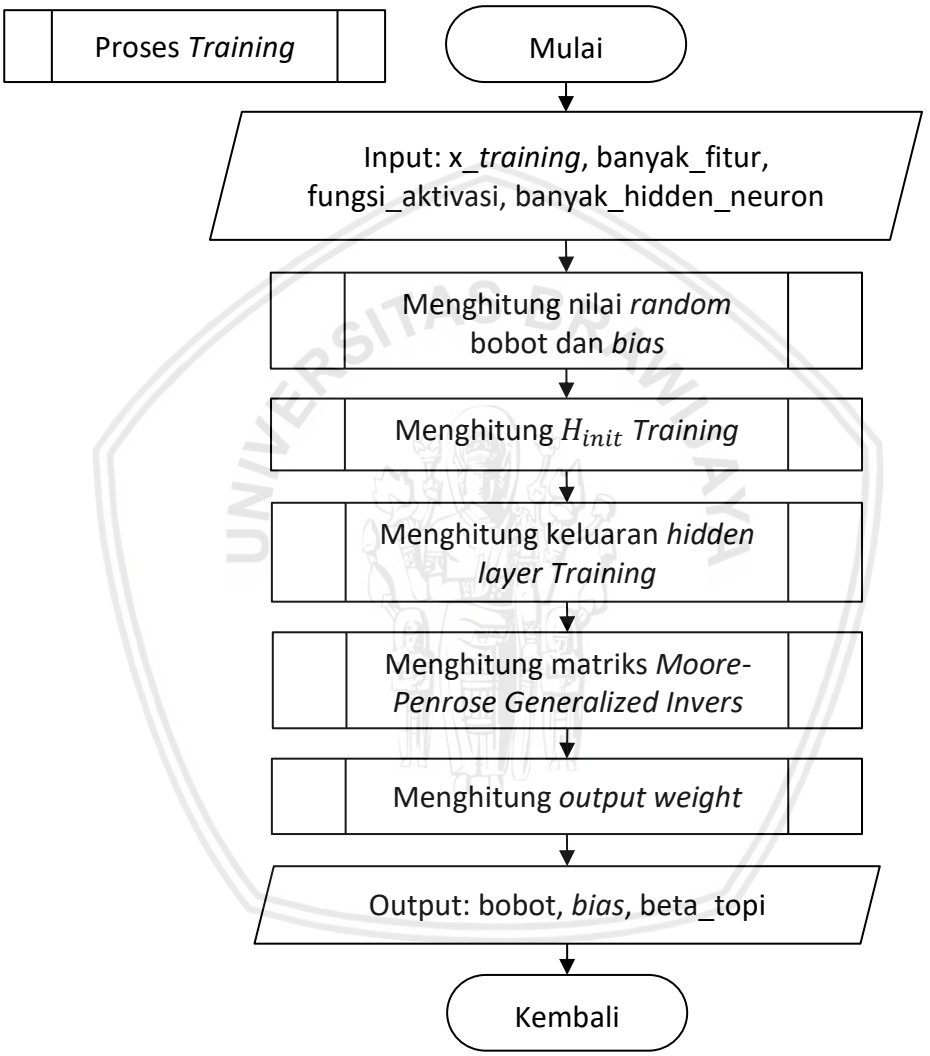
Gambar 4.3 Diagram Alir Normalisasi Data

Dari diagram alir pada bagian Gambar 4.3 dapat dijelaskan setiap langkah dari proses normalisasi data sebagai berikut.

1. Memasukkan input data dari harga cabai rawit, variabel minimum untuk menyimpan nilai minimum sebesar Rp5000,00, dan variabel maksimum untuk menyimpan nilai maksimum sebesar Rp150.000,00.
2. Membuat variabel list kosong dengan nama hasil_normalisasi.
3. Melakukan perulangan variabel *i* sebanyak 0 sampai kurang dari panjang baris data.

4. Membuat variabel list kosong dengan nama temp_normalisasi.
5. Melakukan perulangan variabel *j* sebanyak 0 sampai kurang dari panjang *kolom* data pada setiap indeks ke-*i*.
6. Melakukan perhitungan dari rumus *Min-Max Normalization* sesuai dengan Persamaan 2.5 pada *data* indeks baris ke-*i* dan kolom ke-*j*.
7. Menghasilkan keluaran data setelah proses normalisasi.

4.2.3 Alir Proses *Training*



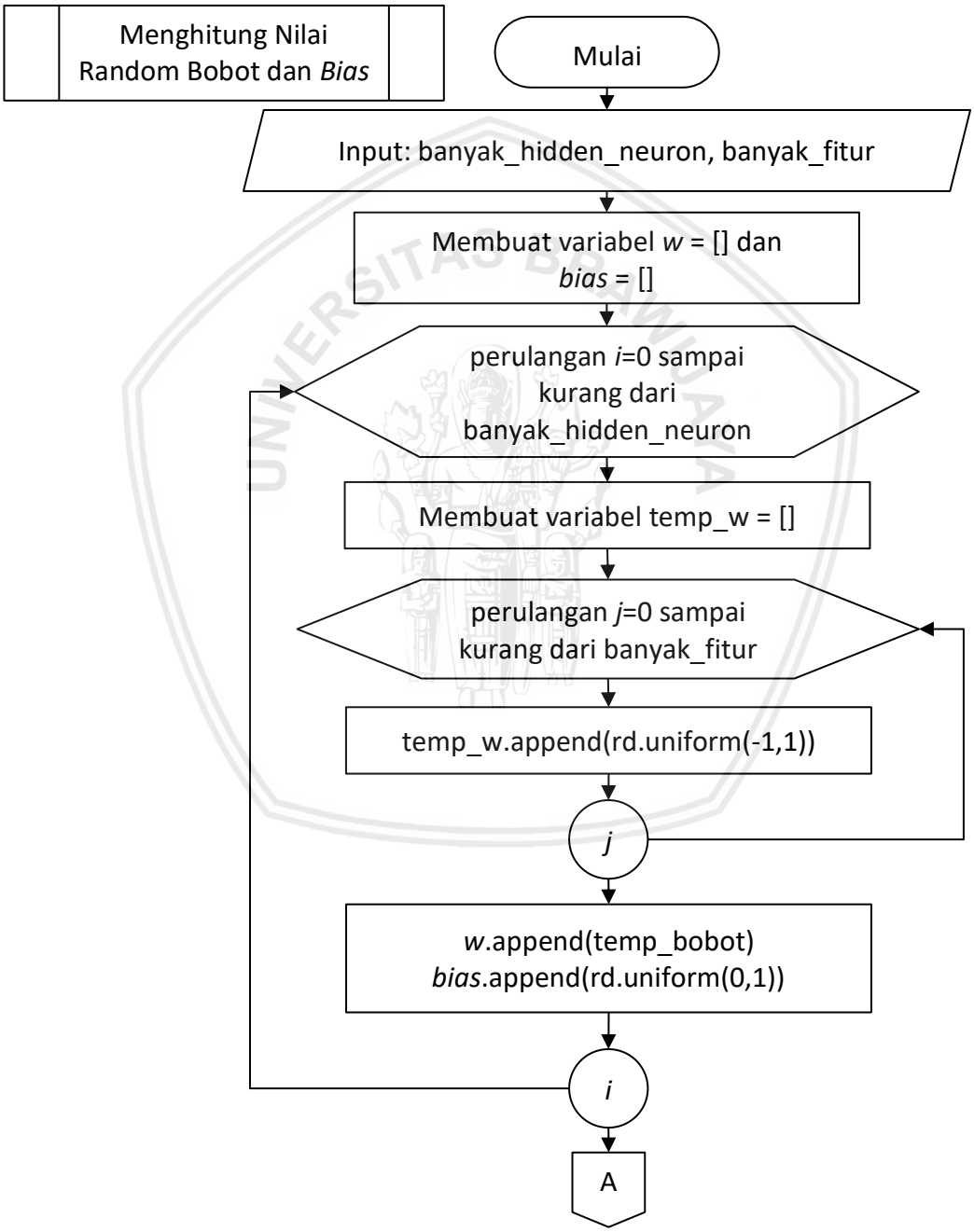
Gambar 4.4 Diagram Alir Proses *Training*

Dari diagram alir pada bagian Gambar 4.4 dapat dijelaskan setiap langkah dari proses *training* sebagai berikut.

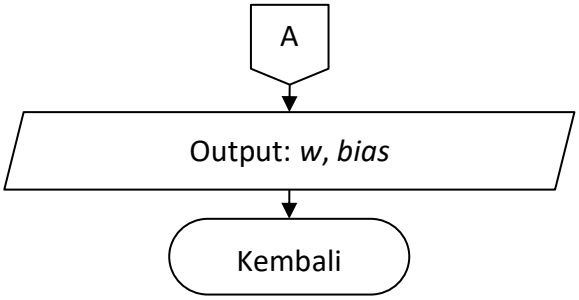
1. Memasukkan input x_training dari hasil data *training* yang telah di normalisasi, banyak_fitur, banyak_hidden_neuron dan fungsi_aktivasi.
2. Melakukan proses menghitung nilai bobot dan *bias* yang diperoleh melalui cara *random* dengan rentang bobot [-1,1] dan rentang *bias* [0,1].
3. Melakukan proses menghitung nilai H_{init} dari input data_training, bobot, dan *bias*.

4. Melakukan proses menghitung pada bagian proses keluaran *hidden layer* atau nilai *H* berdasarkan jenis dari input fungsi_aktivasi.
5. Melakukan proses menghitung pada bagian proses matriks *Moore-Penrose Generalized Inverse* dengan cara inversi Operasi Baris Elementer (OBE).
6. Melakukan proses menghitung bagian *output weight* pada proses *training*.
7. Menghasilkan keluaran nilai bobot, *bias*, dan hasil dari beta_topi (*output weight*) yang dijadikan input pada tahap selanjutnya di proses *testing*.

4.2.3.1 Proses Menghitung Nilai *Random* Bobot dan *Bias*



Gambar 4.5 Diagram Alir Menghitung Nilai *Random* Bobot dan *Bias*

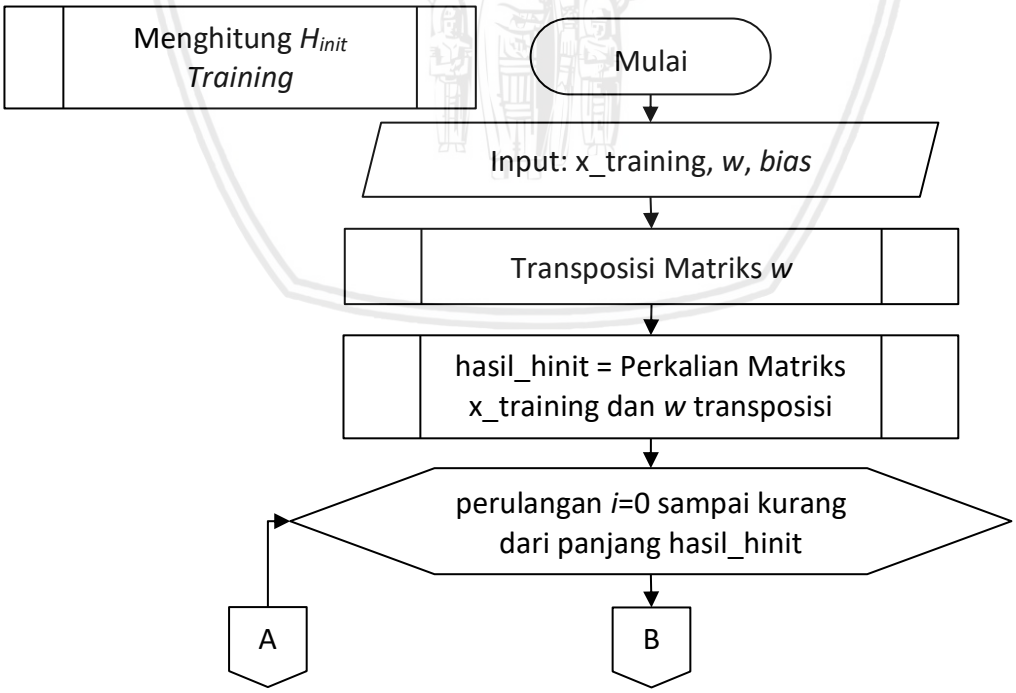


Gambar 4.5 Diagram Alir Menghitung Nilai Random Bobot dan *Bias* (Lanjutan)

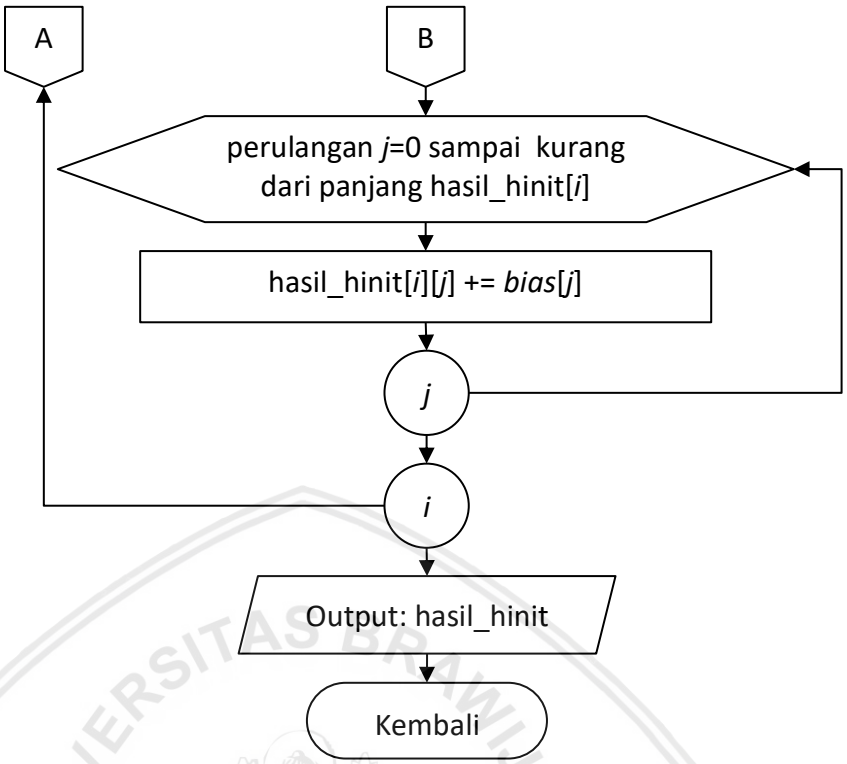
Dari diagram alir pada bagian Gambar 4.5 dapat dijelaskan setiap langkah dari proses menghitung nilai random untuk nilai bobot dan *bias* sebagai berikut.

1. Memasukkan input variabel banyak_hidden_neuron dan banyak_fitur.
2. Melakukan bagian perulangan sebanyak *i* dengan nilai *i*=0 sampai kurang dari nilai banyak_hidden_neuron.
3. Melakukan perulangan sebanyak *j* dengan nilai *j*=0 sampai kurang dari nilai banyak_fitur.
4. Melakukan pembuatan nilai random dengan rentang nilai [-1,1] kemudian disimpan pada variabel temp_w.
5. Melakukan pembuatan nilai *bias* dengan rentang [0,1] kemudian disimpan pada variabel *bias* dan menyimpan nilai temp_w ke dalam variabel *w*.
6. Menghasilkan keluaran nilai matriks *w* dan *bias*.

4.2.3.2 Proses Menghitung H_{init} *Training*



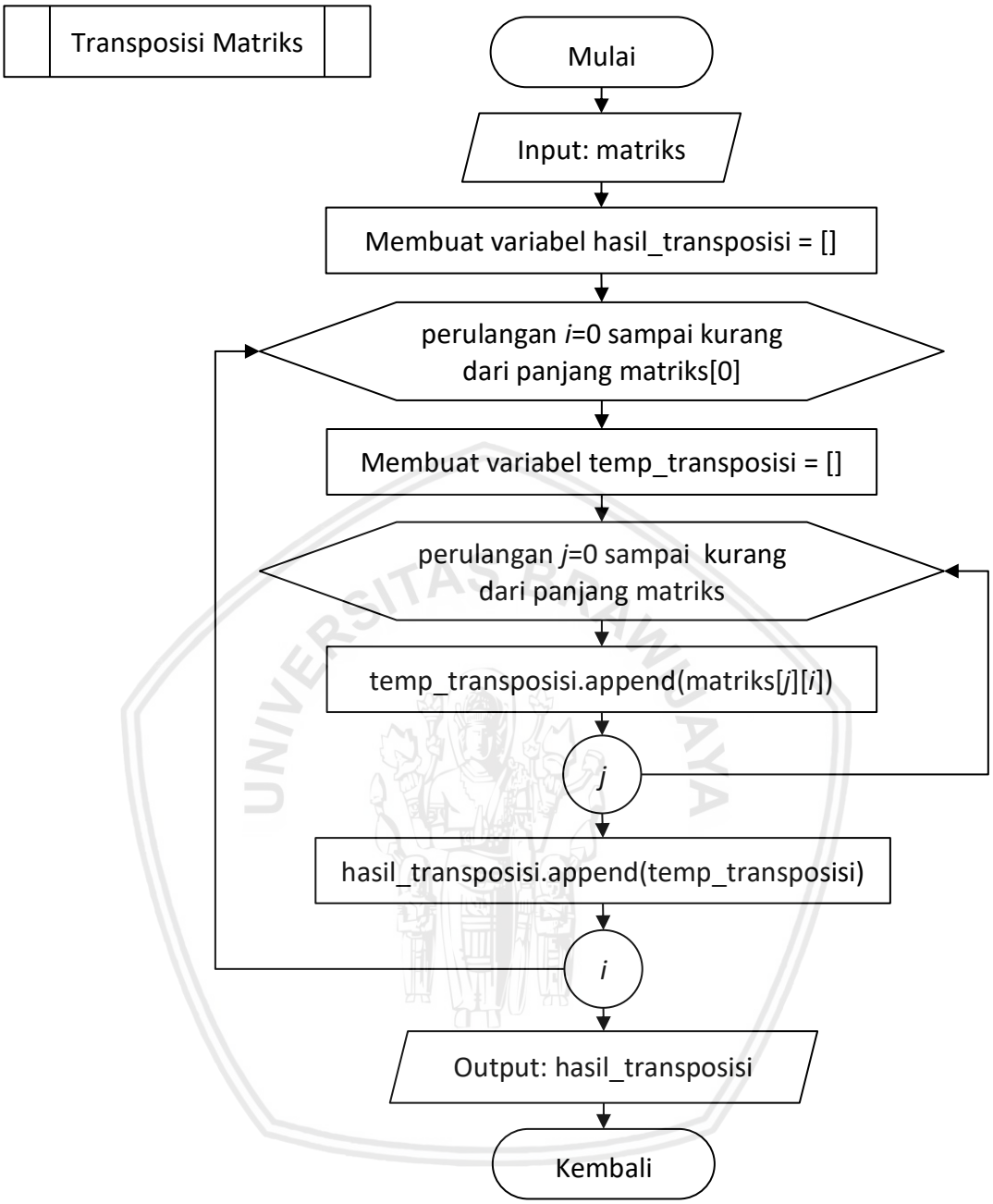
Gambar 4.6 Diagram Alir Menghitung H_{init} *Training*



Gambar 4.6 Diagram Alir Menghitung H_{init} *Training* (Lanjutan)

Dari diagram alir pada bagian Gambar 4.6 dapat dijelaskan setiap langkah dari proses H_{init} *training* sebagai berikut.

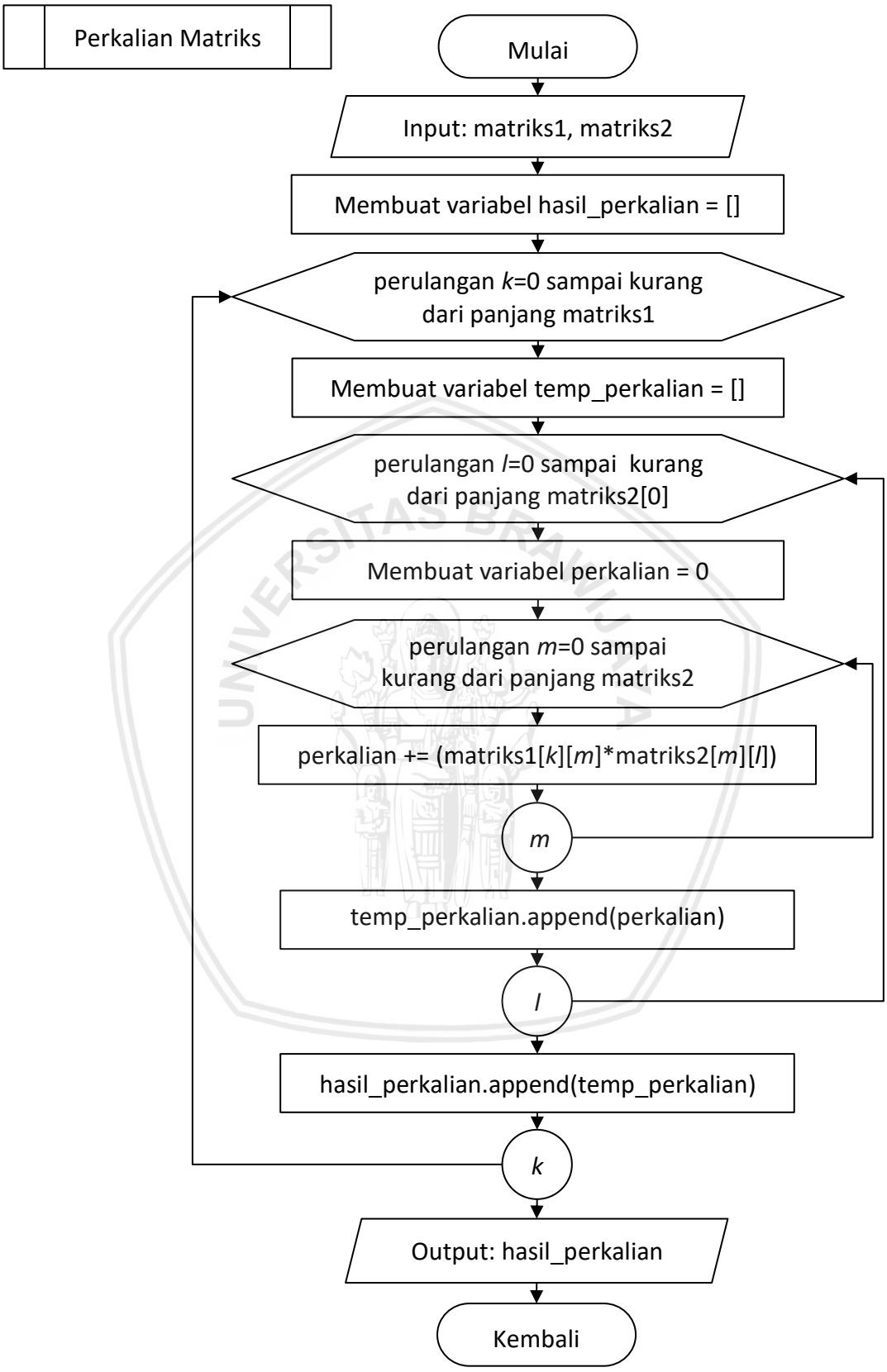
1. Memasukkan input nilai dari data x_training, nilai *w* sebagai bobot, dan masukan nilai *bias*.
2. Melakukan proses transposisi pada nilai *w* dan menghasilkan nilai hasil tranposisi *w*.
3. Melakukan proses perkalian matriks antara x_training dengan nilai hasil tranposisi *w* kemudian disimpan pada variabel hasil_hinit.
4. Melakukan perulangan sebanyak *i* dimulai dari *i*=0 sampai kurang dari panjang hasil_hinit.
5. Melakukan perulangan sebanyak *j* dimulai dari *j*=0 sampai kurang dari panjang hasil_hinit pada setiap indeks ke-*i*.
6. Melakukan perhitungan H_{init} *Training* dengan menambahkan nilai hasil_hinit baris ke-*i* dan kolom ke-*j* dengan nilai *bias* indeks ke-*j*.
7. Menghasilkan keluaran berupa nilai hasil_hinit.



Gambar 4.7 Diagram Alir Transposisi Matriks

Dari diagram alir pada bagian Gambar 4.7 dapat dijelaskan setiap langkah dari proses transposisi matriks sebagai berikut.

1. Memasukkan input nilai matriks sebelum dilakukan transposisi.
2. Membuat variabel list kosong dengan nama hasil_transposisi.
3. Melakukan perulangan sebanyak *i* dimulai dari *i*=0 sampai kurang dari panjang matriks[0], 0 menandakan baris matriks.
4. Melakukan perulangan dari nilai *j*=0 sampai kurang dari panjang matriks.
5. Menyimpan nilai matriks[*j*][*i*] ke dalam variable temp_transposisi.
6. Menyimpan nilai temp_transposisi ke dalam variable hasil_transposisi.
7. Menghasilkan keluaran nilai berupa matriks nilai hasil_transposisi.

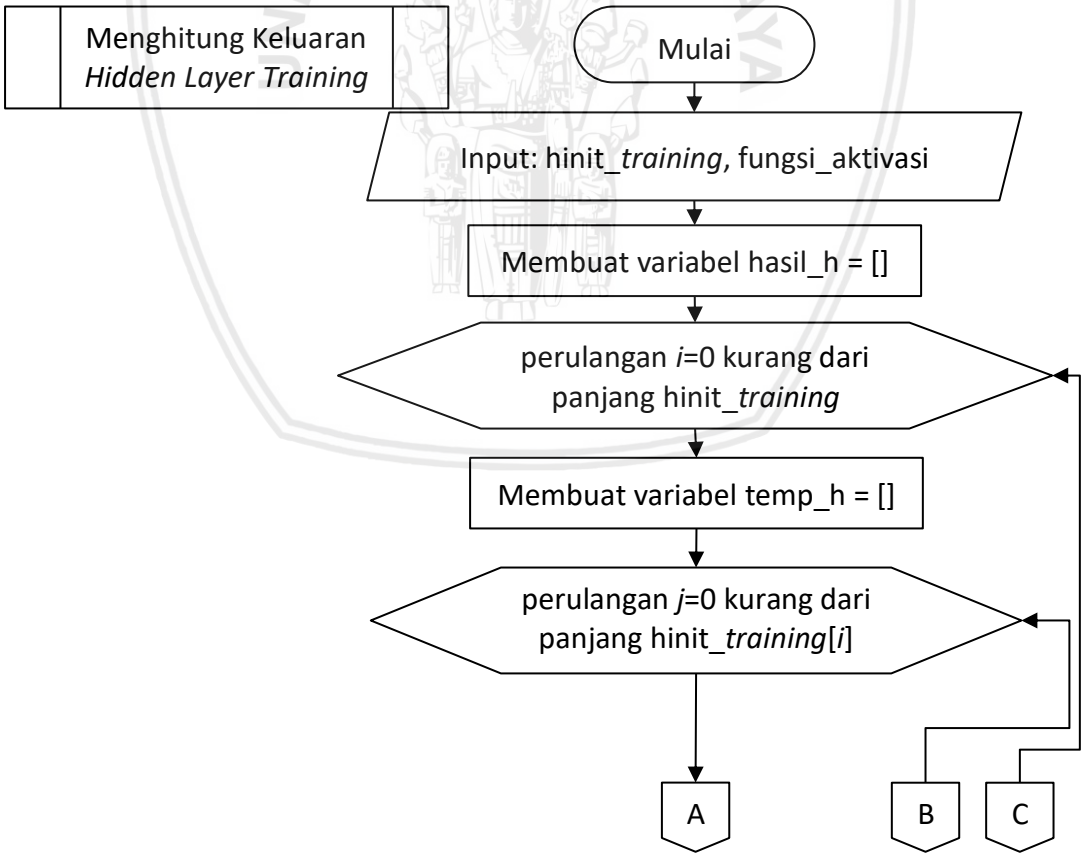


Gambar 4.8 Diagram Alir Perkalian Matriks

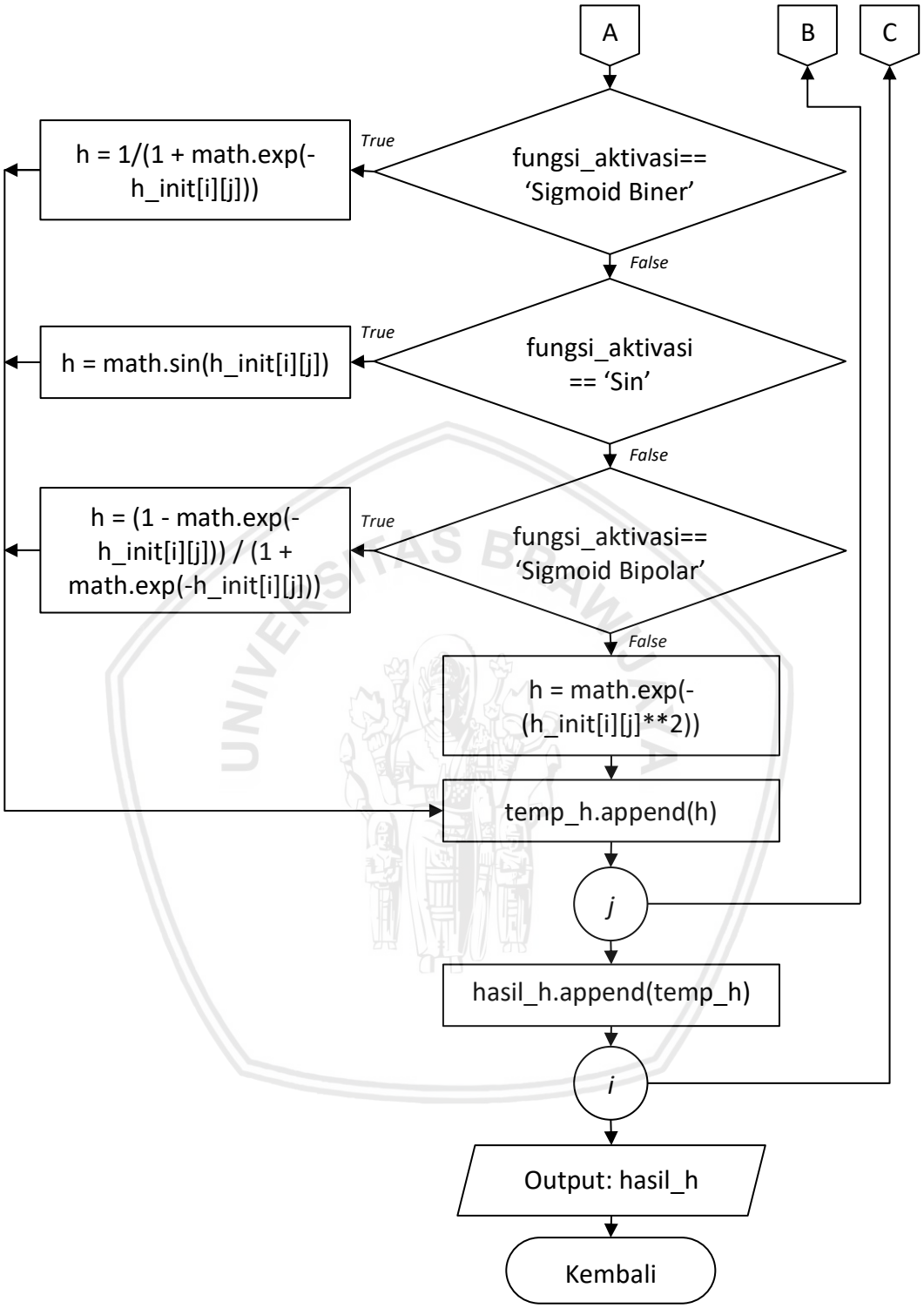
Dari diagram alir pada bagian Gambar 4.8 dapat dijelaskan setiap langkah dari proses perkalian matriks sebagai berikut.

1. Memasukkan input variabel nilai matriks1 dan nilai matriks2.
2. Membuat variabel list kosong dengan nama hasil_perkalian.
3. Melakukan perulangan sebanyak *k* dimulai dari *k*=0 sampai kurang dari panjang matriks1.
4. Membuat variabel list kosong dengan nama temp_perkalian.
5. Melakukan perulangan sebanyak *l* dimulai dari *l*=0 sampai kurang dari panjang matriks2[0], 0 menandakan baris matriks2.
6. Membuat variabel dengan nama perkalian yang bernilai 0.
7. Melakukan perulangan sebanyak *m* dimulai dari *m*=0 sampai kurang dari panjang matriks2.
8. Melakukan perkalian antara nilai matriks1 baris ke-*k* kolom ke-*m* dengan nilai matriks2 baris ke-*m* kolom ke-*l* dan kemudian menyimpan nilai hasil perkalian ke dalam variabel perkalian.
9. Menyimpan nilai variabel perkalian ke dalam variable temp_perkalian.
10. Menyimpan nilai temp_perkalian ke dalam variable hasil_perkalian.
11. Menghasilkan keluaran nilai berupa matriks nilai hasil_perkalian.

4.2.3.3 Proses Menghitung Keluaran *Hidden Layer Training*



Gambar 4.9 Diagram Alir Menghitung Keluaran *Hidden Layer Training*



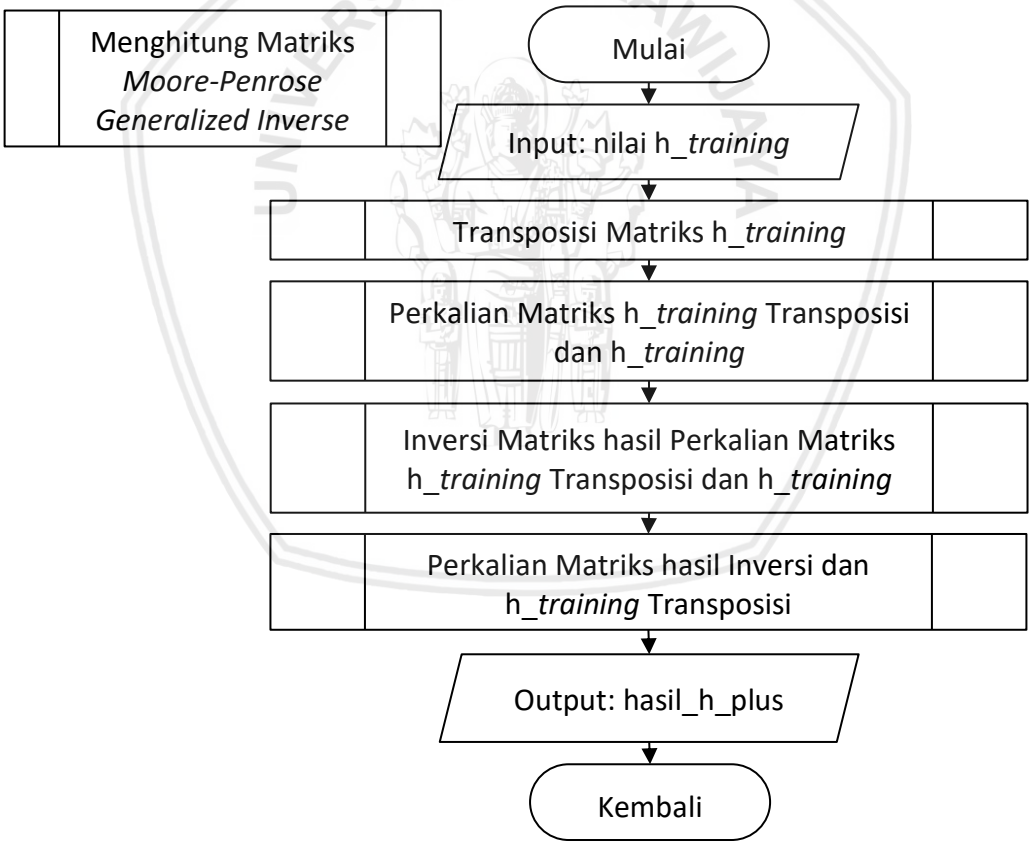
Gambar 4.9 Diagram Alir Menghitung Keluaran *Hidden Layer Training* (Lanjutan)

Dari diagram alir pada bagian Gambar 4.9 dapat dijelaskan setiap langkah proses keluaran *hidden layer* di bagian proses *training* sebagai berikut.

1. Memasukkan nilai input hinit_training dan fungsi_aktivasi.
2. Membuat variabel list kosong dengan nama hasil_h.

3. Melakukan perulangan sebanyak *i* sampai kurang dari panjang hinit_training.
4. Membuat variable list kosong dengan nama temp_h.
5. Melakukan perulangan sebanyak *j* dimulai *j*=0 sampai kurang dari panjang dari hinit_training indeks ke-*i*.
6. Melakukan pengecekan kondisi jika fungsi aktivasi sama dengan "*Sigmoid Biner*" maka menjalankan proses perhitungan rumus fungsi aktivasi *Sigmoid Biner* dan jika salah cek kondisi selanjutnya.
7. Melakukan pengecekan kondisi jika fungsi aktivasi sama dengan "Sin" maka proses perhitungan rumus fungsi *Sin* dan jika salah cek kondisi selanjutnya.
8. Melakukan pengecekan kondisi jika fungsi aktivasi sama dengan "*Sigmoid Bipolar*" maka menjalankan perhitungan rumus fungsi aktivasi *Sigmoid Bipolar*, jika salah menjalankan perhitungan fungsi aktivasi *Radial Basis*.
9. Menyimpan nilai hasil dari nilai h ke dalam variabel temp_h.
10. Menyimpan nilai pada variabel temp_h ke dalam hasil_h.
11. Menghasilkan keluaran dengan nilai hasil_h pada proses *training*.

4.2.3.4 Alir Menghitung Matriks *Moore-Penrose Generalized Inverse*

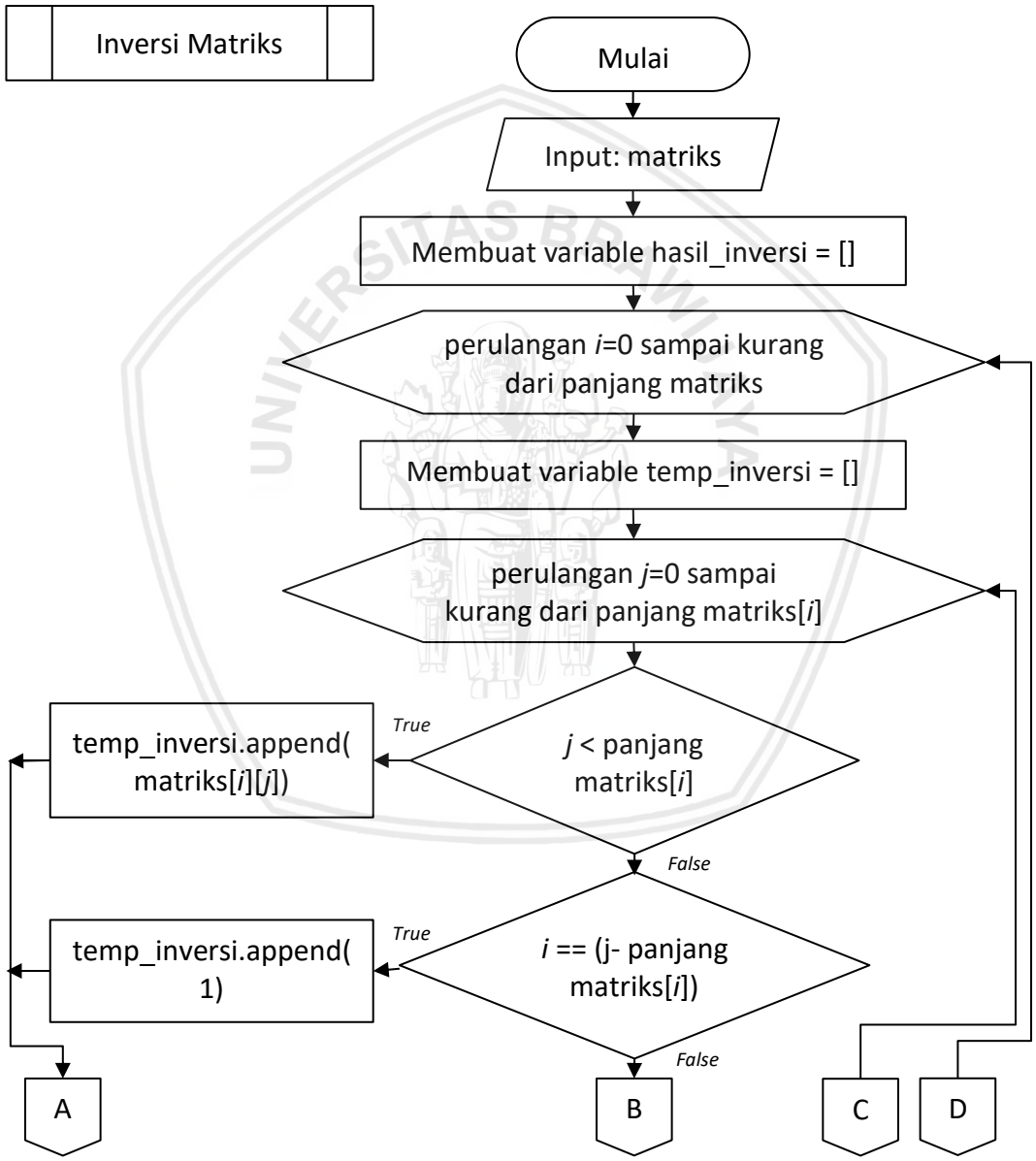


Gambar 4.10 Diagram Alir Menghitung Matriks *Moore-Penrose Generalized Inverse*

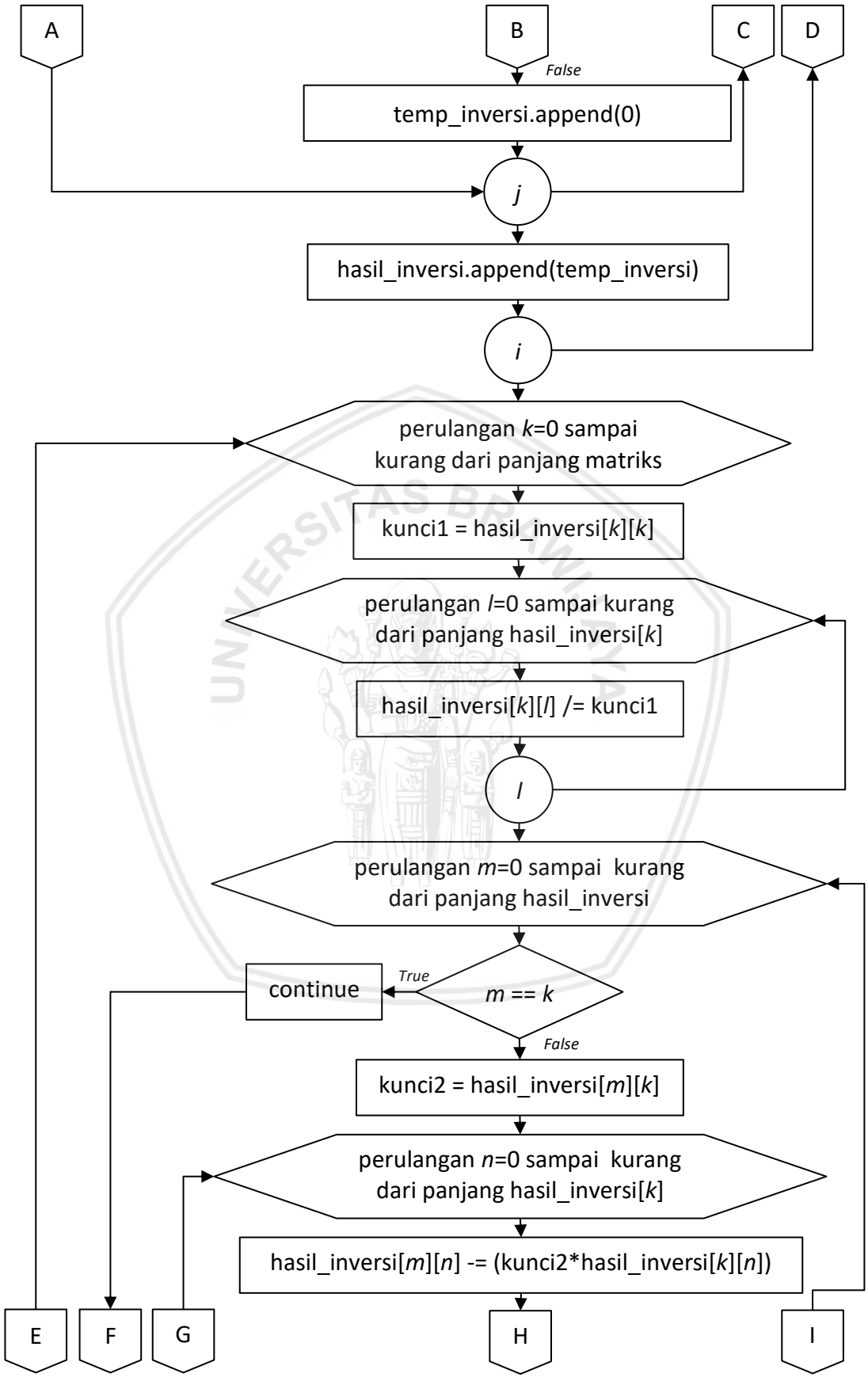
Dari diagram alir pada bagian Gambar 4.10 dijelaskan setiap langkah dari proses menghitung nilai matriks *Moore-Penrose Generalized Inverse* sebagai berikut.

1. Memasukkan input nilai h_training.

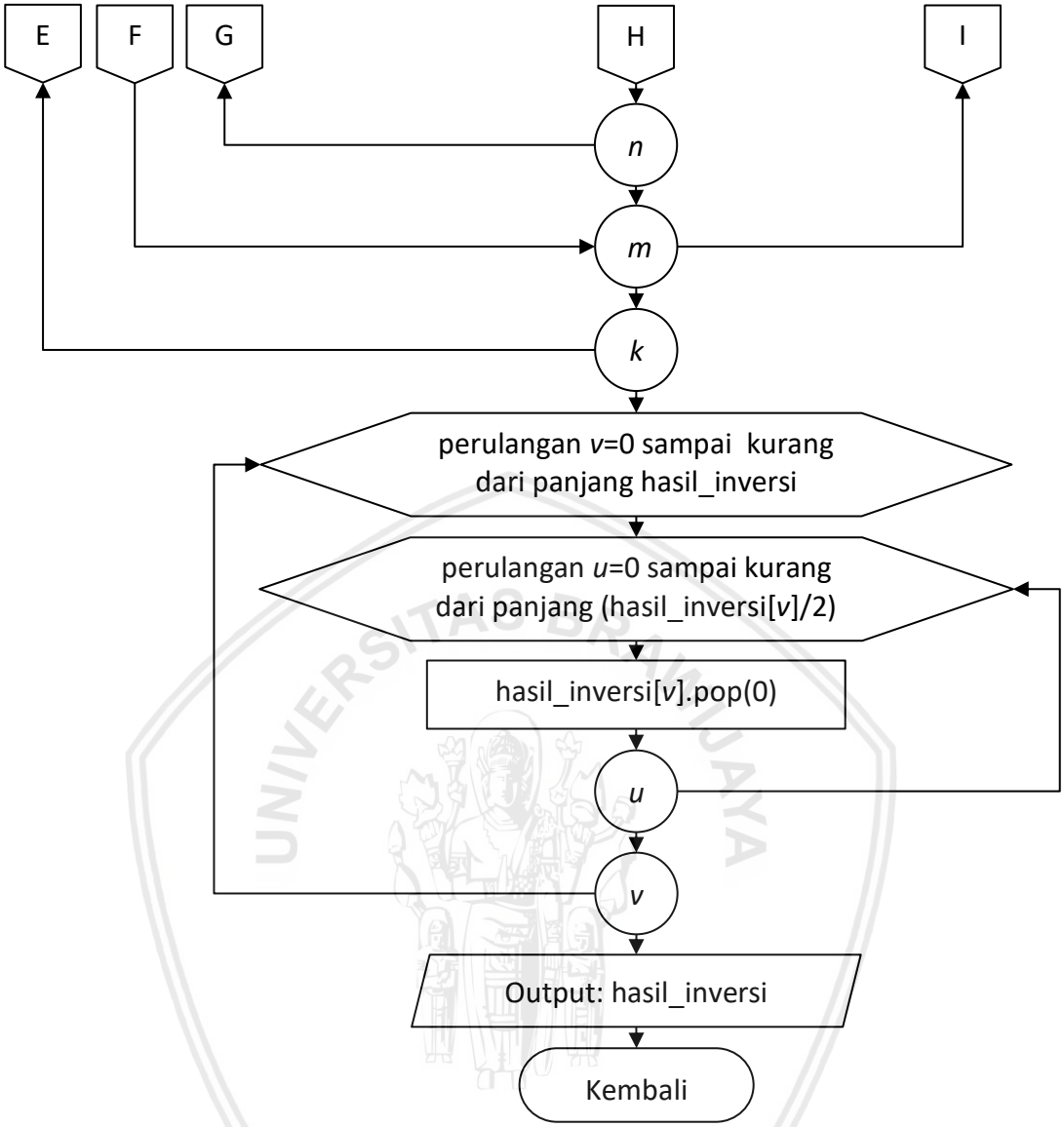
2. Melakukan proses perhitungan transposisi dari nilai h_training.
3. Melakukan bagian pada proses perhitungan perkalian untuk matriks h_training transposisi dengan nilai matriks pada h_training.
4. Melakukan proses perhitungan inversi matriks dari hasil antara perkalian dari matriks H transposisi dengan nilai matriks h_training.
5. Melakukan proses perhitungan perkalian matriks dari hasil nilai inversi dengan nilai matriks h_training transposisi.
6. Menghasilkan keluaran nilai H^+ pada variable hasil_h_plus.



Gambar 4.11 Diagram Alir Inversi Matriks



Gambar 4.11 Diagram Alir Inversi Matriks (Lanjutan)



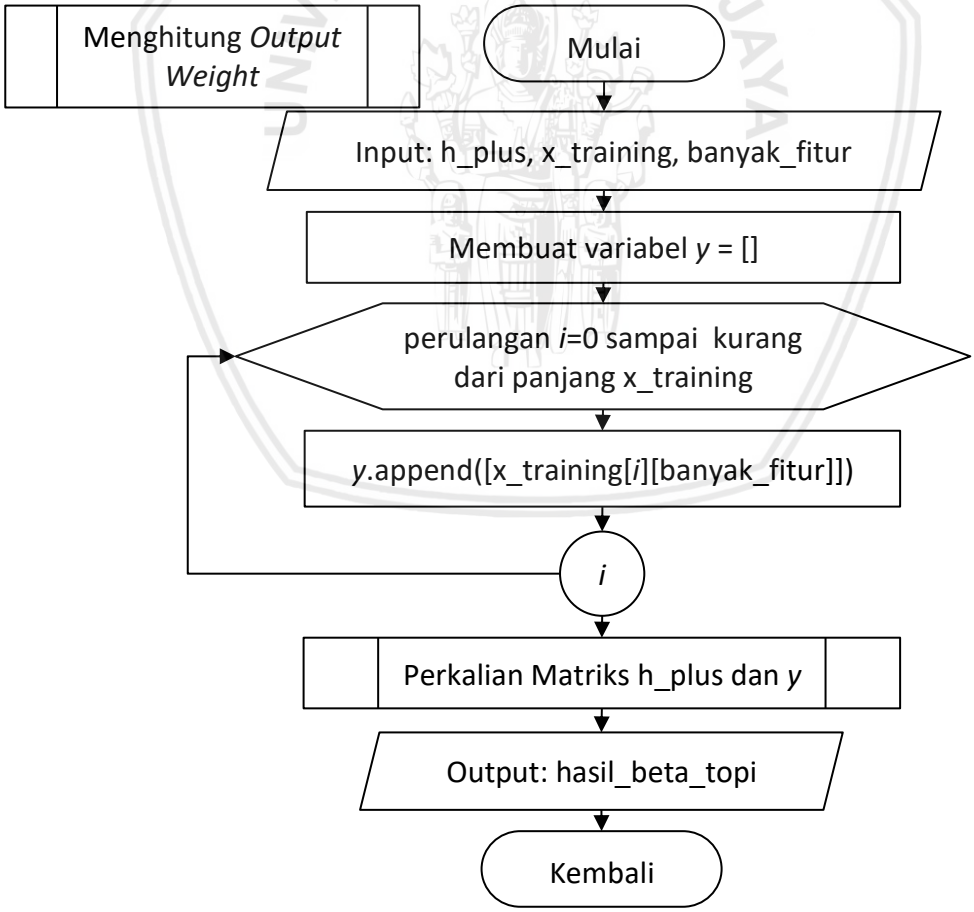
Gambar 4.11 Diagram Alir Inversi Matriks (Lanjutan)

Dari diagram alir pada bagian Gambar 4.11 dijelaskan setiap langkah dari proses menghitung nilai inversi matriks sebagai berikut.

1. Memasukkan input nilai variabel matriks sebelum dilakukan inversi.
2. Membuat variabel list kosong dengan nama hasil_inversi.
3. Melakukan perulangan sebanyak *i*=0 sampai kurang dari panjang matriks.
4. Membuat variabel list kosong dengan nama temp_inversi.
5. Melakukan perulangan sebanyak *j*=0 sampai kurang dari panjang matriks[*i*].
6. Melakukan seleksi kondisi nilai *j* == panjang matriks[*i*], jika bernilai benar memasukkan nilai matriks[*i*][*j*] ke dalam variabel temp_inversi dan jika bernilai salah melakukan seleksi berikutnya.
7. Melakukan seleksi kondisi nilai *i* == (*j*-panjang matriks[*i*]), jika bernilai benar memasukkan nilai 1 ke dalam variabel temp_inversi dan jika bernilai salah masukkan nilai 0 ke variabel temp_inversi.
8. Menyimpan nilai temp_inversi ke dalam variabel hasil_inversi.

9. Melakukan perulangan dari *k*=0 sampai kurang dari panjang dari matriks.
10. Menyimpan nilai matriks baris ke-*k* dan kolom ke-*k* ke dalam variabel kunci1.
11. Melakukan perulangan *l*=0 sampai kurang dari panjang hasil_inversi ke-*k*.
12. Melakukan perhitungan dengan operator penugasan pembagian pada nilai hasil_inversi baris ke-*k* dan kolom ke-*l* dibagi dengan nilai kunci1.
13. Melakukan perulangan dari *m*=0 sampai kurang dari panjang variabel matriks.
14. Melakukan seleksi kondisi nilai *m* == *k*, jika bernilai benar maka menjalankan perintah *continue* dan jika bernilai salah memasukkan nilai variabel matriks baris ke-*m* dan kolom ke-*k* ke dalam variabel kunci2.
15. Menyimpan hasil_inversi baris ke-*m* dan kolom ke-*k* dalam variabel kunci2.
16. Melakukan perulangan dari *n*=0 sampai kurang dari panjang hasil_inversi[*k*].
17. Melakukan operator penugasan pengurangan nilai hasil_inversi baris ke-*m* dan kolom ke-*n* dikurang kunci2 dikali *matriks* baris ke-*k* dan kolom ke-*n*.
18. Melakukan perulangan dimulai *v*=0 sampai kurang dari panjang hasil_inversi.
19. Melakukan perulangan *u*=0 sampai kurang dari panjang hasil_inversi[v]/2.
20. Memilih matriks baris ke-*v* untuk dihapus dari *list* matriks pada indeks ke-*0*.
21. Menghasilkan keluaran matriks hasil_inversi.

4.2.3.5 Alir Menghitung *Output Weight*

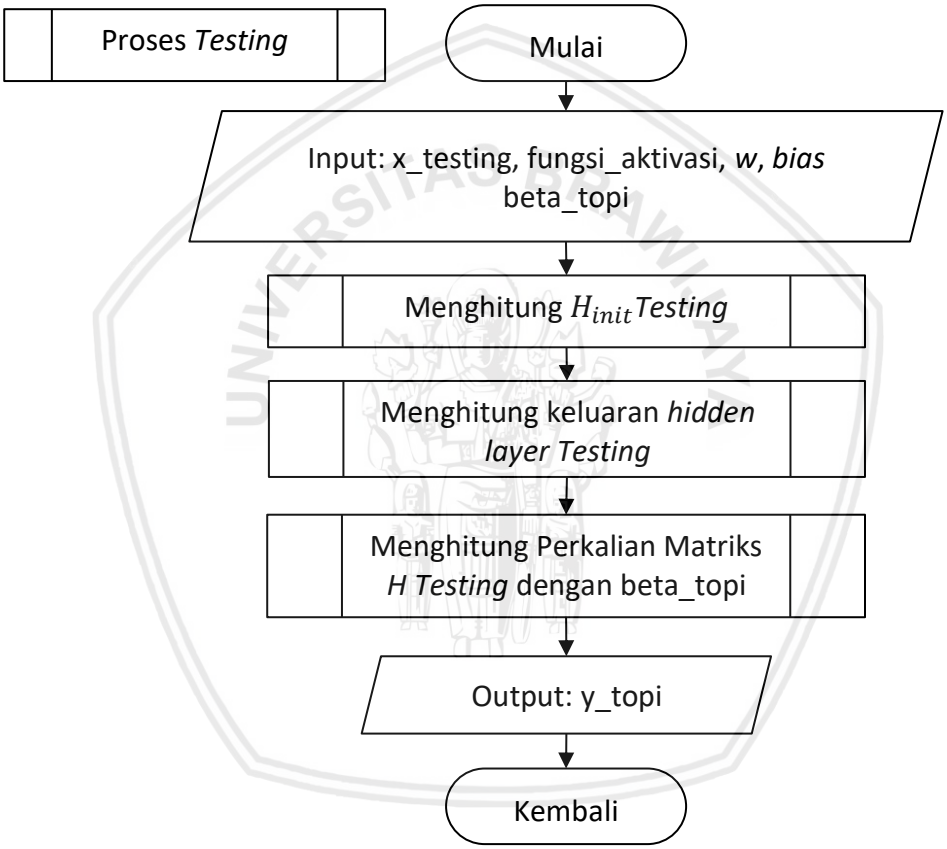


Gambar 4.12 Diagram Alir Menghitung *Output Weight*

Dari diagram alir pada bagian Gambar 4.12 dapat dijelaskan setiap langkah dari proses perhitungan nilai *output weight* sebagai berikut.

1. Memasukkan input h_plus, x_training, dan banyak_fitur.
2. Membuat variabel *y* dengan nilai *list* kosong.
3. Melakukan perulangan sebanyak *i* dimulai dari *i*=0 sampai kurang dari panjang dari x_training.
4. Menyimpan nilai data aktual ke dalam variabel *y*.
5. Melakukan proses perhitungan perkalian matriks h_plus dengan matriks *y*.
6. Menghasilkan keluaran nilai $\hat{\beta}$ (*beta topi*) dalam variabel hasil_beta_topi.

4.2.4 Alir Proses *Testing*



Gambar 4.13 Diagram Alir Proses *Testing*

Dari diagram alir pada bagian Gambar 4.13 dapat dijelaskan setiap langkah dari proses *testing* sebagai berikut.

1. Memasukkan input *x_testing*, nilai untuk bobot (*w*), nilai untuk *bias*, jenis *fungsi_aktivasi*, dan nilai beta_topi.
2. Melakukan proses menghitung nilai H_{init} dari input x_testing, dan nilai bobot serta nilai dari *bias* yang telah diperoleh dari proses *training*.
3. Melakukan proses menghitung keluaran *hidden layer testing* berdasarkan dari jenis input fungsi_aktivasi.
4. Melakukan proses perkalian antara nilai *H Testing* dengan menggunakan nilai beta_topi dari proses *training*.

5. Menghasilkan keluaran nilai y_topi sebagai nilai hasil prediksi.

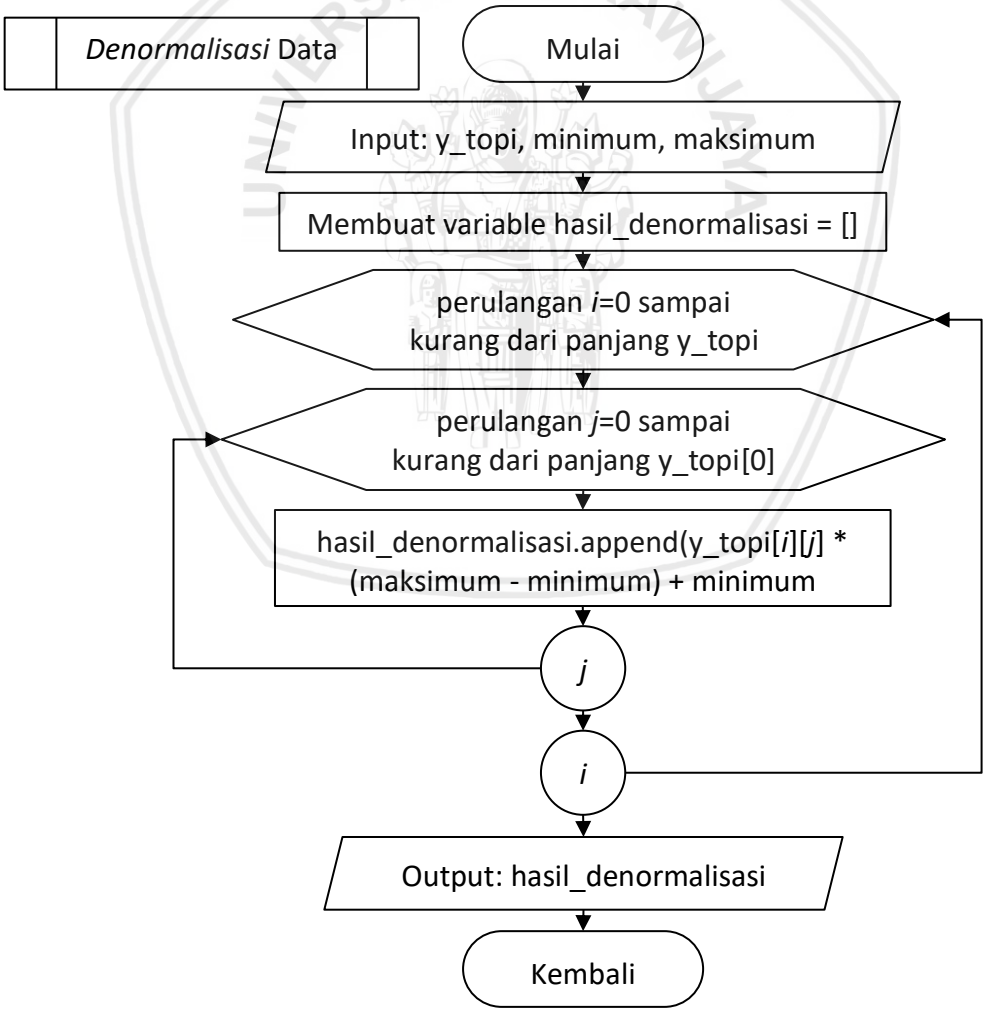
4.2.4.1 Menghitung H_{init} *Testing*

Pada perhitungan H_{init} *testing* dilakukan proses perkalian antara nilai bobot, *bias*, dan data *testing*. Perancangan diagram alir untuk H_{init} *testing* memiliki kesamaan dengan diagram alir H_{init} *training,* hanya terdapat perbedaan nilai parameter input yang digunakan diganti dengan nilai input data *testing*. Diagram alir H_{init} *testing* dapat dilihat pada bagian Gambar 4.6.

4.2.4.2 Menghitung Keluaran *Hidden Layer Testing*

Pada perhitungan nilai *H testing* dilakukan perhitungan menggunakan rumus fungsi aktivasi. Perancangan pada *H testing* memiliki kesamaan dengan diagram alir *H testing* hanya terdapat perbedaan pada pemberian nilai parameter input yang digunakan diganti dengan nilai H_{init} *testing*. Diagram alir *H testing* terdapat pada bagian Gambar 4.9.

4.2.5 Alir *Denormalisasi* Data

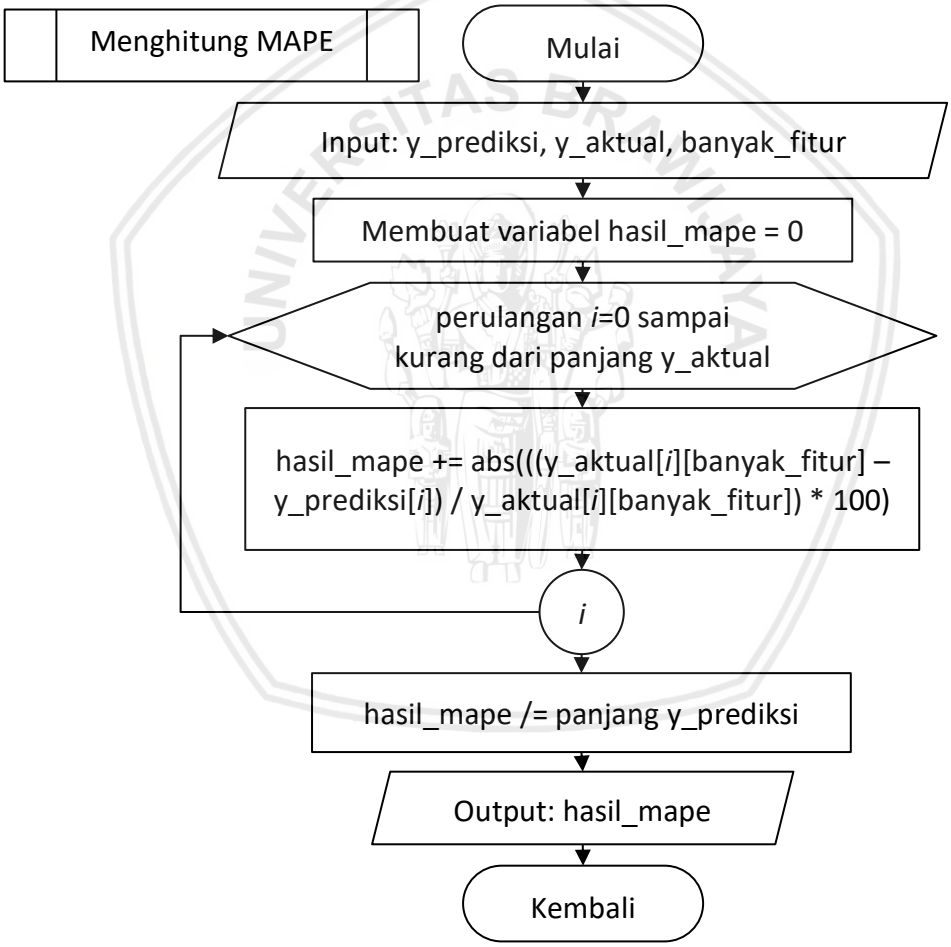


Gambar 4.14 Diagram Alir *Denormalisasi* Data

Dari diagram alir pada bagian Gambar 4.14 dapat dijelaskan setiap langkah dari proses *denormalisasi* data sebagai berikut.

1. Memasukkan input dari nilai y_topi, minimum dengan nilai sebesar Rp5000,00 dan maksimum dengan nilai sebesar Rp150.000,00.
2. Membuat variabel list kosong dengan nilai hasil_denormalisasi.
3. Melakukan perulangan dari *i*=0 sampai kurang dari panjang y_topi.
4. Melakukan perulangan dari *j*=0 sampai kurang dari panjang y_topi[0].
5. Melakukan perhitungan rumus *denormalisasi* data sesuai pada Persamaan 2.6 dan setelah itu hasil dari rumus *denormalisasi* disimpan pada variabel prediksi dengan indeks ke-*i*.
6. Menghasilkan keluaran nilai hasil_denormalisasi.

4.2.6 Alir Menghitung MAPE



Gambar 4.15 Diagram Alir Menghitung MAPE

Dari diagram alir pada bagian Gambar 4.15 dapat dijelaskan setiap langkah dari proses menghitung nilai MAPE sebagai berikut.

1. Memasukkan input y_prediksi (hasil dari proses *denormalisasi*), y_aktual atau data sebenarnya, dan banyak_fitur.
2. Membuat variabel dengan nama hasil_mape yang bernilai 0.

3. Melakukan perulangan sebanyak *i* dimulai dengan nilai *i*=0 sampai kurang dari panjang y_aktual.
4. Melakukan proses perhitungan dengan menggunakan Persamaan 2.13 pada bagian menjumlahkan nilai sigma.
5. Melakukan proses perhitungan MAPE dari nilai hasil_mape sebelumnya dibagi dengan panjang y_prediksi.
6. Menghasilkan nilai MAPE untuk mengetahui selisih *error* yang dihasilkan.

4.3 Perhitungan Manualisasi

Peda proses di bagian perhitungan manualisasi akan dilakukan tahap untuk menghitung setiap langkah yang dikerjakan oleh algoritme secara manual. Pada perhitungan ini menjelaskan langkah manual dari algoritme ELM untuk kasus prediksi. Dalam perhitungan prediksi tentang harga cabai rawit di Kota Malang akan dijelaskan setiap langkah tahapan dari proses yaitu normalisasi data, proses *training*, proses *testing*, *denormalisasi* data, dan perhitungan evaluasi menggunakan MAPE. Sebelum menuju proses algoritme ELM diperlukan persiapan data dalam perhitungan sebanyak 12 dapat dilihat pada Tabel 4.1.

Tabel 4.1 Data Harga Cabai Rawit di Kota Malang

No	Tanggal	Harga (rupiah/kilogram)
1	1 Januari 2017	72.400
2	2 Januari 2017	73.200
3	3 Januari 2017	77.200
4	4 Januari 2017	78.400
5	5 Januari 2017	83.600
6	6 Januari 2017	89.800
7	7 Januari 2017	92.000
8	8 Januari 2017	93.200
9	9 Januari 2017	93.200
10	10 Januari 2017	93.200
11	11 Januari 2017	94.400
12	12 Januari 2017	88.600

4.3.1 Inisialisasi Banyak Fitur, *Hidden Neuron,* dan Fungsi Aktivasi

Sebelum data harga cabai rawit diproses terdapat beberapa inisialisasi yang akan dilakukan yaitu inisialisasi banyak fitur, banyak *hidden neuron*, dan jenis fungsi aktivasi yang akan dipakai. Pada perhitungan ini akan menggunakan banyak *hidden neuron* sebanyak 3 dan fungsi aktivasi yang dipakai fungsi *sigmoid biner*. Sedangkan untuk banyak fitur sebanyak 5 fitur. Dalam proses perubahan data menjadi bentuk fitur dapat dilakukan dengan analisis teknikal. Dalam perhitungan ini menggunakan data 5 hari sebelumnya sebagai fitur dan target di

hari keenam. Sehingga banyak fitur yang terbentuk sebanyak 5 dan 1 data target. Kelima fitur tersebut dituliskan dalam bentuk *X1* (5 hari sebelumnya), *X2* (4 hari sebelumnya), *X3* (3 hari sebelumnya), *X4* (2 hari sebelumnya), *X5* (1 hari sebelumnya). Target yang akan diprediksi dituliskan dalam bentuk *Y* (target prediksi hari keenam). Hasil yang dibentuk akan dilakukan pembagian data untuk *training* sebanyak 5 data dan data untuk *testing* sebanyak 2 data. Dari data yang terbentuk dapat dilihat pada bagian Tabel 4.2 dan bagian Tabel 4.3.

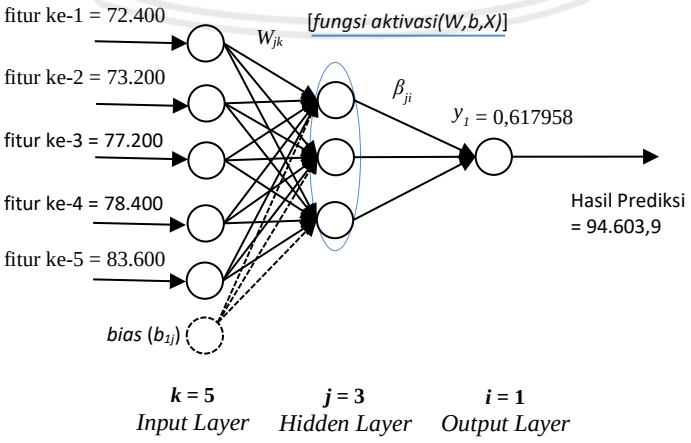
Tabel 4.2 Data *Training*

Data ke-	X1	X2	X3	X4	X5	Y
1	72.400	73.200	77.200	78.400	83.600	89.800
2	73.200	77.200	78.400	83.600	89.800	92.000
3	77.200	78.400	83.600	89.800	92.000	93.200
4	78.400	83.600	89.800	92.000	93.200	93.200
5	83.600	89.800	92.000	93.200	93.200	93.200

Tabel 4.3 Data *Testing*

Data ke-	X1	X2	X3	X4	X5	Y
1	89.800	92.000	93.200	93.200	93.200	94.400
2	92.000	93.200	93.200	93.200	94.400	88.600

Dari data yang telah dilakukan analisis teknikal menjadi 5 fitur berdasarkan hari sebelumnya. Dari nilai di atas diambil data ke-1 pada data *testing* sebagai contoh bentuk penerapan dari nilai tersebut dalam arsitektur algoritme ELM. Hasil penerapan pada arsitektur algoritme ELM terdapat pada Gambar 4.16.



Gambar 4.16 Penerapan Arsitektur Algoritme *Extreme Learning Machine*

Pada arsitektur di atas digunakan untuk arsitektur proses *training* dan proses *testing*. Pada arsitektur proses *training* data menggunakan input dari data *training* sebanyak 5 dengan fitur ke-1 sampai fitur ke-5. Selanjutnya menentukan nilai random *bias* antara [0,1] dan bobot (*w*) antara [-1,1]. Dari nilai fitur, *w*, dan *bias* dilakukan perhitungan pada *hidden layer* untuk nilai H_{init} *training*, setelah itu menghitung nilai (*H*) *training* menggunakan fungsi aktivasi. Dari nilai *H* hitung nilai H^{+} dengan matriks *Moore-Penrose Generalized Inverse* dan hasil untuk keluaran *hidden layer* menghasilkan nilai $\hat{\beta}$. Dari penjelasan arsitektur proses *testing* data yang digunakan sebagai input fitur diambil dari data *testing* dengan panjang fitur sebanyak 5. Setelah itu lakukan perhitungan nilai H_{init} *testing* dengan nilai bobot (*w*) dan *bias* diambil dari proses *training*, selanjutnya menghitung nilai (*H*) *testing* dengan fungsi aktivasi. Selanjutnya menghitung nilai $\hat{Y}$ dari nilai *H testing* dikalikan dengan nilai $\hat{\beta}$ dari proses *training*. Hasil prediksi nilai $\hat{Y}$ akan dilakukan *denormalisasi* data sehingga menghasilkan nilai prediksi.

4.3.2 Perhitungan Normalisasi Data

Sebelum data hasil menjadi bentuk fitur diproses terlebih dahulu dilakukan proses menyeimbangkan nilai data dengan normalisasi. Dalam normalisasi data ini digunakan perhitungan *min-max normalization*. Pada *min-max normalization* terlebih dahulu harus menentukan berapa nilai minimum dan maksimum. Untuk menanggulangi agar hasil normalisasi tidak menemukan nilai yang memiliki hasil dominan bernilai 0 dan 1. Maka untuk nilai minimum dan maksimum akan ditetapkan dengan nilai yang tidak pernah dijangkau data berdasarkan hasil analisis data dan informasi pakar. Data untuk nilai minimum dan nilai maksimum dapat dilihat pada bagian Tabel 4.4.

Tabel 4.4 Nilai Harga Minimum dan Maksimum

	Harga (rupiah/Kilogram)
Minimum	5.000
Maksimum	150.000

Proses perhitungan selanjutnya adalah menghitung normalisasi data menggunakan sesuai dengan rumus pada Persamaan 2.5. Berikut hasil perhitungan manual untuk normalisasi data *training* pada data ke-1 dan fitur X1.

$$X_{(1,1)} = \frac{\left(X_{(1,1)} - X_{min}\right)}{(X_{max} - X_{min})}$$

$$X_{(1,1)} = \frac{(72400 - 5000)}{(150000 - 5000)}$$

$$X_{(1,1)} = \frac{67400}{145000}$$

$$X_{(1,1)} = 0{,}464828$$

Hasil detail normalisasi dapat diamati pada bagian Tabel 4.5 dan Tabel 4.6.

Tabel 4.5 Normalisasi Data *Training*

Data ke-	X1	X2	X3	X4	X5	Y
1	0,464828	0,470345	0,497931	0,506207	0,542069	0,584828
2	0,470345	0,497931	0,506207	0,542069	0,584828	0,6
3	0,497931	0,506207	0,542069	0,584828	0,6	0,608276
4	0,506207	0,542069	0,584828	0,6	0,608276	0,608276
5	0,542069	0,584828	0,6	0,608276	0,608276	0,608276

Tabel 4.6 Normalisasi Data *Testing*

Data ke-	X1	X2	X3	X4	X5	Y
1	0,584828	0,6	0,608276	0,608276	0,608276	0,616552
2	0,6	0,608276	0,608276	0,608276	0,616552	0,576552

4.3.3 Perhitungan Proses *Training*

Proses perhitungan selanjutnya adalah menghitung proses *training* yang terbagi menjadi beberapa tahap. Berikut tahapan *training* pada algoritme ELM.

4.3.3.1 Menghitung Nilai H_{init}

Pada saat nilai H_{init} pada proses *training* menggunakan Persamaan 2.7. Sebelum dilakukan proses perhitungan H_{init} dibutuhkan nilai bobot (*weight*) dan *bias* yang dihasilkan dengan cara *random*. Rentang *random* bobot adalah [-1,1] dan *bias* [0,1]. Hasil *random* untuk nilai *weight* dapat dilihat pada Tabel 4.7 dan hasil *weight* transposisi di Tabel 4.8, untuk *bias* terdapat pada bagian Tabel 4.9.

Tabel 4.7 Hasil Weight (*W*)

j \ *k*	1	2	3	4	5
1	0,990742	0,818194	0,280393	-0,63475	-0,95839
2	0,016035	-0,23643	0,319082	0,546846	-0,79365
3	0,074944	-0,5763	-0,5852	-0,09018	-0,2244

Tabel 4.8 Hasil *Weight* Transposisi (W^T)

k \ *j*	1	2	3
1	0,990742	0,016035	0,074944
2	0,818194	-0,236431	-0,576296
3	0,280393	0,319082	-0,5852

Tabel 4.8 Hasil *Weight* Transposisi (W^T) (Lanjutan)

k \ j	1	2	3
4	-0,634746	0,546846	-0,090182
5	-0,958388	-0,793647	-0,224396

Tabel 4.9 Hasil *Bias*

1 \ j	1	2	3
1	0,602592	0,765555	0,941669

Setelah nilai *w* dan *bias* didapatkan, langkah selanjutnya adalah menghitung nilai H_{init}. Proses contoh perhitungan perhitungan H_{init} untuk bagian baris ke-1 dan kolom ke-1 sebagai berikut.

$$H_{init_{(1,1)}} = X_{training}.W^T + bias$$

$$H_{init_{(1,1)}} = (0{,}464828 \text{ x } 0{,}990742) + (0{,}470345 \text{ x } 0{,}818194) + (0{,}497931 \text{ x } 0{,}280393) + (0{,}506207 \text{ x}(-0{,}634746)) + (0{,}542069 \text{ x}(-0{,}958388)) + 0{,}602592$$

$$H_{init_{(1,1)}} = 0{,}46052462 + 0{,}38483345 + 0{,}13961636 + (-0{,}32131286) + (-0{,}51951242) + 0{,}602592$$

$$H_{init_{(1,1)}} = 0{,}746741$$

Hasil perhitungan dari H_{init} *training* dapat dilihat pada bagian Tabel 4.10.

Tabel 4.10 Hasil H_{init} (*Training*)

k \ j	1	2	3
1	0,746741	0,667291	0,246769
2	0,713355	0,649173	0,213613
3	0,715831	0,670443	0,182664
4	0,7478	0,677469	0,134369
5	0,817316	0,677302	0,10279

4.3.3.2 Menghitung Nilai Keluaran *Hidden Layer*

Hasil pada saat perhitungan keluaran dari *hidden layer* dilakukan, dibutuhkan persamaan fungsi aktivasi. Pada bagian perhitungan manual ini fungsi aktivasi *sigmoid biner* dipilih sebagai proses untuk menghasilkan nilai *H*. Hasil yang diperoleh melalui fungsi aktivasi sesuai dengan rumus pada Persamaan 2.8 untuk *H* data *training* untuk baris ke-1 dan bagian kolom ke-1.

$$H_{(1,1)} = \frac{1}{1 + \exp(-0{,}746741)}$$

$$H_{(1,1)} = \frac{1}{1 + 0{,}473909}$$

$$H_{(1,1)} = \frac{1}{1{,}473909}$$

$$H_{(1,1)} = 0{,}678468$$

Hasil perhitungan dari *H training* dengan fungsi aktivasi *sigimoid biner* dapat dilihat pada bagian Tabel 4.11.

Tabel 4.11 Hasil Keluaran *Hidden Layer* (*Training*)

j / *k*	1	2	3
1	0,678468	0,660896	0,561381
2	0,671142	0,656824	0,553201
3	0,671688	0,661602	0,545539
4	0,678699	0,663174	0,533542
5	0,693666	0,663136	0,525675

4.3.3.3 Menghitung Nilai Matriks *Moore-Penrose Generalized Inverse*

Hasil dari perhitungan matriks *Moore-Penrose Generalized* (H^+) dengan menggunakan rumus sesuai pada Persamaaan 2.9. Contoh hasil perhitungan dari penerapan persamaan tersebut sebagai berikut.

$$(H^T.\ H)_{(1,1)} = (0{,}678468 \text{ x } 0{,}678468) + (0{,}671142 \text{ x } 0{,}671142) + (0{,}671688 \text{ x } 0{,}671688) + (0{,}678699 \text{ x } 0{,}678699) + (0{,}693666 \text{ x } 0{,}693666)$$

$$(H^T.\ H)_{(1,1)} = 0{,}460319 + 0{,}450432 + 0{,}451165 + 0{,}460632 + 0{,}481173$$

$$(H^T.\ H)_{(1,1)} = 2{,}30372$$

Hasil transposisi nilai H terdapat pada Tabel 4.12 sedangkan untuk perhitungan perkalian H^TxH terdapat pada Tabel 4.13 dan hasil inversi dapat ditampilkan pada bagian Tabel 4.14.

Tabel 4.12 Hasil *H* Transposisi

k / *j*	1	2	3	4	5
1	0,678468	0,671142	0,671688	0,678699	0,693666
2	0,660896	0,656824	0,661602	0,663174	0,663136
3	0,561381	0,553201	0,545539	0,533542	0,525675

Tabel 4.13 Hasil Perkalian H^TxH

j \ j	1	2	3
1	2,30372	2,2437	1,845345
2	2,2437	2,185468	1,797725
3	1,845345	1,797725	1,479794

Selanjutnya dari hasil perkalian H^TxH akan dihitung nilai inversi $(H^TxH)^{-1}$. Perhitungan nilai inversi matriks berikut menggunakan metode Operasi Baris Elementer (OBE). Berikut langkah dalam penyelesaian inversi matriks dengan OBE.

1. Langkah pertama tambahkan matriks identitas pada matriks *H*.
$$[H|I] = \left[\begin{array}{ccc|ccc} 2{,}30372 & 2{,}2437 & 1{,}845345 & 1 & 0 & 0 \\ 2{,}2437 & 2{,}185468 & 1{,}797725 & 0 & 1 & 0 \\ 1{,}845345 & 1{,}797725 & 1{,}479794 & 0 & 0 & 1 \end{array}\right]$$
2. Langkah kedua melakukan operasi $R1/2{,}30372$ artinya membagi matriks baris ke-1 dengan nilai $2{,}30372$.
$$[H|I] = \left[\begin{array}{ccc|ccc} 1 & 0{,}973946 & 0{,}801028 & 0{,}434081 & 0 & 0 \\ 2{,}2437 & 2{,}185468 & 1{,}797725 & 0 & 1 & 0 \\ 1{,}845345 & 1{,}797725 & 1{,}479794 & 0 & 0 & 1 \end{array}\right]$$
3. Langkah ketiga melakukan operasi $\mathrm{R2} - (2{,}2437 \text{ x } \mathrm{R1})$ artinya matriks baris ke-2 dikurangi dengan hasil perkalian antara $2{,}2437$ dan baris ke-1.
$$[H|I] = \left[\begin{array}{ccc|ccc} 1 & 0{,}973946 & 0{,}801028 & 0{,}434081 & 0 & 0 \\ 0 & 0{,}000224 & 0{,}000458 & -0{,}973946 & 1 & 0 \\ 1{,}845345 & 1{,}797725 & 1{,}479794 & 0 & 0 & 1 \end{array}\right]$$
4. Langkah keempat melakukan operasi $\mathrm{R3} - (1{,}845345 \text{ x } \mathrm{R1})$ artinya matriks baris ke-3 dikurangi dengan hasil perkalian antara 1,845345 dan baris ke-1.
$$[H|I] = \left[\begin{array}{ccc|ccc} 1 & 0{,}973946 & 0{,}801028 & 0{,}434081 & 0 & 0 \\ 0 & 0{,}000224 & 0{,}000458 & -0{,}973946 & 1 & 0 \\ 0 & 0{,}000458 & 0{,}00162 & -0{,}801028 & 0 & 1 \end{array}\right]$$
5. Langkah kelima melakukan operasi $\mathrm{R2}/0{,}000224$ artinya membagi matriks baris ke-2 dengan nilai 0,000224.
$$[H|I] = \left[\begin{array}{ccc|ccc} 1 & 0{,}973946 & 0{,}801028 & 0{,}434081 & 0 & 0 \\ 0 & 1 & 2{,}040953 & -4342{,}777182 & 4458{,}948455 & 0 \\ 0 & 0{,}000458 & 0{,}00162 & -0{,}801028 & 0 & 1 \end{array}\right]$$
6. Langkah keenam melakukan operasi $\mathrm{R1} - (0{,}973946 \text{ x } \mathrm{R2})$ artinya matriks baris ke-1 dikurangi dengan hasil perkalian antara 0,973946 dan baris ke-2.
$$[H|I] = \left[\begin{array}{ccc|ccc} 1 & 0 & -1{,}186751 & 4230{,}066659 & -4342{,}777182 & 0 \\ 0 & 1 & 2{,}040953 & -4342{,}777182 & 4458{,}948455 & 0 \\ 0 & 0{,}000458 & 0{,}00162 & -0{,}801028 & 0 & 1 \end{array}\right]$$
7. Langkah ketujuh melakukan operasi $\mathrm{R3} - (0{,}000458 \text{ x } \mathrm{R2})$ artinya matriks baris ke-3 dikurangi dengan hasil perkalian antara 0,000458 dan baris ke-2.

$$[H|I] = \left[\begin{array}{ccc|ccc} 1 & 0 & -1{,}186751 & 4230{,}066659 & -4342{,}777182 & 0 \\ 0 & 1 & 2{,}040953 & -4342{,}777182 & 4458{,}948455 & 0 \\ 0 & 0 & 0{,}000686 & 1{,}186751 & -2{,}040953 & 1 \end{array}\right]$$

8. Langkah kedelapan melakukan operasi $\mathrm{R3}/0{,}000686$ artinya membagi matriks baris ke-3 dengan nilai 0,000686.

$$[H|I] = \left[\begin{array}{ccc|ccc} 1 & 0 & -1{,}186751 & 4230{,}066659 & -4342{,}777182 & 0 \\ 0 & 1 & 2{,}040953 & -4342{,}777182 & 4458{,}948455 & 0 \\ 0 & 0 & 1 & 1729{,}511141 & -2974{,}382623 & 1457{,}349647 \end{array}\right]$$

9. Langkah kesembilan melakukan operasi $\mathrm{R1} - (-1{,}186751 \text{ x } \mathrm{R3})$ artinya matriks baris ke-1 dikurangi dengan hasil perkalian antara -1,186751 dan baris ke-3.

$$[H|I] = \left[\begin{array}{ccc|ccc} 1 & 0 & 0 & 6282{,}565724 & -7872{,}628714 & 1729{,}511141 \\ 0 & 1 & 2{,}040953 & -4342{,}777182 & 4458{,}948455 & 0 \\ 0 & 0 & 1 & 1729{,}511141 & -2974{,}382623 & 1457{,}349647 \end{array}\right]$$

10. Langkah kesepuluh melakukan operasi $\mathrm{R2} - (3{,}533924 \text{ x } \mathrm{R3})$ artinya artinya matriks baris ke-2 dikurangi dengan hasil perkalian antara 3,533924 dan baris ke-3.

$$[H|I] = \left[\begin{array}{ccc|ccc} 1 & 0 & 0 & 6282{,}565724 & -7872{,}628714 & 1729{,}511141 \\ 0 & 1 & 0 & -7872{,}628714 & 10529{,}52459 & -2974{,}382623 \\ 0 & 0 & 1 & 1729{,}511141 & -2974{,}382623 & 1457{,}349647 \end{array}\right]$$

Hasil dari perhitungan inversi matriks dengan cara Operasi Baris Elementer (OBE) dapat dilihat pada bagian Tabel 4.14.

Tabel 4.14 Hasil Inversi ($H^T xH$)$^{-1}$

j \ *j*	1	2	3
1	6282,565724	-7872,628714	1729,511141
2	-7872,628714	10529,52459	-2974,382623
3	1729,511141	-2974,382623	1457,349647

Setelah hasil perkalian dilakukan inversi menggunakan perkalian matriks *Moore-Penrose Generalized Inverse*, hasil dari perhitungan inversi akan dilakukan kembali perkalian dengan matriks H^T. Hasil akhir dari perkalian matriks H^T akan menghasilkan nilai H^+. Berikut contoh hasil perhitungan dapat dilihat pada perkalian matriks.

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

$$H^+_{(1,1)} = (6282{,}565724 \text{ x } 0{,}678468) + (-7872{,}628714 \text{ x } 0{,}660896) + (1729{,}511141 \text{ x } 0{,}561381)$$

$$H^+_{(1,1)} = 4262{,}519802 + (-5202{,}988827) + 970{,}914694$$

$$H^+_{(1,1)} = 30{,}445669$$

Hasil dari perhitungan matriks *Moore-Penrose Generalized Inverse* terdapat pada bagian Tabel 4.15.

Tabel 4.15 Hasil H^+

j \ k	1	2	3	4	5
1	30,445669	2,329535	-45,107118	-34,184767	46,53949
2	-52,167866	-13,038762	55,76257	52,803651	-42,026636
3	25,788789	13,308956	-11,124544	-21,158494	-6,624844

4.3.3.4 Menghitung Nilai *Output Weight*

Perhitungan nilai dari *output weight* ($\hat{\beta}$) dilakukan dengan cara menggunakan rumus sesuai pada Persamaan 2.10. Contoh perhitungan *output weight* ($\hat{\beta}$) dapat ditulis sebagai berikut.

$$\hat{\beta}_{(1,1)} = H^{+}.Y$$

$$\hat{\beta}_{(1,1)} = (30{,}445669 \text{ x } 0{,}584828) + (2{,}329535 \text{ x } 0{,}6) + (-45{,}107118 \text{ x } 0{,}608276) + (-34{,}184767 \text{ x } 0{,}608276) + (46{,}53949 \text{ x } 0{,}608276)$$

$$\hat{\beta}_{(1,1)} = 17{,}80548 + 1{,}397721 + (-27{,}437577) + (-20{,}793773) + 28{,}308855$$

$$\hat{\beta}_{(1,1)} = -0{,}719295$$

Setelah perhitungan dilakukan pada semua baris dan kolom, maka akan menghasilkan nilai *output weight* ($\hat{\beta}$). Hasil perhitungan nilai *output weight* ($\hat{\beta}$) dapat dilihat pada bagian Tabel 4.16.

Tabel 4.16 Hasil *Output Weight* (*Training*)

j \ i	1
1	-0,719295
2	2,141947
3	-0,599351

4.3.4 Perhitungan Proses *Testing*

Proses perhitungan selanjutnya adalah menghitung proses *training* yang terbagi menjadi beberapa tahap. Berikut tahapan *training* pada algoritme ELM.

4.3.4.1 Menghitung Nilai H_{init}

Pada bagian untuk perhitungan nilai H_{init} pada proses *testing* menggunakan Persamaan 2.11. Terlebih dahulu dibutuhkan nilai dari hasil *training* berupa nilai bobot (*weight*) dan *bias*. Hasil untuk nilai *weight* dan *bias* yang digunakan dalam

perhitungan manual tersebut dapat dilihat pada Tabel 4.8 dan Tabel 4.9. Proses perhitungan nilai H_{init} dapat dituliskan sebagai berikut.

$$H_{init_{(1,1)}} = X_{testing}.W^T + bias$$

$$H_{init_{(1,1)}} = (0{,}584828 \text{ x } 0{,}990742) + (0{,}6 \text{ x } 0{,}818194) + (0{,}608276 \text{ x } 0{,}280393) + (0{,}608276 \text{ x}(-0{,}634746)) + (0{,}608276 \text{ x}(-0{,}958388)) + 0{,}602592$$

$$H_{init_{(1,1)}} = 0{,}57941362 + 0{,}4909164 + 0{,}17055633 + (-0{,}38610075) + (-0{,}58296442) + 0{,}602592$$

$$H_{init_{(1,1)}} = 0{,}874413$$

Hasil perhitungan dari H_{init} pada *testing* dapat dilihat pada bagian Tabel 4.17.

Tabel 4.17 Hasil H_{init} (*Testing*)

H_{init}	1	2	3
1	0,874413	0,677041	0,092407
2	0,888285	0,668759	0,086918

4.3.4.2 Menghitung Nilai Keluaran *Hidden Layer*

Pada bagian menghitung nilai *H* dilakukan dengan menggunakan bantuan persamaan fungsi aktivasi. Fungsi aktivasi yang digunakan untuk menentukan hasil nilai *H* pada perhitungan hidden layer adalah fungsi aktivasi *sigmoid biner*. Hasil perhitungan manual dari nilai *H* pada proses *testing* sesuai dengan rumus pada Persamaan 2.8. Berikut contoh hasil perhitungan nilai *H* data *testing* pada baris ke-1 dan kolom ke-1 sebagai berikut.

$$H_{(1,1)} = \frac{1}{1 + \exp(-0{,}874413)}$$

$$H_{(1,1)} = \frac{1}{1 + 0{,}417107}$$

$$H_{(1,1)} = \frac{1}{1{,}417107}$$

$$H_{(1,1)} = 0{,}705663$$

Hasil perhitungan dari *H testing* dengan fungsi aktivasi *sigimoid biner* dapat dilihat pada bagian Tabel 4.18.

Tabel 4.18 Hasil Keluaran *Hidden Layer* (*Testing*)

H	1	2	3
1	0,705663	0,663078	0,523085
2	0,708536	0,661225	0,521716

4.3.4.3 Menghitung Nilai *Output Layer*

Pada bagian untuk menghitung nilai *output layer* ($\hat{Y}$) akan menggunakan proses perhitungan yang sudah dilakukan sebelumnya yaitu hasil pada nilai *output weight* proses *training*. Tahap selanjutnya setelah hasil tersebut didapatkan akan dilakukan perkalian matriks dengan nilai *H* hasil perhitungan terhadap data *testing*. Contoh perhitungan nilai $\hat{Y}$ pada proses *testing* sesuai dengan rumus pada Persamaan 2.12. Berikut hasil perhitungan nilai $\hat{Y}$ data *testing* pada baris ke-1 dan kolom ke-1 sebagai berikut.

$$\hat{Y}_{(1,1)} = H.\hat{\beta}$$

$$\hat{Y}_{(1,1)} = (0{,}705663 \text{ x } (-0{,}719295)) + (0{,}663078 \text{ x } 2{,}141947) + (0{,}523085 \text{ x } (-0{,}599351))$$

$$\hat{Y}_{(1,1)} = -0{,}5075799 + 1{,}4202779 + (-0{,}3135115)$$

$$\hat{Y}_{(1,1)} = 0{,}599187$$

Hasil perhitungan dari $\hat{Y}$ dapat dilihat pada bagian Tabel 4.19.

Tabel 4.19 Hasil *Output Layer*

$\hat{Y}$	1
1	0,599187
2	0,593971

4.3.5 Perhitungan *Denormalisasi* Data

Pada bagian menghitung nilai *denormalisasi* bertujuan untuk perubahan nilai hasil prediksi menjadi bentuk data yang sebelum dinormalisasi. Nilai parameter minimum dan maksimum sama dengan pada proses *training* dengan menggunakan Tabel 4.4. Contoh perhitungan nilai *denormalisasi* data pada proses *testing* menggunakan Persamaan 2.6. Berikut untuk hasil perhitungan nilai *denormalisasi* data pada baris ke-1 dan kolom ke-1 sebagai berikut.

$$X_{(1,1)} = X_{(1,1)} \,.\, (X_{max} - X_{min}) + X_{min}$$

$$X_{(1,1)} = 0{,}599187 \,.\, (150000 - 5000) + 5000$$

$$X_{(1,1)} = 0{,}599187 \,.\, 145000 + 5000$$

$$X_{(1,1)} = 86882{,}115 + 5000$$

$$X_{(1,1)} = 91882{,}115$$

Hasil perhitungan dari *denormalisasi* dapat dilihat pada bagian Tabel 4.20.

Tabel 4.20 Hasil *Denormalisasi* Data

Denormalisasi	1
1	91882,115
2	91125,795

4.3.6 Perhitungan MAPE

Setelah didapatkan hasil hasil prediksi setelah *denormalisasi*, maka langkah selanjutnya adalah menghitung besarnya nilai *error* yang dihasilkan dengan menggunakan rumus sesuai pada Persamaan 2.13. Berikut contoh hasil perhitungan MAPE antara nilai prediksi dan nilai aktual sebagai berikut.

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

$$MAPE = \frac{1}{2}\left(\left|\frac{94400 - 91882{,}115}{94400}\ x\ 100\right| + \left|\frac{88600 - 91125{,}795}{88600}\ x\ 100\right|\right)$$

$$MAPE = \frac{1}{2}(2{,}667251 + 2{,}850784)$$

$$MAPE = \frac{1}{2}(5{,}518035)$$

$$MAPE = 2{,}759018$$

Tabel untuk hasil nilai prediksi dan nilai aktual terdapat pada Tabel 4.21.

Tabel 4.21 Perbandingan Nilai Prediksi dan Nilai Aktual

MAPE	Nilai Prediksi	Nilai Aktual
1	91882,115	94.400
2	91125,795	88.600

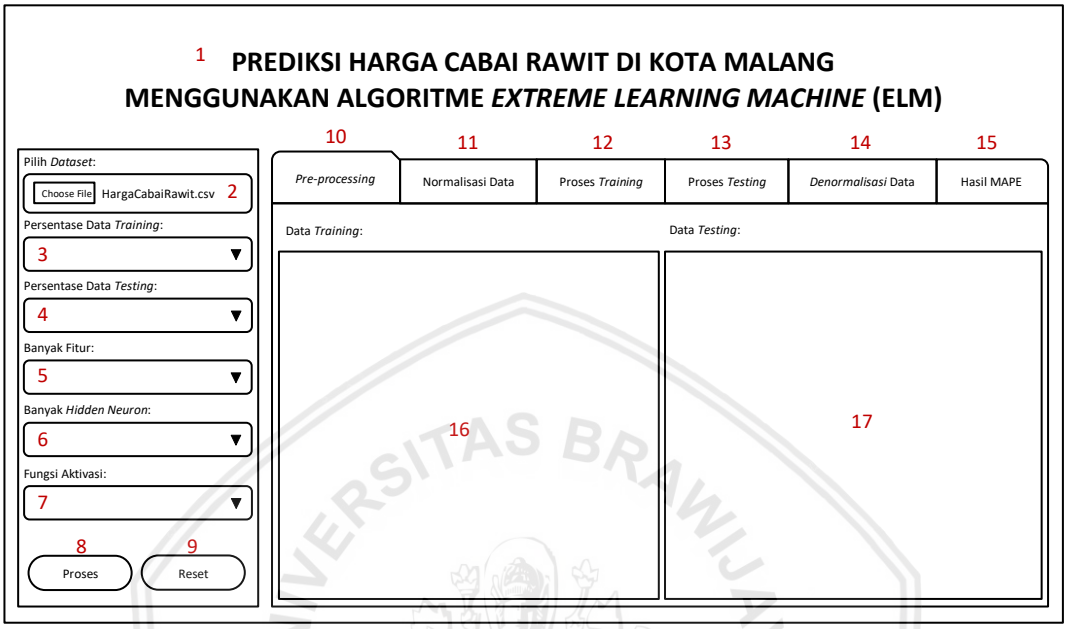
Dari tabel di atas diperoleh hasil selisih dari nilai prediksi dan nilai aktual untuk data ke-1 sebesar Rp2.517,88 sedangkan data ke-2 sebesar Rp2.525,79. Selisih nilai tersebut jika diamati bahwa nilai prediksi yang dihasilkan melebihi dari hasil nilai aktual, sehingga dari pihak konsumen akan mengalami kerugian dalam pembelian harga cabai rawit rata-rata sebesar Rp2.521,83 per kilogram karena harga yang diprediksi lebih tinggi.

4.4 Perancangan Antarmuka

Pada bagian untuk perancangan antarmuka sistem bertujuan untuk menjelaskan rancangan dari tampilan implementasi untuk penelitian tentang prediksi harga cabai rawit di Kota Malang menggunakan algoritme ELM. Perancangan antarmuka penelitian ini terdapat 6 halaman untuk menampilkan *pre-processing* data, normalisasi data, proses *training*, proses *testing*, *denormalisasi* data, dan hasil MAPE. Pada halaman proses *training* terdapat 6 sub halaman untuk menampilkan nilai bobot, *bias*, H_{init} *training*, *H training*, H^{+}, dan $\hat{\beta}$. Sedangkan pada halaman proses *testing* terdapat 3 sub halaman untuk menampilkan nilai H_{init} *testing*, *H testing*, dan $\hat{Y}$.

4.4.1 Perancangan Antarmuka *Pre-processing* Data

Pada halaman rancangan antarmuka ini sebagai halaman pertama yang akan ditampilkan saat pertama kali dijalankan. Berikut rancangan antarmuka untuk halaman inisialisasi parameter dan data dapat dilihat pada bagian Gambar 4.17.



Gambar 4.17 Perancangan Antarmuka *Pre-processing* Data

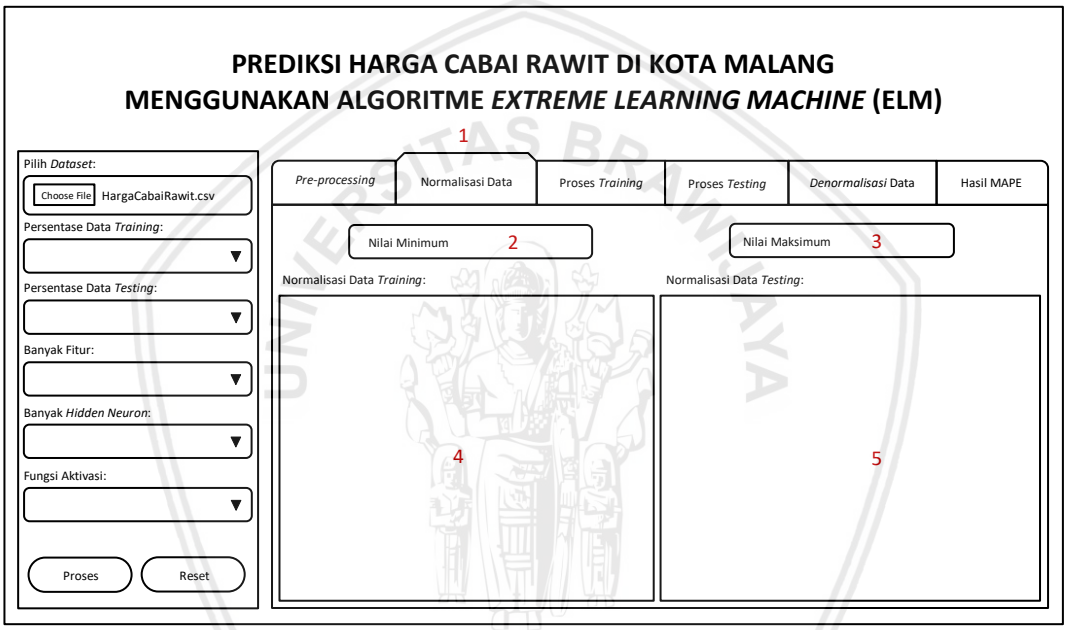
Keterangan:

1. Nama judul dari penelitian.
2. *Form* input *file* berfungsi sebagai masukan file data.
3. *Form* input *select* berfungsi memilih masukan untuk banyak persentase data *training*.
4. *Form* input *select* berfungsi memilih masukan untuk banyak persentase data *testing*.
5. *Form* input *select* berfungsi memilih masukan untuk banyak fitur.
6. *Form* input *select* berfungsi memilih masukan untuk banyak *hidden neuron*.
7. *Form* input *select* berfungsi memilih masukan untuk fungsi aktivasi.
8. *Button* Proses berfungsi sebagai tombol dimulai proses algoritme.
9. *Button* Reset berfungsi sebagai tombol reset nilai *select* menjadi kosong.
10. *Tab* Data berfungsi sebagai tampilan untuk halaman hasil data.
11. *Tab* Normalisasi Data berfungsi sebagai tampilan untuk halaman dari hasil normalisasi data.
12. *Tab* Proses *Training* berfungsi sebagai tampilan untuk halaman dari hasil proses *training* data.
13. *Tab* Proses *Testing* berfungsi sebagai tampilan untuk halaman dari hasil proses *testing* data.
14. *Tab Denormalisasi* Data berfungsi untuk menampilkan halaman dari hasil *denormalisasi* data.

15. *Tab* Hasil MAPE berfungsi untuk menampilkan halaman dari hasil perhitungan MAPE.
16. Tabel Data *Training* berfungsi menampilkan tabel hasil untuk data *training* setelah diubah menjadi fitur.
17. Tabel Data *Testing* berfungsi menampilkan tabel hasil untuk data *testing* setelah diubah menjadi fitur.

4.4.2 Perancangan Antarmuka Halaman Normalisasi Data

Pada halaman rancangan antarmuka ini akan menampilkan rancangan halaman normalisasi data. Dalam rancangan normalisasi data dibutuhkan form input dan table. Berikut rancangan antarmuka untuk halaman normalisasi data dapat dilihat pada bagian Gambar 4.18.



Gambar 4.18 Perancangan Antarmuka Halaman Normalisasi

Keterangan:

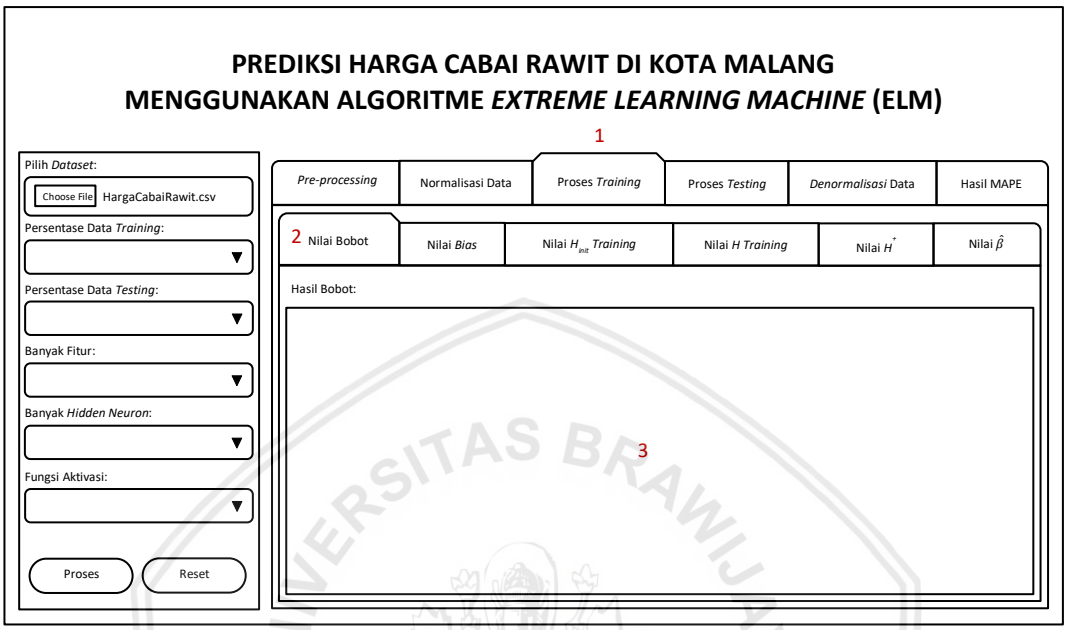
1. *Tab* Normalisasi Data berfungsi menampilkan halaman normalisasi data.
2. *Form* nilai minimum berfungsi untuk menampilkan hasil nilai minimum.
3. *Form* nilai maksimum berfungsi untuk menampilkan hasil nilai maksimum.
4. Tabel Normalisasi Data *Training* berfungsi untuk menampilkan tabel hasil normalisasi dari data *training*.
5. Tabel Hasil Normalisasi Data *Testing* berfungsi untuk menampilkan tabel hasil normalisasi dari data *testing*.

4.4.3 Perancangan Antarmuka Halaman Proses *Training*

Pada halaman rancangan antarmuka ini akan menampilkan rancangan halaman dari hasil proses *training* data. Rancangan halaman proses *training* terdapat 6 sub halaman berupa *tab* yaitu *tab* nilai bobot, *bias*, H_{init} *training*, *H training*, nilai H^{+}, dan $\hat{\beta}$.

4.4.3.1 Perancangan Antarmuka Nilai Bobot

Pada rancangan antarmuka ini untuk nilai bobot menampilkan hasil dari bobot yang telah di *generate* secara *random*. Berikut untuk tampilan antarmuka halaman nilai bobot dapat dilihat pada Gambar 4.19.



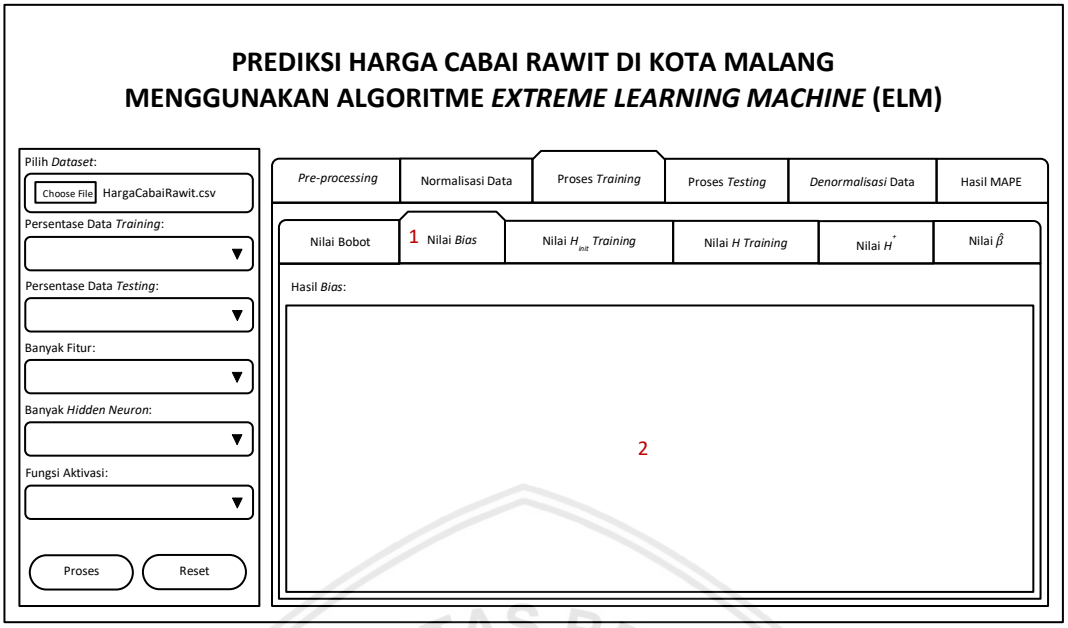
Gambar 4.19 Perancangan Antarmuka Halaman Nilai Bobot

Keterangan:

1. *Tab* Proses *Training* berfungsi menampilkan halaman proses *training* data.
2. *Tab* Nilai Bobot berfungsi untuk menampilkan halaman dari hasil nilai bobot.
3. Tabel untuk menampilkan hasil dari random nilai bobot.

4.4.3.2 Perancangan Antarmuka Nilai *Bias*

Pada rancangan ini akan ditampilkan nilai *bias* yang diperoleh secara random. Berikut tampilan antarmuka nilai *bias* dapat dilihat pada bagian Gambar 4.20.



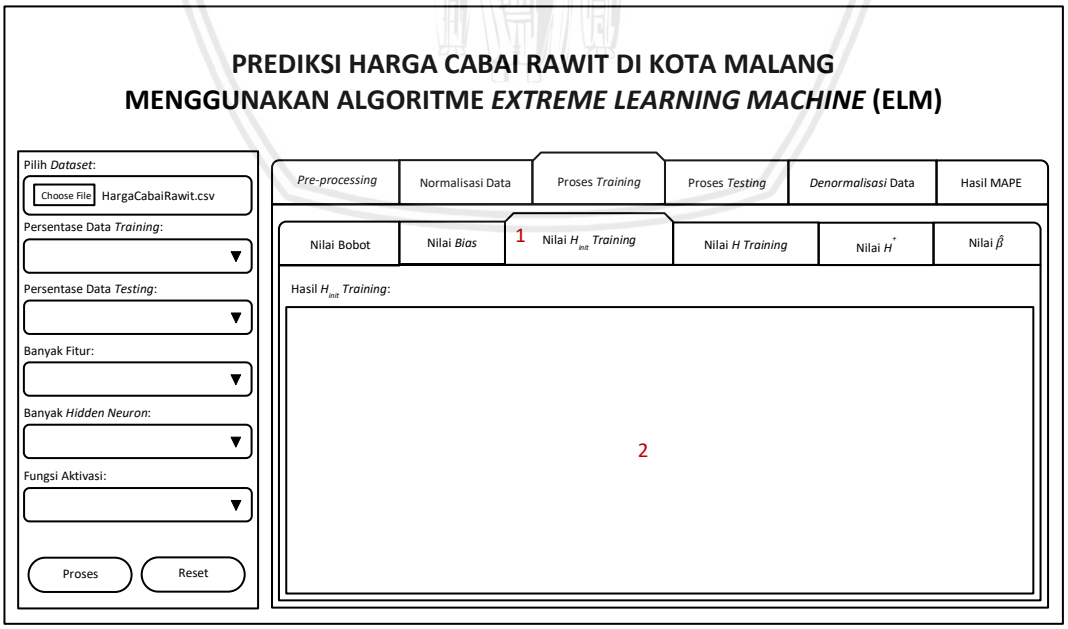
Gambar 4.20 Perancangan Antarmuka Halaman Nilai *Bias*

Keterangan:

1. *Tab* Nilai *Bias* berfungsi untuk menampilkan halaman dari hasil nilai *bias*.
2. Tabel sebagai tampilan untuk hasil dari nilai random pada *bias*.

4.4.3.3 Perancangan Antarmuka Nilai H_{init} *Training*

Pada rancangan antarmuka ini akan ditampilkan hasil dari nilai H_{init} untuk proses *training*. Berikut untuk tampilan antarmuka halaman nilai H_{init} *training* dapat dilihat pada bagian Gambar 4.21.



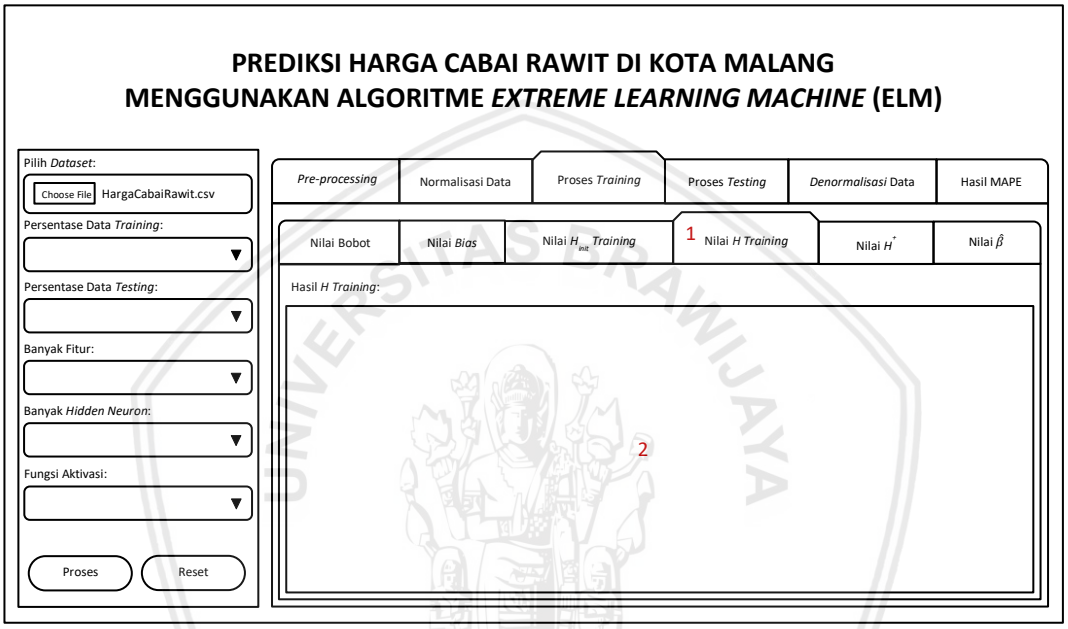
Gambar 4.21 Perancangan Antarmuka Halaman Nilai H_{init} *Training*

Keterangan:

1. *Tab* Nilai H_{init} *training* berfungsi untuk menampilkan halaman dari hasil nilai H_{init} *training*.
2. Tabel sebagai tampilan hasil perhitungan dari nilai H_{init} di proses *training*.

4.4.3.4 Perancangan Antarmuka Nilai *H Training*

Pada rancangan ini akan ditampilkan gambar untuk hasil hasil dari perhitungan nilai *H Training*. Berikut untuk tampilan antarmuka halaman nilai *H Training* dapat dilihat pada bagian Gambar 4.22.



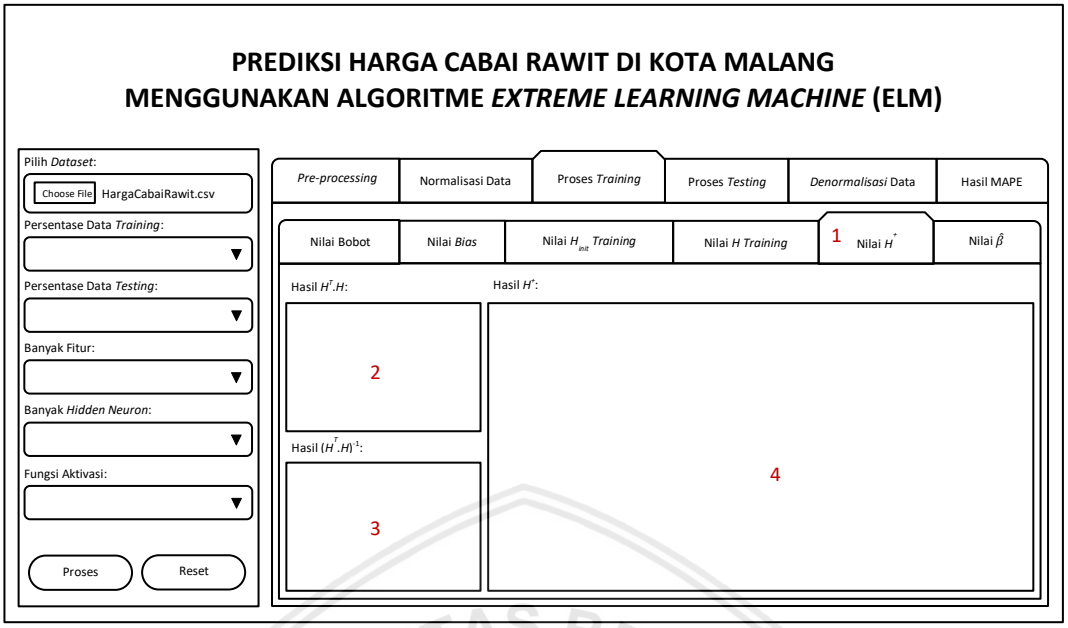
Gambar 4.22 Perancangan Antarmuka Halaman Nilai *H Training*

Keterangan:

1. *Tab* Nilai *H Training* berfungsi untuk menampilkan halaman dari hasil nilai *H Training* menggunakan fungsi aktivasi.
2. Tabel sebagai tampilan untuk hasil dari perhitungan nilai *H Training*.

4.4.3.5 Perancangan Antarmuka Nilai H^+

Pada rancangan ini akan ditampilkan hasil nilai H^+. Berikut untuk tampilan antarmuka halaman nilai H^+ dapat dilihat pada bagian Gambar 4.23.



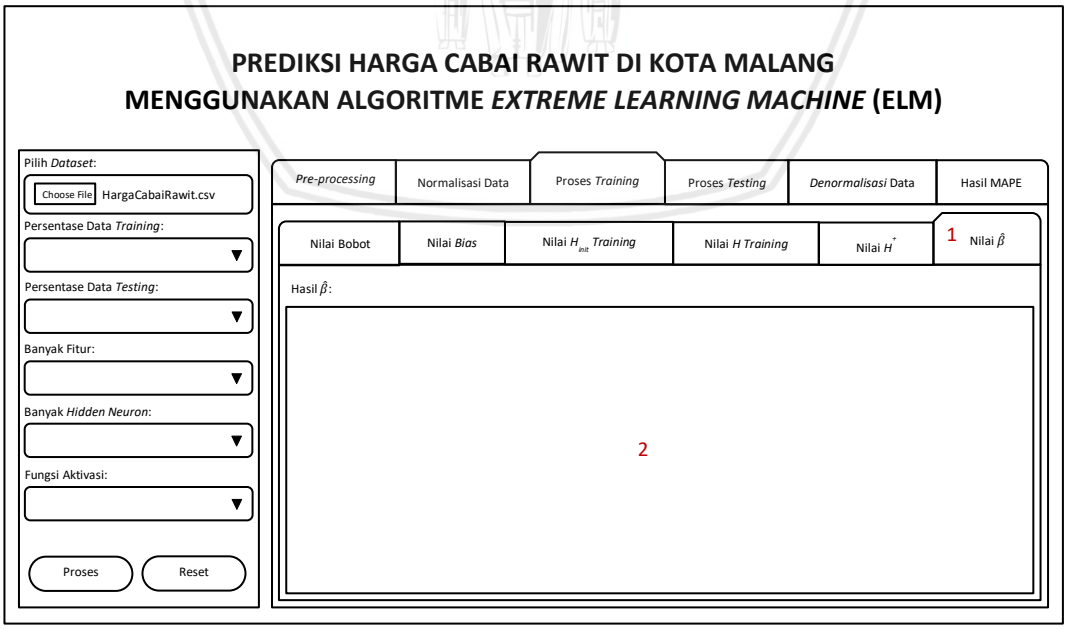
Gambar 4.23 Perancangan Antarmuka Halaman Nilai H^+

Keterangan:

1. *Tab* Nilai H^+ untuk perhitungan *Moore-Penrose Generalized Inverse*.
2. Tabel untuk menampilkan hasil perkalian matriks antara nilai H^T dan H.
3. Tabel untuk menampilkan hasil perhitungan inversi dari nilai H^T dikali H.
4. Tabel untuk menampilkan hasil perhitungan dari nilai H^+.

4.4.3.6 Perancangan Antarmuka Nilai $\widehat{\boldsymbol{\beta}}$

Berikut tampilan antarmuka halaman nilai $\hat{\beta}$ terdapat pada Gambar 4.24.



Gambar 4.24 Perancangan Antarmuka Halaman Nilai $\widehat{\boldsymbol{\beta}}$

Keterangan:

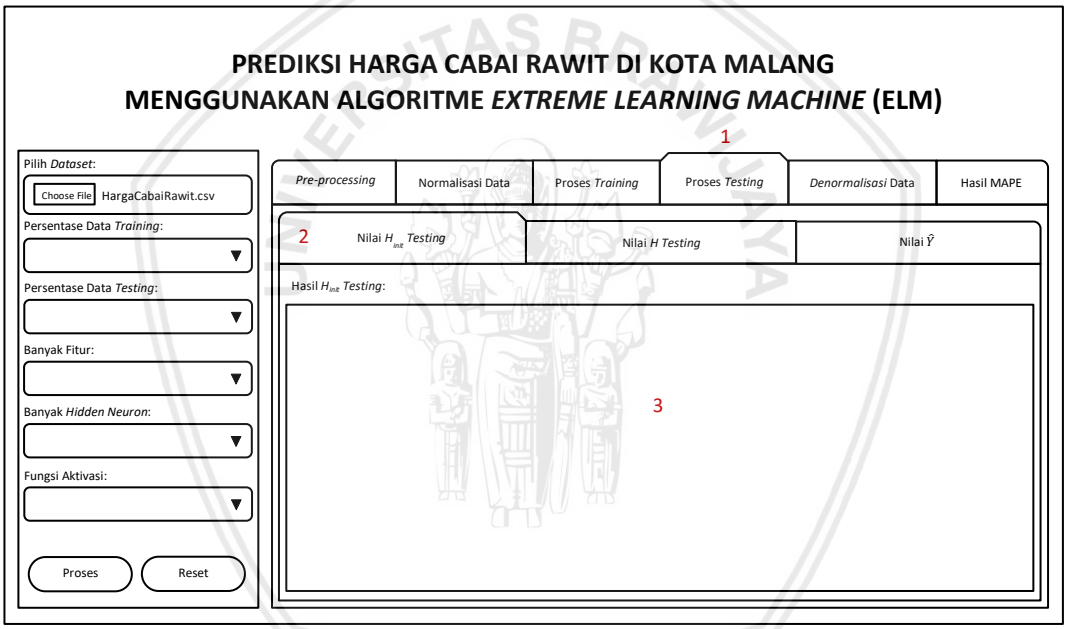
1. *Tab* Nilai $\hat{\beta}$ berfungsi untuk menampilkan halaman hasil perhitungan nilai $\hat{\beta}$.
2. Tabel untuk menampilkan hasil dari perhitungan nilai $\hat{\beta}$.

4.4.4 Perancangan Antarmuka Halaman Proses *Testing*

Pada halaman rancangan antarmuka ini akan menampilkan rancangan halaman antarmuka dari hasil proses *testing* data. Rancangan halaman proses *training* terdapat 3 sub halaman berupa *tab* yaitu *tab* nilai H_{init} *testing*, *H testing*, dan nilai $\hat{Y}$. Pada perhitungan H_{init} *testing* untuk nilai bobot dan *bias* menggunakan dari hasil pada proses *training*.

4.4.4.1 Perancangan Antarmuka Nilai H_{init} *Testing*

Pada rancangan ini menampilkan gambar hasil dari perhitungan H_{init} *testing*. Berikut antarmuka halaman nilai H_{init} *testing* dapat dilihat pada Gambar 4.25.



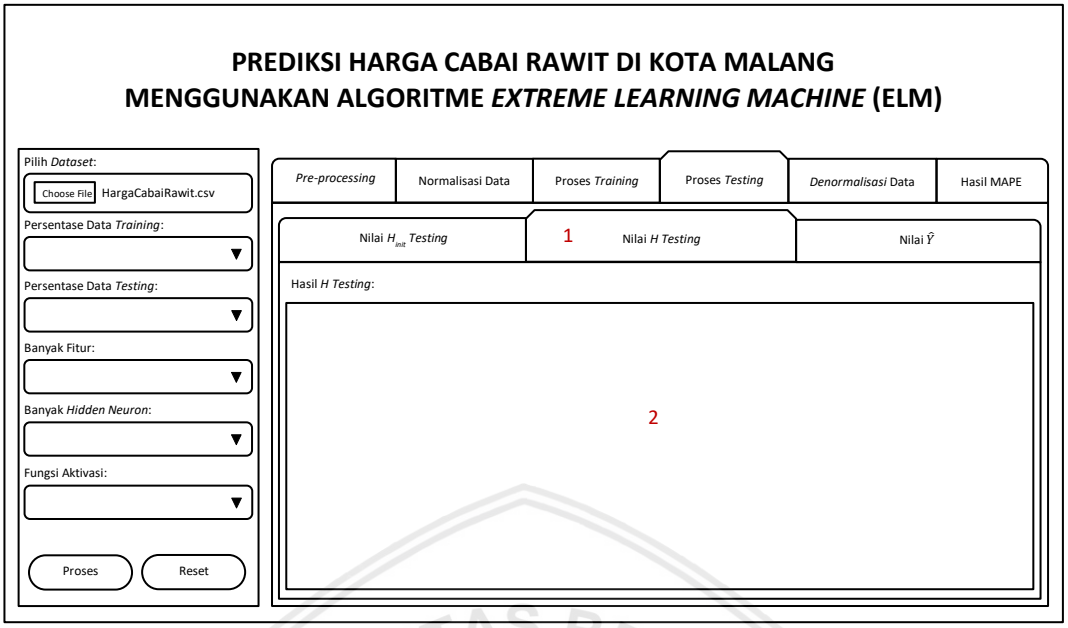
Gambar 4.25 Perancangan Antarmuka Halaman Nilai H_{init} *Testing*

Keterangan:

1. *Tab* Proses *Testing* berfungsi menampilkan halaman proses *testing* data.
2. *Tab* Nilai Bobot berfungsi untuk menampilkan halaman dari hasil nilai bobot.
3. Tabel untuk menampilkan hasil dari proses perhitungan H_{init} *testing*.

4.4.4.2 Perancangan Antarmuka Nilai *H Testing*

Pada rancangan ini sebagai tampilan untuk perhitungan *H* pada proses *testing*. Berikut antarmuka halaman nilai H_{init} *testing* dapat dilihat pada Gambar 4.26.



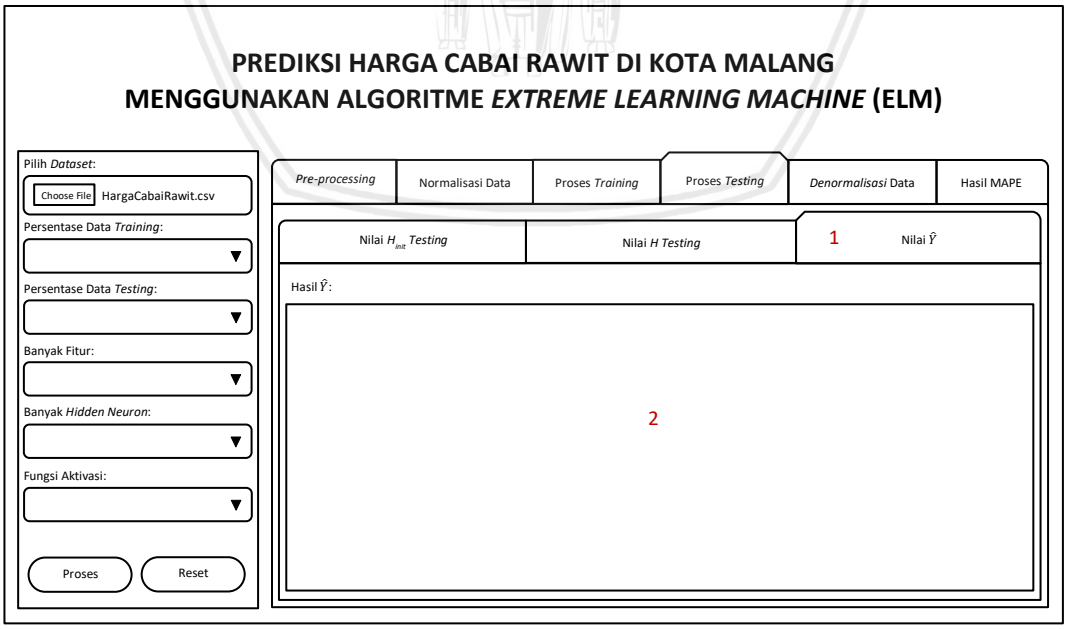
Gambar 4.26 Perancangan Antarmuka Halaman Nilai *H Testing*

Keterangan:

1. *Tab* Nilai *H Testing* berfungsi untuk menampilkan halaman dari hasil perhitungan nilai *H* pada proses *testing*.
2. Tabel sebagai tampilan untuk hasil dari proses perhitungan *H testing*.

4.4.4.3 Perancangan Antarmuka Nilai $\hat{Y}$

Pada rancangan ini sebagai tampilan untuk perhitungan nilai $\hat{Y}$ pada proses *testing*. Berikut antarmuka halaman nilai $\hat{Y}$ terdapat pada Gambar 4.27.



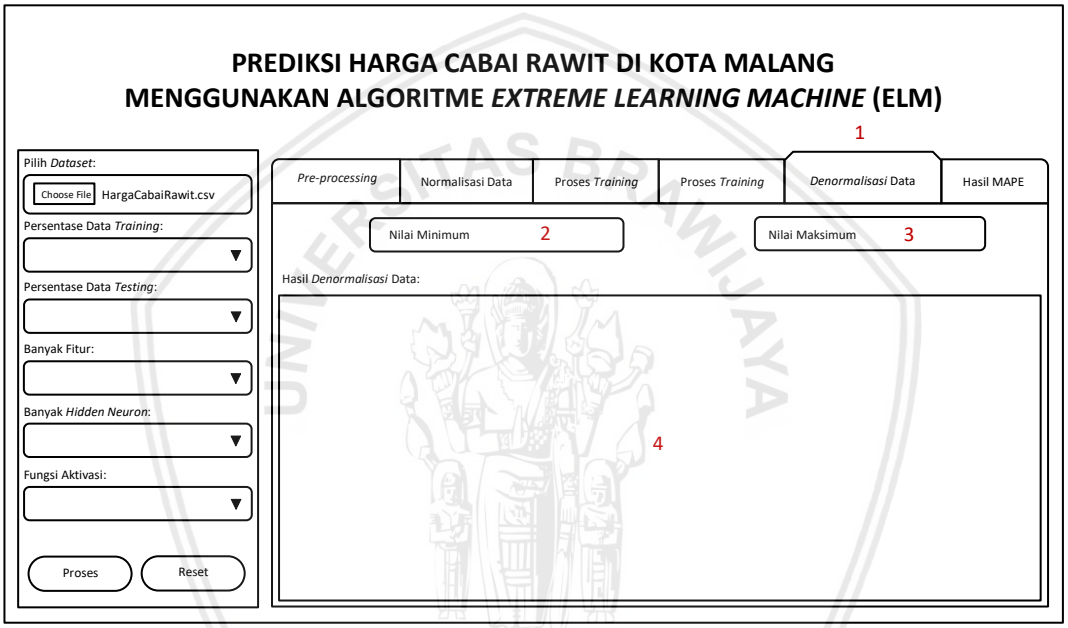
Gambar 4.27 Perancangan Antarmuka Halaman Nilai $\hat{Y}$

Keterangan:

1. *Tab* Nilai $\hat{Y}$ berfungsi untuk menampilkan halaman dari hasil perhitungan nilai $\hat{Y}$ pada bagian untuk proses *testing*.
2. Tabel sebagai tampilan untuk hasil dari proses perhitungan $\hat{Y}$.

4.4.5 Perancangan Antarmuka Halaman *Denormalisasi* Data

Pada halaman rancangan antarmuka ini akan menampilkan rancangan untuk halaman *denormalisasi* data. Dalam rancangan tampilan *denormalisasi* data dibutuhkan form input untuk nilai minimum dan maksimum serta tabel untuk menampilkan hasil perhitungan *denormalisasi*. Berikut rancangan antarmuka untuk halaman *denormalisasi* data terdapat pada Gambar 4.28.



Gambar 4.28 Perancangan Antarmuka Halaman *Denormalisasi* Data

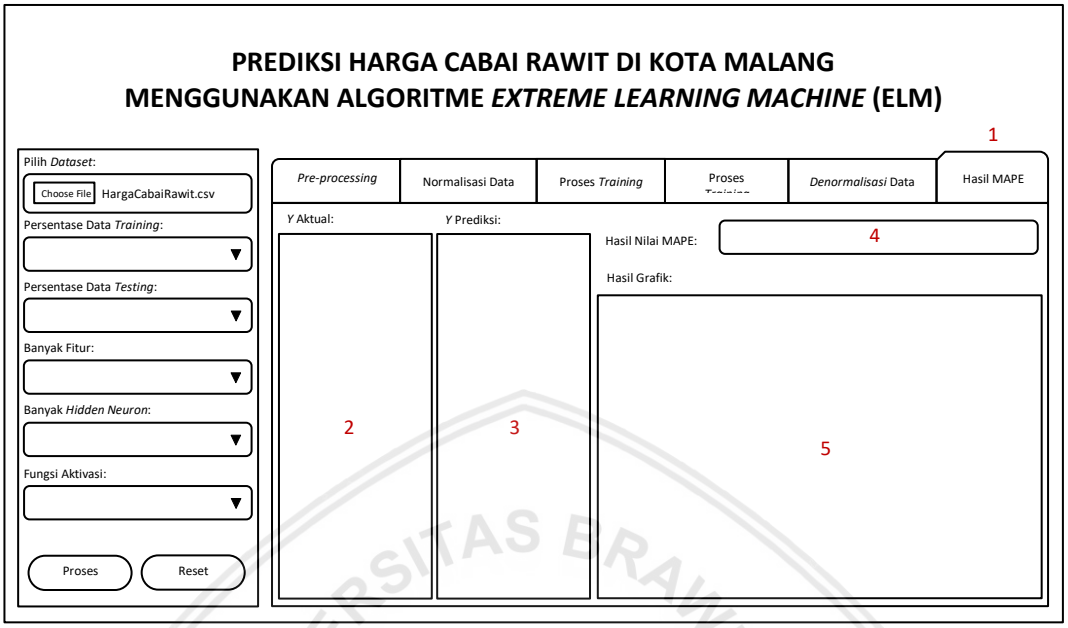
Keterangan:

1. *Tab Denormalisasi* Data berfungsi menampilkan halaman hasil dari perhitungan *denormalisasi* data.
2. *Form* nilai minimum berfungsi untuk menampilkan nilai minimum yang digunakan.
3. *Form* nilai maksimum berfungsi untuk menampilkan nilai maksimum yang digunakan.
4. Tabel hasil *Denormalisasi* Data yang berfungsi untuk menampilkan tabel hasil *denormalisasi* data.

4.4.6 Perancangan Antarmuka Halaman Hasil MAPE

Pada halaman rancangan ini akan menampilkan rancangan untuk halaman hasil perhitungan MAPE. Dalam rancangan tampilan hasil MAPE dibutuhkan form input untuk menampilkan hasil akhir dari nilai MAPE, tabel untuk menampilkan

data aktual, data prediksi dan grafik garis. Berikut rancangan antarmuka untuk halaman hasil MAPE terdapat pada bagian Gambar 4.29.



Gambar 4.29 Perancangan Antarmuka Halaman Hasil MAPE

Keterangan:

1. *Tab* Hasil MAPE berfungsi menampilkan halaman hasil dari perhitungan evaluasi prediksi menggunakan *Mean Absolute Percentage Error* (MAPE).
2. Tabel *Y* aktual berfungsi sebagai menampilkan nilai aktual berdasarkan nilai *Y* dari data *testing*.
3. Tabel *Y* prediksi berfungsi sebagai menampilkan nilai prediksi berdasarkan nilai *Y* dari hasil proses *testing* setelah dilakukan *denormalisasi*.
4. *Form* input yang berfungsi untuk menampilkan nilai akhir dari hasil perhitungan antara *Y* aktual dan *Y* prediksi.
5. Grafik garis yang berfungsi untuk menampilkan gambaran grafik garis antara *Y* aktual dengan *Y* prediksi.

4.5 Perancangan Pengujian Algoritme

Pada perancangan untuk pengujian yang akan dilakukan untuk algoritme *Extreme Learning Machine* (ELM) bertujuan agar dapat mengetahui nilai parameter yang optimal dalam prediksi harga cabai rawit di Kota Malang. Perancangan yang digunakan dalam pengujian nilai parameter pada algoritme ELM sebagai berikut.

1. Pengujian Banyak Fitur
2. Pengujian Persentase Banyak Data *Training* dan Data *Testing*.
3. Pengujian Banyak *Neuron* pada *Hidden Layer*.
4. Pengujian Fungsi Aktivasi

Dari skenario pengujian di atas, proses pengujian dilakukan secara berulang berdasarkan nilai parameter yang terbaik yang didapatkan sebelumnya. Pada setiap nilai parameter pengujian akan dibuat percobaan sebanyak 10 percobaan. Percobaan dilakukan sebanyak 10 kali dikarenakan angka tersebut sudah dapat mewakili jumlah percobaan yang dapat dilakukan karena pada algoritme ELM hasil yang diperoleh dari setiap hasil prediksi selalu berbeda. Sehingga dalam menaggulangi nilai yang dihasilkan selalu berbeda, maka diperlukan percobaan sebanyak 10 dengan harapan hasil akhir yang diperoleh dapat diambil dari nilai rata-rata hasil evaluasi MAPE.

4.5.1 Pengujian Banyak Fitur

Pada bagian pengujian banyak fitur ini dilakukan untuk mengetahui nilai fitur berdasarkan hari sebelumnya yang menghasilkan nilai yang optimal. Nilai yang digunakan dalam parameter banyak fitur adalah 2 sampai 10. Dari nilai tersebut akan diketahui apakah pengujian pada banyak fitur dapat memengaruhi nilai MAPE. Nilai akhir MAPE ditentukan dari rata-rata MAPE yang menghasilkan nilai terkecil dari keseluruhan nilai yang diujikan. Perancangan pengujian untuk banyak fitur dapat dilihat pada bagian Tabel 4.22.

Tabel 4.22 Perancangan Pengujian Banyak Fitur

Banyak Fitur	MAPE Percobaan ke-										Rata-rata MAPE
	1	2	3	4	5	6	7	8	9	10	
2											
3											
4											
5											
6											
7											
8											
9											
10											

4.5.2 Pengujian Persentase Banyak Data *Training* dan Data *Testing*

Pada bagian pengujian ini dilakukan perbandingan persentase antara banyak data *training* dan data *testing*. Pengujian dimaksudkan untuk mengetahui nilai dari pengaruh persentase perbandingan data yang menghasilkan MAPE terkecil. Pengujian perbandingan banyak data dilakukan sebanyak 9 variasi yaitu dengan perbandingan 90%:10%, 80%:20%, 70%:30%, 60%:40%, 50%:50%, 40%:60%, 30%:70%, dan 20%:80%. Perancangan pengujian ini dapat dilihat pada bagian Tabel 4.23.

Tabel 4.23 Parancangan Pengujian Persentase Banyak Data *Training* dan Data *Testing*

Data *Training* : Data *Testing*	MAPE Percobaan ke-										Rata-rata MAPE
	1	2	3	4	5	6	7	8	9	10	
90%:10%											
80%:20%											
70%:30%											
60%:40%											
50%:50%											
40%:60%											
30%:70%											
20%:80%											

4.5.3 Pengujian Banyak *Neuron* pada *Hidden Layer*

Pada bagian pengujian ini bertujuan untuk mengetahui pengaruh yang dihasilkan dari perubahan banyak *neuron* yang terdapat di arsitektur *hidden layer* terhadap MAPE dan mengetahui berapa banyak *neuron* yang paling optimal. Nilai pengujian yang digunakan antara 2 sampai 10. Nilai tersebut merupakan nilai yang akan dijadikan input pada parameter *hidden layer*. Perancangan pengujian untuk banyak *neuron* pada *hidden layer* terdapat pada bagian Tabel 4.24.

Tabel 4.24 Perancangan Pengujian Banyak *Neuron* pada *Hidden Layer*

Banyak *Neuron*	MAPE Percobaan ke-										Rata-rata MAPE
	1	2	3	4	5	6	7	8	9	10	
2											
3											
4											
5											
6											
7											
8											
9											
10											

4.5.4 Pengujian Fungsi Aktivasi

Pada pengujian ini akan dilakukan uji untuk bagian saat menghitung keluaran *hidden layer* (*H*) dengan menggunakan fungsi aktivasi yang berbeda. Pengujian ini bertujuan untuk mengetahui apakah terdapat perbedaan hasil MAPE terhadap jenis fungsi aktivasi yang digunakan. Terdapat beberapa fungsi aktivasi yang akan digunakan ada 4 yaitu *sigmoid biner*, *sin*, *sigmoid bipolar*, dan *radial basis*. Perancangan untuk pengujian fungsi aktivasi terdapat pada Tabel 4.25.

Tabel 4.25 Perancangan Pengujian Fungsi Aktivasi

Fungsi Aktivasi	MAPE Percobaan ke-										Rata-rata MAPE
	1	2	3	4	5	6	7	8	9	10	
Sigmoid Biner											
Sin											
Sigmoid Bipolar											
Radial Basis											

BAB 5 IMPLEMENTASI

5.1 Implementasi Sistem

Pada bagian bab ini akan dilakukan penerapan dari perancangan sesuai dengan yang telah dibuat sebelumnya di rancangan algoritme *Extreme Learning Machine*. Dalam implementasi ini akan dibagi menjadi beberapa proses implementasi menjadi kode program algoritme ELM yaitu kode program *pre-processing*, kode program untuk normalisasi data, kode program untuk proses *training*, kode program unruk proses *testing*, kode program proses *denormalisasi* data, dan kode program menghitung hasil nilai MAPE. Pada setiap proses akan memiliki sub proses kode program sesuai dengan perancangan algoritme ELM.

5.1.1 Implementasi Algoritme ELM

Pada bagian proses implementasi ini merupakan penjelasan kode program dalam menjalankan algoritme ELM secara umum dari inisialisasi nilai awal yang dibutuhkan seperti data harga cabai rawit, banyak fitur, persentase data *training*, persentase data *testing*, banyak *neuron* pada bagian *hidden layer*, nilai data minimum, nilai data maksimum, dan fungsi aktivasi, serta proses *pre-processing*, normalisasi data, proses *training*, proses *testing*, *denormalisasi* data, dan menghitung hasil nilai MAPE. Kode program untuk implementasi fungsi main terdapat pada Kode Program 5.1.

Algoritme 1: Fungsi Main

```
1  def main(request):
2      if request.method == 'POST':
3          dataset = request.FILES['inputDataset']
4          persentase_data_training =
5          int(request.POST['persentaseDataTraining'])
6          persentase_data_testing =
7          int(request.POST['persentaseDataTesting'])
8          banyak_fitur = int(request.POST['banyakFitur'])
9          banyak_hidden_neuron =
10         int(request.POST['banyakHiddenNeuron'])
11         fungsi_aktivasi = request.POST['fungsiAktivasi']
12     minimum = 5000
13     maksimum = 150000
14     data = preprocessing_data(dataset, banyak_fitur)
15     data_training =
16     data[:int(persentase_data_training*len(data)/100)]
17     data_testing =
18     data[int(persentase_data_training*len(data)/100):]
19     x_training = proses_normalisasi(data_training, minimum,
20     maksimum)
21     x_testing = proses_normalisasi(data_testing, minimum, maksimum)
22     bobot, bias, betatopi = proses_training(x_training,
23     banyak_fitur, fungsi_aktivasi, banyak_hidden_neuron)
24     ytopi = proses_testing(x_testing, fungsi_aktivasi, bobot, bias,
25     betatopi)
26     denormalisasi = proses_denormalisasi(ytopi, minimum, maksimum)
27     mape = proses_mape(denormalisasi, data_testing, banyak_fitur)
28     return render(request, 'hasil.html', {
29         'data_training' : data_training,
30         'data_testing' : data_testing,
31         'banyak_fitur' : banyak_fitur,
32         'minimum' : minimum,
```

```
33          'maksimum' : maksimum,
34          'x_training' : x_training,
35          'x_testing' : x_testing,
36          'bobot' : bobot,
37          'bias' : bias,
38          'hinit_training' : hinit_training,
39          'h_training' : h_training,
40          'h_transposexh' : h_transposexh,
41          'h_transposexh_inversi' : h_transposexh_inversi,
42          'h_plus' : h_plus,
43          'beta' : betatopi,
44          'hinit_testing' : hinit_testing,
45          'h_testing' : h_testing,
46          'y_aktual' : [data_testing[i][banyak_fitur] for i in
47          range(len(data_testing))],
48          'y_prediksi' : ytopi,
49          'denormalisasi_data' : denormalisasi,
50          'mape' : mape
51          })
```

Kode Program 5.1 Algoritme ELM

Penjelasan untuk implementasi Kode Program 5.1 dijabarkan sebagai berikut:

1. Baris 1 menjelaskan membuat fungsi dengan nama *main*.
2. Baris 2-11 menjelaskan pengecekan kondisi jika *method* untuk *request* adalah *POST* maka akan melakukan proses menyimpan nilai masukan untuk *dataset*, persentase banyak dari data *training*, persentase banyak dari data *testing*, banyak fitur, banyak untuk *hidden neuron*, dan penggunaan fungsi aktivasi.
3. Baris 12-13 menjelaskan inisialisasi nilai minimum sebesar 5000 dan nilai maksimum sebesar 150000.
4. Baris 14 menjelaskan pemanggilkan fungsi *pre-processing* dengan nilai *arguments* yang diberikan adalah nilai *dataset* dan banyak fitur.
5. Baris 15-18 menjelaskan pembagian data untuk data *training* dan data *testing*.
6. Baris 19-21 menjelaskan proses pemanggilan fungsi normalisasi pada setiap data *training* dan data *testing*.
7. Baris 22-23 menjelaskan pemanggilan fungsi proses *training* data dengan *arguments* nilai data *training* hasil normalisasi, banyak fitur, fungsi aktivasi, dan banyak *hidden neuron*.
8. Baris 24-25 menjelaskan pemanggilan fungsi proses *testing* data dengan *arguments* masukan yaitu nilai data *testing* hasil dari normalisasi, fungsi aktivasi, nilai bobot, nilai *bias*, dan hasil pada nilai $\hat{\beta}$ (*beta topi*) yang diperoleh dari hasil proses *training*.
9. Baris 26 menjelaskan pemanggilan fungsi proses *denormalisasi* data dengan *arguments* masukan yaitu nilai hasil proses testing, nilai minimum, dan maksimum.
10. Baris 27 menjelaskan pemanggilan fungsi proses menghitung nilai MAPE dengan *arguments* masukan data *denormalisasi*, data *testing* sebelum normalisasi, dan banyak fitur.
11. Baris 28-51 menjelaskan pengembalian data menggunakan fungsi *render* untuk ditampilkan pada halaman hasil.*html.*

5.1.2 Implementasi *Pre-processing* Data

Pada bagian implementasi *pre-processing* data merupakan proses untuk membaca data dan mengubah data menjadi bentuk fitur berdasarkan hari sebelumnya menggunakan metode analisis teknikal. Kode program untuk implementasi fungsi *pre-processing* data terdapat pada Kode Program 5.2.

Algoritme 2: Fungsi *Pre-processing* Data

```
1  def preprocessing_data(dataset, banyak_fitur):
2      dataset = pd.read_csv(dataset, delimiter=';', names =
3      ['Tanggal', 'Harga (Rupiah)'], usecols=['Harga (Rupiah)'])
4      banyak_fitur += 1
5      hasil_fitur = []
6      for i in range((len(dataset)-banyak_fitur)+1):
7          kolom = []
8          j = i
9          while j < (i+banyak_fitur):
10             kolom.append(dataset.values[j][0])
11             j += 1
12         hasil_fitur.append(kolom)
13     return hasil_fitur
```

Kode Program 5.2 *Pre-processing* Data

Penjelasan untuk implementasi Kode Program 5.2 dijabarkan sebagai berikut:

1. Baris 1 menjelaskan deklarasi membuat fungsi *preprocessing_data* dengan parameter *dataset* sebagai data harga cabai rawit dan banyak_fitur.
2. Baris 2-3 menjelaskan proses membaca file *dataset* menggunakan *library pandas* untuk *file* dengan format *csv*.
3. Baris 4-5 menjelaskan penambahan angka 1 pada variabel banyak_fitur dan inisialisasi variabel hasil_fitur dengan nilai *list* kosong.
4. Baris 6-13 menjelaskan proses membuat fitur dari *dataset* berdasarkan banyak fitur yang ditentukan.

5.1.3 Implementasi Normalisasi Data

Pada bagian implementasi normalisasi dari data secara proses bertujuan untuk mengubah data menggunakan nilai minimum dan maksimum untuk diubah dalam rentang nilai antara [0,1]. Untuk implementasi fungsi normalisasi data terdapat pada Kode Program 5.3.

Algoritme 3: Fungsi Normalisasi Data

```
1  def proses_normalisasi(data, minimum, maksimum):
2      hasil_normalisasi = []
3      for i in range(len(data)):
4          temp_normalisasi = []
5          for j in range(len(data[i])):
6              temp_normalisasi.append((data[i][j]-minimum)/(maksimum-
7              minimum))
8          hasil_normalisasi.append(temp_normalisasi)
9      return hasil_normalisasi
```

Kode Program 5.3 Normalisasi Data

Penjelasan untuk implementasi Kode Program 5.3 dijabarkan sebagai berikut:

1. Baris 1 menjelaskan deklarasi fungsi dari proses_normalisasi dengan parameter nilai data, nilai minimum, dan nilai maksimum, nilai data

disesuaikan dengan penggunaan fungsi jika pada bagian proses *training* maka menggunakan nilai dari data *training* hasil normalisasi dan jika pada bagian proses *testing* maka menggunakan nilai dari data *testing* hasil normalisasi.
2. Baris 2-4 menjelaskan inisialisasi nilai untuk proses perulangan pertama dan kedua dengan nilai *list* kosong.
3. Baris 5-7 menjelaskan proses perhitungan normalisasi untuk data yang disimpan pada nilai temp_normalisasi.
4. Baris 8-9 menjelaskan menyimpan nilai hasil temp_normalisasi ke dalam hasil_normalisasi dan mengembalikan hasil dari hasil_normalisasi.

5.1.4 Implementasi Proses *Training*

Pada bagian implementasi proses *training* merupakan proses melatih algoritme agar dapat memiliki pengetahuan dengan memberikan nilai data *training* sebagai data masukan. Kode program untuk implementasi fungsi proses *training* terdapat pada Kode Program 5.4.

Algoritme 4: Fungsi Proses *Training*

```
1  def proses_training(x_training, banyak_fitur, fungsi_aktivasi,
2  banyak_hidden_neuron):
3      global hinit_training, h_training, h_plus
4      w, bias = hitung_bobot_bias(banyak_hidden_neuron, banyak_fitur)
5      hinit_training = hitung_hinit(x_training, w, bias)
6      h_training = hitung_keluaran_hidden_layer(hinit_training,
7      fungsi_aktivasi)
8      h_plus = hitung_moore_penrose_generalized_inverse(h_training)
9      beta_topi = output_weight(h_plus, x_training, banyak_fitur)
10     return w, bias, beta_topi
```

Kode Program 5.4 Proses *Training*

Penjelasan untuk implementasi Kode Program 5.4 dijabarkan sebagai berikut:

1. Baris 1-3 menjelaskan deklarasi fungsi *proses_training* dengan parameter data *training* setelah normalisasi, banyak_fitur, fungsi_aktivasi yang digunakan, dan banyak *hidden neuron* pada *hidden layer*.
2. Baris 4 menjelaskan proses pembuatan nilai bobot dan *bias* yang didapatkan dengan cara *random* pada *arguments* banyak *hidden neuron* dan banyak fitur.
3. Baris 5 menjelaskan proses menghitung H_{init} dengan nilai *arguments* data *training*, bobot, dan *bias*.
4. Baris 6-7 menjelaskan proses untuk menghitung nilai pada bagian keluaran *hidden layer* dengan menghasilkan nilai *H training* dengan nilai *arguments* data hasil menghitung H_{init} *training* dan fungsi aktivasi yang digunakan.
5. Baris 8 menjelaskan proses menghitung nilai H^+ dari nilai *H training*.
6. Baris 9-10 menjelaskan proses menghitung nilai $\hat{\beta}$ dengan nilai *arguments* yaitu nilai H^+, data *training*, dan banyak fitur, kemudian mengembalikan nilai bobot, *bias*, dan nilai $\hat{\beta}$.

5.1.5 Implementasi Bobot dan *Bias*

Pada bagian implementasi nilai bobot dan *bias* dilakukan dengan cara *random* menggunakan distribusi *uniform*. Data bobot dilakukan dengan *random*

dalam rentang [-1,1], sedangkan data *bias* dalam rentang [0,1]. Kode program implementasi fungsi hitung bobot *bias* terdapat pada bagian Kode Program 5.5.

Algoritme 5: Fungsi Hitung Bobot dan *Bias*

```
1  def hitung_bobot_bias(banyak_hidden_neuron, banyak_fitur):
2      w = []
3      bias = []
4      for i in range(banyak_hidden_neuron):
5          temp_w = []
6          for j in range(banyak_fitur):
7              temp_w.append(rd.uniform(-1,1))
8          w.append(temp_w)
9          bias.append(rd.uniform(0,1))
10     return w, bias
```

Kode Program 5.5 Bobot dan *Bias*

Penjelasan untuk implementasi Kode Program 5.5 dijabarkan sebagai berikut:

1. Baris 1 menjelaskan deklarasi fungsi hitung_bobot_bias dengan parameter yaitu nilai banyak untuk *neuron* pada *hidden layer* dan banyak fitur.
2. Baris 2-8 menjelaskan pembuatan variabel *w* dan *bias* dengan nilai *list* kosong, setelah itu membuat nilai bobot secara *random* menggunakan *library random* dengan fungsi *uniform* antara [-1,1].
3. Baris 9-10 menjelaskan pembuatan nilai *random bias* menggunakan fungsi *uniform* dengan rentang antara [0,1] dan mengembalikan nilai *w* dan *bias*.

5.1.6 Implementasi Hitung H_{init}

Pada bagian implementasi fungsi ini bertujuan untuk mendapatkan hasil dari nilai H_{init} pada proses *training* dan *testing*. Jika fungsi hitung H_{init} dipanggil pada bagian *training* maka akan menggunakan nilai dari data *training*, sedangkan jika dipanggil pada bagian *testing* maka akan menggunakan nilai dari data *testing*. Kode program implementasi fungsi hitung H_{init} terdapat pada Kode Program 5.6.

Algoritme 6: Fungsi Hitung H_{init}

```
1  def hitung_hinit(x, w, bias):
2      w_transpose = transposisi_matriks(w)
3      hasil_hinit = perkalian_matriks(x, w_transpose)
4      for i in range(len(hasil_hinit)):
5          for j in range(len(hasil_hinit[i])):
6              hasil_hinit[i][j] += bias[j]
7      return hasil_hinit
```

Kode Program 5.6 H_{init}

Penjelasan untuk implementasi Kode Program 5.6 dijabarkan sebagai berikut:

1. Baris 1 menjelaskan deklarasi fungsi hitung_hinit dengan parameter yaitu nilai *x* untuk nilai dari data *training* atau *testing*, nilai *w* untuk bobot, dan *bias*.
2. Baris 2 menjelaskan proses transposisi nilai *w*.
3. Baris 3 menjelaskan proses perkalian matriks antara nilai *x* dan *w* hasil transposisi.
4. Baris 4-7 menjelaskan proses penambahan nilai *bias* dari nilai hasil perhitungan H_{init} kemudian mengembalikan nilai hasil H_{init}.

5.1.7 Implementasi Transposisi Matriks

Pada bagian implementasi fungsi transposisi matriks bertujuan untuk melakukan pemindahan posisi dari matriks. Kode program dari implementasi fungsi transposisi matriks terdapat pada Kode Program 5.7.

```
Algoritme 7: Fungsi Transposisi Matriks
1  def transposisi_matriks(matriks):
2      hasil_transposisi = []
3      for i in range(len(matriks[0])):
4          temp_transposisi = []
5          for j in range(len(matriks)):
6              temp_transposisi.append(matriks[j][i])
7          hasil_transposisi.append(temp_transposisi)
8      return hasil_transposisi
```

Kode Program 5.7 Transposisi Matriks

Penjelasan untuk implementasi Kode Program 5.7 dijabarkan sebagai berikut:

1. Baris 1 menjelaskan deklarasi fungsi dengan nama transposisi_matriks dengan nilai input pada parameter yaitu nilai *matriks*.
2. Baris 2-4 menjelaskan inisialisasi dari pembuatan variabel dengan nama hasil_transposisi dan temp_transposisi dengan nilai *list* kosong, kemudian dilakukan perulangan sebanyak jumlah kolom *matriks*.
3. Baris 5-8 menjelaskan pengambilan nilai indeks baris ke-*j* dan kolom ke-*i* sebanyak baris *matriks* dan mengembalikan nilai hasil transposisi matriks.

5.1.8 Implementasi Perkalian Matriks

Pada bagian implementasi akan dilakukan proses dari perkalian antara dua matriks yang berbeda dan menghasilkan matriks hasil perkalian. Kode program implementasi fungsi perkalian matriks terdapat pada bagian Kode Program 5.8.

```
Algoritme 8: Fungsi Perkalian Matriks
1  def perkalian_matriks(matriks1, matriks2):
2      hasil_perkalian = []
3      for k in range(len(matriks1)):
4          temp_perkalian = []
5          for l in range(len(matriks2[0])):
6              perkalian = 0
7              for m in range(len(matriks2)):
8                  perkalian += (matriks1[k][m]*matriks2[m][l])
9              temp_perkalian.append(perkalian)
10         hasil_perkalian.append(temp_perkalian)
11     return hasil_perkalian
```

Kode Program 5.8 Perkalian Matriks

Penjelasan untuk implementasi Kode Program 5.8 dijabarkan sebagai berikut:

1. Baris 1 menjelaskan deklarasi fungsi perkalian_matriks dengan parameter yaitu nilai matriks1 dan matriks2.
2. Baris 2-3 menjelaskan inisialisasi variabel *list* kosong untuk menyimpan nilai hasil perulangan sebanyak jumlah matriks1.
3. Baris 4-5 menjelaskan inisialisasi variabel *list* kosong untuk menyimpan nilai hasil perulangan sebanyak kolom matriks2.

4. Baris 6-9 menjelaskan inisialisasi variabel bernilai 0 untuk menyimpan nilai hasil perulangan sebanyak baris matriks2.
5. Baris 10-11 menjelaskan menyimpan nilai hasil perkalian matriks kemudian mengembalikannya.

5.1.9 Implementasi Keluaran *Hidden Layer*

Pada bagian implementasi akan dilakukan proses implementasi untuk mengubah data dari hasil perhitungan H_{init} dengan menggunakan fungsi aktivasi antara lain yaitu fungsi *Sigmoid Biner*, fungsi *Sin*, fungsi *Sigmoid Bipolar*, dan fungsi *Radial Basis* yang akan menghasilkan nilai *H*. Kode program implementasi fungsi untuk mendapatkan nilai *H* terdapat pada bagian Kode Program 5.9.

Algoritme 9: Fungsi Hitung Keluaran *Hidden Layer*

```
1  def hitung_keluaran_hidden_layer(hinit, fungsi_aktivasi):
2      hasil_h = []
3      for i in range(len(hinit)):
4          temp_h = []
5          for j in range(len(hinit[i])):
6              if fungsi_aktivasi == 'Sigmoid Biner':
7                  h = 1/(1 + math.exp(-hinit[i][j]))
8              elif fungsi_aktivasi == 'Sin':
9                  h = math.sin(hinit[i][j])
10             elif fungsi_aktivasi == 'Sigmoid Bipolar':
11                 h = (1 - math.exp(-hinit[i][j])) / (1 + math.exp(-
12                 hinit[i][j]))
13             else:
14                 h = math.exp(-(hinit[i][j]**2))
15             temp_h.append(h)
16         hasil_h.append(temp_h)
17     return hasil_h
```

Kode Program 5.9 Keluaran *Hidden Layer*

Penjelasan untuk implementasi Kode Program 5.9 dijabarkan sebagai berikut:

1. Baris 1 menjelaskan deklarasi fungsi hitung_keluaran_hidden_layer dengan parameter yaitu nilai H_{init} (jika fungsi dipanggil di proses *training* maka menggunakan nilai H_{init} *training*, sedangkan jika dipanggil pada proses *testing* maka menggunakan nilai H_{init} *testing*) dan fungsi aktivasi.
2. Baris 2-3 menjelaskan inisialisasi variabel *list* kosong dengan nama hasil_h dan melakukan perulangan sebanyak panjang dari H_{init} kemudian membuat *list* kosong kembali dengan nama temp_h.
3. Baris 4-15 menjelaskan inisialisasi variabel *temporary* dengan nilai *list* kosong untuk digunakan sebagai nilai yang menyimpan hasil perhitungan rumus fungsi aktivasi yang digunakan dalam perulangan sebanyak panjang baris H_{init}.
4. Baris 16-17 menjelaskan menyimpan nilai *temporary* ke dalam nilai hasil *H* dan mengembalikannya.

5.1.10 Implementasi *Moore-Penrose Generalized Inverse*

Pada bagian implementasi *Moore-Penrose Generalized Inverse* akan dilakukan proses untuk mencari nilai H^+ dengan nilai parameter *H training*. Kode program untuk implementasi fungsi *Moore-Penrose Generalized Inverse* terdapat pada bagian Kode Program 5.10.

```
Algoritme 10: Fungsi Hitung Moore-Penrose Generalized Inverse
1  def hitung_moore_penrose_generalized_inverse(h_training):
2      global h_transposexh, h_transposexh_inversi
3      h_transpose = transposisi_matriks(h_training)
4      h_transposexh = perkalian_matriks(h_transpose, h_training)
5      h_transposexh_inversi = inversi_matriks(h_transposexh)
6      hasil_h_plus = perkalian_matriks(h_transposexh_inversi,
7      h_transpose)
8      return hasil_h_plus
```

Kode Program 5.10 *Moore-Penrose Generalized Inverse*

Penjelasan untuk implementasi Kode Program 5.10 sebagai berikut:

1. Baris 1-2 menjelaskan deklarasi fungsi hitung_moore_penrose_generalized _inverse dengan nilai parameter adalah *H training*.
2. Baris 3 menjelaskan perhitungan fungsi transposisi dari matriks *H training*.
3. Baris 4 menjelaskan perhitungan perkalian matriks *H* transposisi dan *H*.
4. Baris 5 menjelaskan pemanggilan fungsi menghitung nilai inversi matriks dengan nilai *arguments* yaitu hasil perkalian *H* transposisi dan *H*.
5. Baris 6-8 menjelaskan proses menghitung dan mengembalikan nilai H^+ dengan nilai *arguments* yaitu hasil inversi matriks dan *H* transposisi.

5.1.11 Implementasi Inversi Matriks

Pada bagian implementasi akan dilakukan sebuah proses inversi menggunakan metode Operasi Baris Elementer (OBE). Kode program untuk implementasi fungsi inversi matriks terdapat pada Kode Program 5.11.

```
Algoritme 11: Fungsi Inversi Matriks
1  def inversi_matriks(matriks):
2      hasil_inversi = []
3      for i in range(len(matriks)):
4          temp_inversi = []
5          for j in range(2*len(matriks[i])):
6              if j < len(matriks[i]):
7                  temp_inversi.append(matriks[i][j])
8              elif i == j-len(matriks[i]):
9                  temp_inversi.append(1)
10             else:
11                 temp_inversi.append(0)
12         hasil_inversi.append(temp_inversi)
13     for k in range(len(hasil_inversi)):
14         kunci1 = hasil_inversi[k][k]
15         for l in range(len(hasil_inversi[k])):
16             hasil_inversi[k][l] /= kunci1
17         for m in range(len(hasil_inversi)):
18             if m == k:
19                 continue
20             else:
21                 kunci2 = hasil_inversi[m][k]
22                 for n in range(len(hasil_inversi[k])):
23                     hasil_inversi[m][n] -=
24                     (kunci2*hasil_inversi[k][n])
25     for v in range(len(hasil_inversi)):
26         for u in range(int(len(hasil_inversi[v])/2)):
27             hasil_inversi[v].pop(0)
28     return hasil_inversi
```

Kode Program 5.11 Inversi Matriks

Penjelasan untuk implementasi Kode Program 5.11 sebagai berikut:

1. Baris 1 menjelaskan deklarasi fungsi dengan nama inversi_matriks yang memiliki parameter yaitu nilai matriks.
2. Baris 2-3 menjelaskan inisialisasi dari variabel *list* kosong untuk menyimpan nilai hasil matriks identitas dalam perulangan sebanyak panjang matriks.
3. Baris 4-12 menjelaskan inisialisasi variabel *temporary* untuk menyimpan nilai hasil penambahan nilai 0 dan 1 pada setiap baris kemudian simpan ke variabel hasil_inversi.
4. Baris 13-24 menjelaskan proses perhitungan nilai inversi menggunakan Operasi Baris Elementer.
5. Baris 25-28 menjelaskan proses menghapus nilai matriks identitas dan mengembalikan nilai hasil_inversi.

5.1.12 Implementasi *Output Weight*

Pada bagian implementasi akan dilakukan perhitungan nilai *output weight* untuk mencari hasil dari keluaran nilai pada proses *training*. Kode program untuk fungsi *output weight* terdapat pada bagian Kode Program 5.12.

Algoritme 12: Fungsi *Output Weight*

```
1  def output_weight(h_plus, x_training, banyak_fitur):
2      y = []
3      for i in range(len(x_training)):
4          y.append([x_training[i][banyak_fitur]])
5      hasil_beta_topi = perkalian_matriks(h_plus, y)
6      return hasil_beta_topi
```

Kode Program 5.12 *Output Weight*

Penjelasan untuk implementasi Kode Program 5.12 sebagai berikut:

1. Baris 1 menjelaskan deklarasi fungsi output_weight dengan parameter yaitu nilai H^+, data *training* setelah normalisasi, dan banyak fitur.
2. Baris 2-4 menjelaskan proses inisialisasi *list* kosong dan proses menyimpan nilai hasil *Y* sebagai matriks target pada data *training*.
3. Baris 5-6 menjelaskan proses menyimpan dan mengembalikan nilai hasil perkalian antara H^+ dan *Y*.

5.1.13 Implementasi Proses *Testing*

Pada bagian implementasi fungsi ini merupakan proses dalam pengujian terhadap algoritme ELM. Kode program untuk implementasi fungsi proses *testing* terdapat pada Kode Program 5.13.

Algoritme 13: Fungsi Proses T*esting*

```
1  def proses_testing(x_testing, fungsi_aktivasi, w, bias, beta_topi):
2      global hinit_testing, h_testing
3      hinit_testing = hitung_hinit(x_testing, w, bias)
4      h_testing = hitung_keluaran_hidden_layer(hinit_testing,
5      fungsi_aktivasi)
6      y_topi = perkalian_matriks(h_testing, beta_topi)
7      return y_topi
```

Kode Program 5.13 Proses *Testing*

Penjelasan untuk implementasi Kode Program 5.13 sebagai berikut:

1. Baris 1-2 menjelaskan deklarasi fungsi proses_testing dengan parameter data *testing* hasil normalisasi, fungsi aktivasi, bobot, *bias*, dan $\hat{\beta}$ dari hasil *training*.
2. Baris 3 menjelaskan pemanggilan fungsi untuk menghitung H_{init} dengan nilai *arguments* data *testing*, nilai bobot, dan nilai *bias*.
3. Baris 4-5 menjelaskan pemanggilan fungsi untuk menghitung keluaran *hidden layer* yang menghasilkan nilai *H testing* dengan nilai *arguments* data dari menghitung H_{init} *testing* dan fungsi aktivasi yang digunakan.
4. Baris 6-7 menjelaskan pemanggilan fungsi untuk menghitung dan mengembalikan hasil nilai $\hat{Y}$ dengan nilai *arguments* yaitu *H testing* dan $\hat{\beta}$.

5.1.14 Implementasi *Denormalisasi* Data

Pada bagian implementasi ini bertujuan untuk untuk mengubah nilai data hasil prediksi ke dalam data dengan rentang nilai minimal sebesar 5000 dan maksimal sebesar 150000. Kode program implementasi fungsi *denormalisasi* data terdapat pada bagian Kode Program 5.14.

Algoritme 14: Fungsi Proses *Denormalisasi*

```
1  def proses_denormalisasi(y_topi, minimum, maksimum):
2      hasil_denormalisasi = []
3      for i in range(len(y_topi)):
4          for j in range(len(y_topi[0])):
5              hasil_denormalisasi.append(y_topi[i][j] * (maksimum -
6              minimum) + minimum)
7      return hasil_denormalisasi
```

Kode Program 5.14 *Denormalisasi* Data

Penjelasan untuk implementasi Kode Program 5.14 sebagai berikut:

1. Baris 1 menjelaskan deklarasi fungsi proses_denormalisasi dengan parameter yaitu nilai $\hat{Y}$, minimum, dan maksimum.
2. Baris 2-3 menjelaskan inisialisasi *list* kosong untuk menyimpan nilai hasil *denormalisasi* data dengan melakukan perulangan sebanyak panjang $\hat{Y}$.
3. Baris 4-7 menjelaskan proses menghitung nilai hasil *denormalisasi* data dan mengembalikan nilai hasil perhitungan.

5.1.15 Implementasi Menghitung MAPE

Pada bagian implementasi ini akan dilakukan proses untuk mengetahui tingkat hasil kesalahan *error* dari data aktual dan data prediksi. Kode program implementasi fungsi proses MAPE terdapat pada bagian Kode Program 5.15.

Algoritme 15: Fungsi Proses MAPE

```
1  def proses_mape(y_prediksi, y_aktual, banyak_fitur):
2      hasil_mape = 0
3      for i in range(len(y_aktual)):
4          hasil_mape += abs(((y_aktual[i][banyak_fitur] -
5          y_prediksi[i]) / y_aktual[i][banyak_fitur]) * 100)
6      hasil_mape /= len(y_prediksi)
7      return hasil_mape
```

Kode Program 5.15 Menghitung MAPE

Penjelasan untuk implementasi Kode Program 5.15 dijabarkan sebagai berikut:

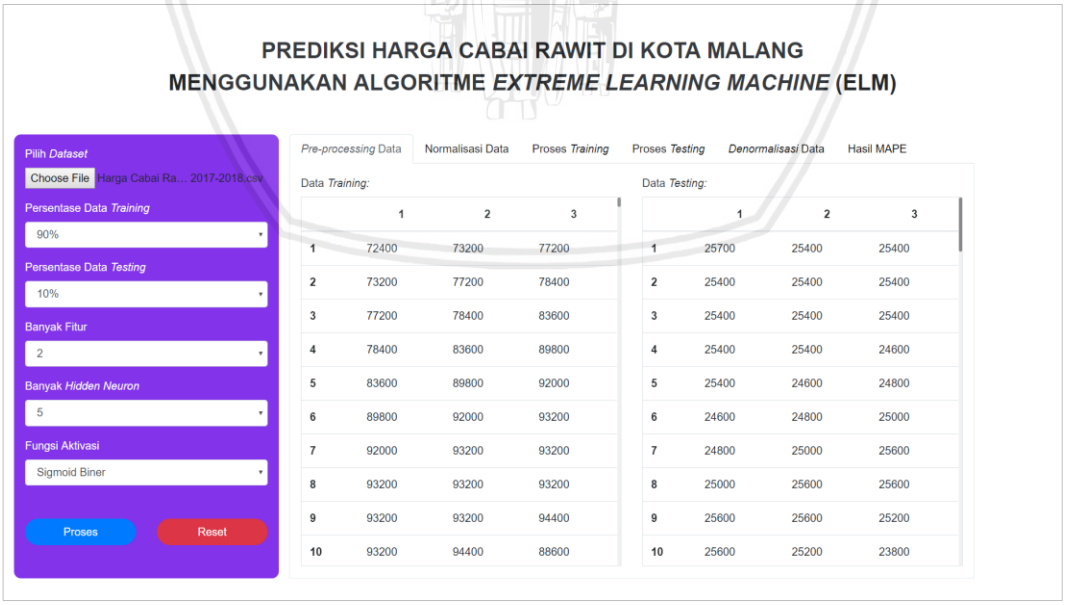
1. Baris 1 menjelaskan deklarasi fungsi proses_mape dengan parameter yaitu nilai data y_prediksi, data y_aktual, dan banyak fitur.
2. Baris 2 menjelaskan inisialisasi nilai 0 untuk menyimpan nilai hasil MAPE.
3. Baris 3-7 menjelaskan proses perhitungan nilai hasil MAPE sebanyak jumlah y_aktual dan pengembalian nilai hasil dari perhitungan MAPE.

5.2 Tampilan Sistem

Pada tampilan sistem akan dijelaskan tampilan dari sistem yang telah dilakukan implementasi dan berjalan pada *platform* website. Tampilan sistem terdapat terdapat 6 halaman untuk menampilkan halaman tampilan *pre-processing* data, halaman tampilan normalisasi data, halaman tampilan proses *training*, halaman tampilan proses *testing*, halaman tampilan *denormalisasi* data, dan halaman tampilan hasil MAPE. Pada halaman proses *training* terdapat 6 sub halaman untuk menampilkan nilai bobot, nilai *bias*, nilai H_{init} *training*, nilai *H training*, nilai H^{+}, dan nilai $\hat{\beta}$. Sedangkan pada halaman proses *testing* terdapat 3 sub halaman untuk menampilkan nilai H_{init} *testing*, nilai *H testing*, dan nilai $\hat{Y}$.

5.2.1 Tampilan *Pre-processing* Data

Pada bagian *pre-processing* data akan menampilkan nilai hasil *pre-processing* pada untuk data *training* dan data *testing*. Banyak fitur yang digunakan adalah 2 dan kolom berikutnya sebagai target. Hasil tampilan *pre-processing* data terdapat pada bagian Gambar 5.1.



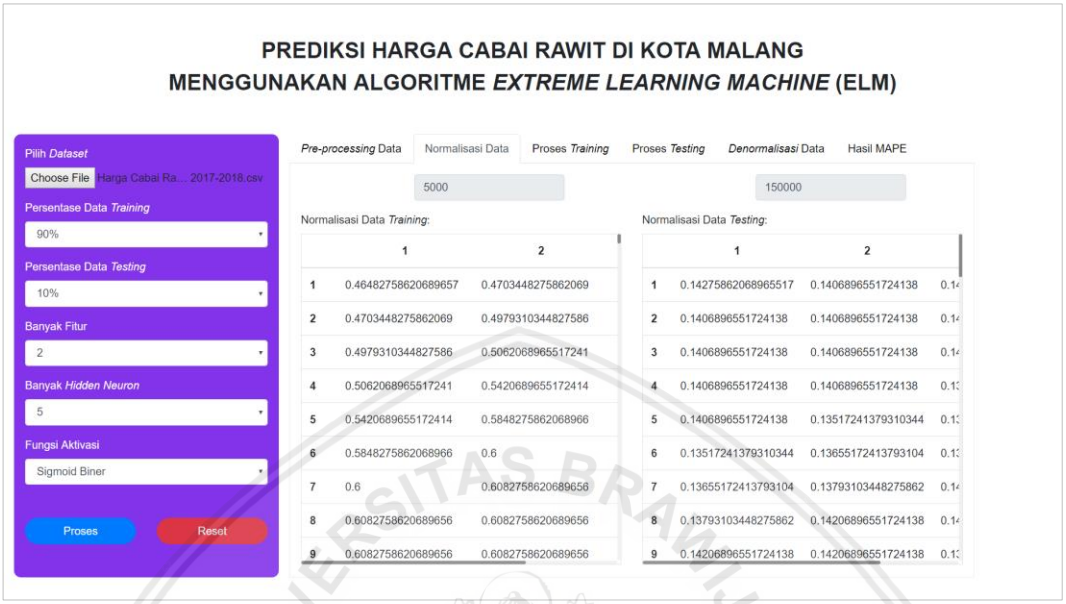
	1	2	3
1	72400	73200	77200
2	73200	77200	78400
3	77200	78400	83600
4	78400	83600	89800
5	83600	89800	92000
6	89800	92000	93200
7	92000	93200	93200
8	93200	93200	93200
9	93200	93200	94400
10	93200	94400	88600

	1	2	3
1	25700	25400	25400
2	25400	25400	25400
3	25400	25400	25400
4	25400	25400	24600
5	25400	24600	24800
6	24600	24800	25000
7	24800	25000	25600
8	25000	25600	25600
9	25600	25600	25200
10	25600	25200	23800

Gambar 5.1 Hasil Tampilan *Pre-processing* Data

5.2.2 Tampilan Normalisasi Data

Pada bagian ini menampilkan normalisasi untuk data di *training* dan untuk data di *testing*. Hasil tampilan normalisasi terdapat pada bagian Gambar 5.2.

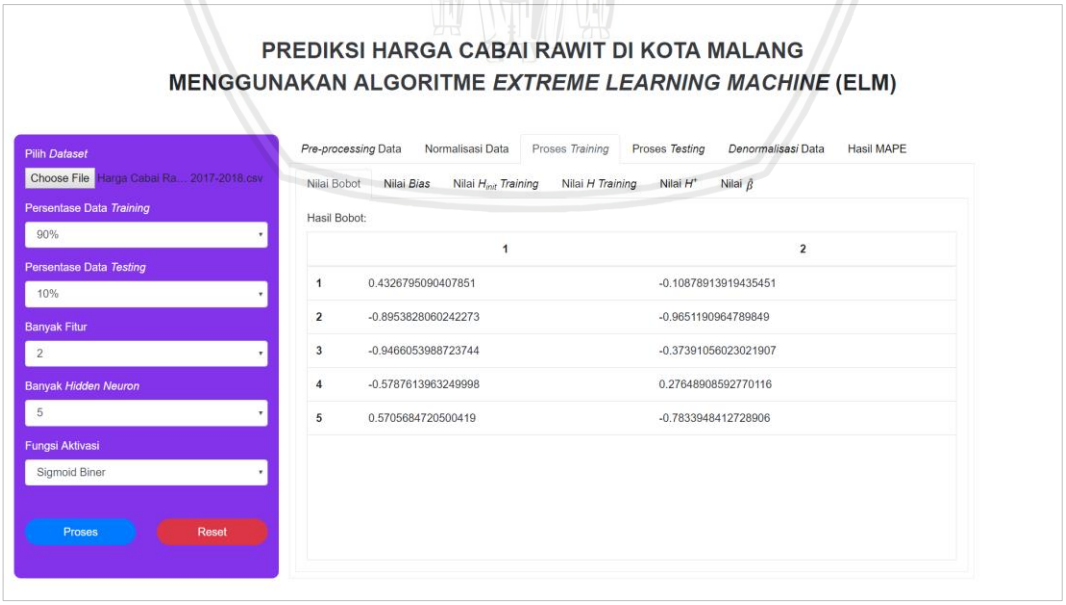


Gambar 5.2 Hasil Tampilan Normalisasi Data

5.2.3 Tampilan Proses *Training*

5.2.3.1 Perancangan Antarmuka Nilai Bobot

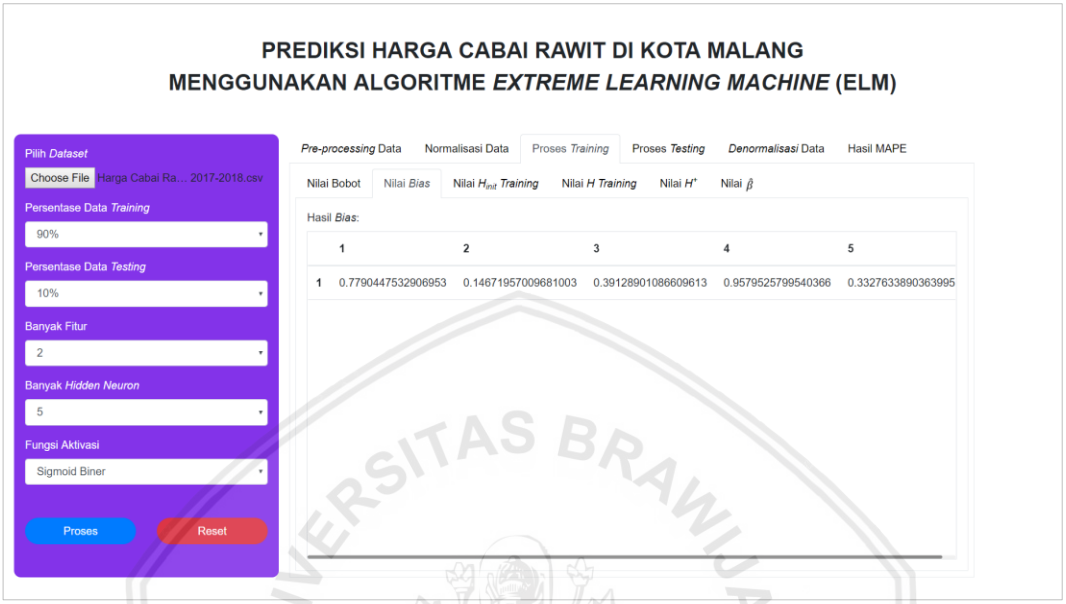
Pada bagian nilai bobot menampilkan hasil bobot pada proses bagian *training* dan proses *testing*. Didapatkan tampilan nilai bobot pada bagian Gambar 5.3.



Gambar 5.3 Hasil Tampilan Nilai Bobot

5.2.3.2 Perancangan Antarmuka Nilai *Bias*

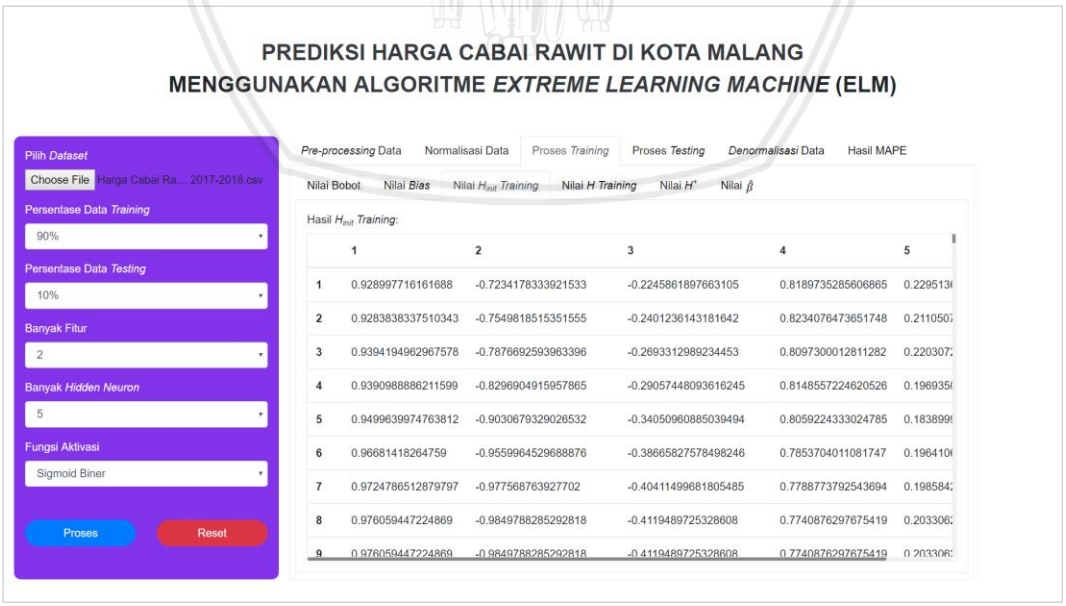
Pada bagian nilai *bias* menampilkan hasil nilai bias untuk bagian proses *training* dan proses *testing* yang diperoleh melalui perhitungan *random* antara [0,1]. Hasil tampilan nilai bias terdapat pada bagian Gambar 5.4.



Gambar 5.4 Hasil Tampilan Nilai *Bias*

5.2.3.3 Perancangan Antarmuka Nilai H_{init} *Training*

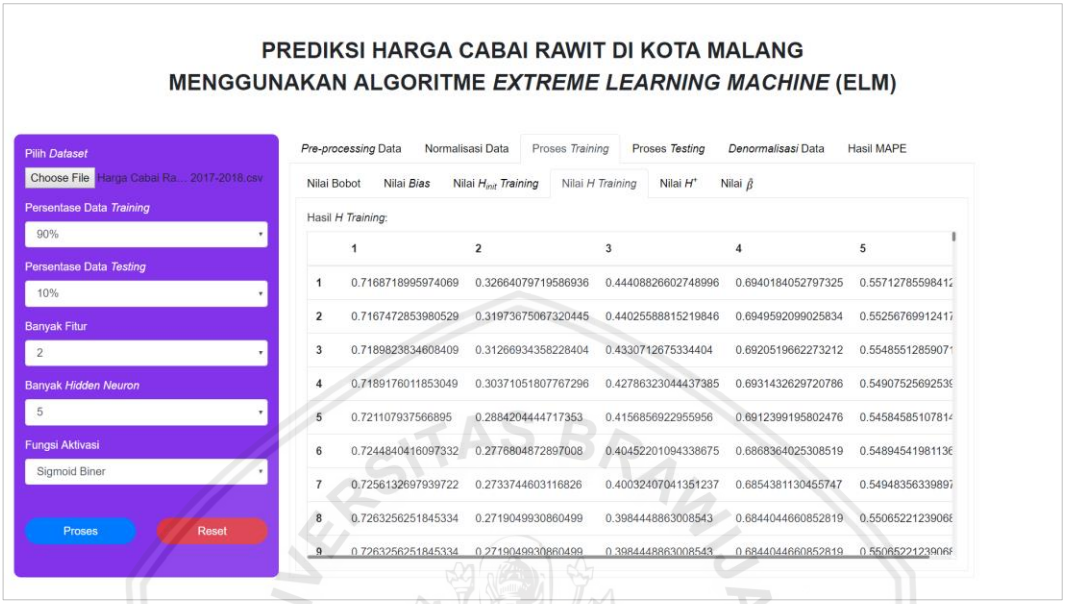
Pada bagian nilai H_{init} *training* menampilkan hasil perhitungan dalam bentuk tabel dengan baris sebanyak jumlah data *training* dan kolom sebanyak jumlah *hidden neuron*. Tampilan nilai H_{init} *training* terdapat pada bagian Gambar 5.5.



Gambar 5.5 Hasil Tampilan Nilai H_{init} *Training*

5.2.3.4 Perancangan Antarmuka Nilai *H Training*

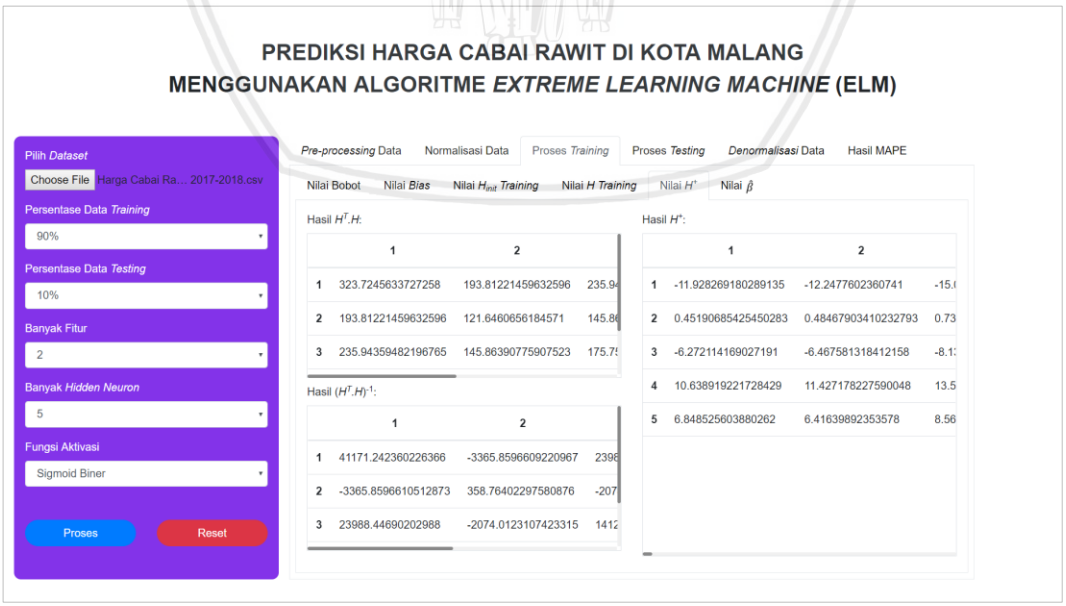
Pada bagian nilai *H training* menampilkan hasil perhitungan fungsi aktivasi *Sigmoid Biner* dalam bentuk tabel. Hasil tampilan nilai *H training* terdapat pada bagian Gambar 5.6.



Gambar 5.6 Hasil Tampilan Nilai *H Training*

5.2.3.5 Perancangan Antarmuka Nilai H^+

Pada bagian nilai H^+ menampilkan 3 hasil tampilan bagian hasil untuk $H^T.H$, hasil $(H^T.H)^{-1}$, dan hasil H^+. Hasil tampilan nilai H^+ terdapat pada bagian Gambar 5.7.



Gambar 5.7 Hasil Tampilan Nilai H^+

5.2.3.6 Perancangan Antarmuka Nilai $\hat{\beta}$

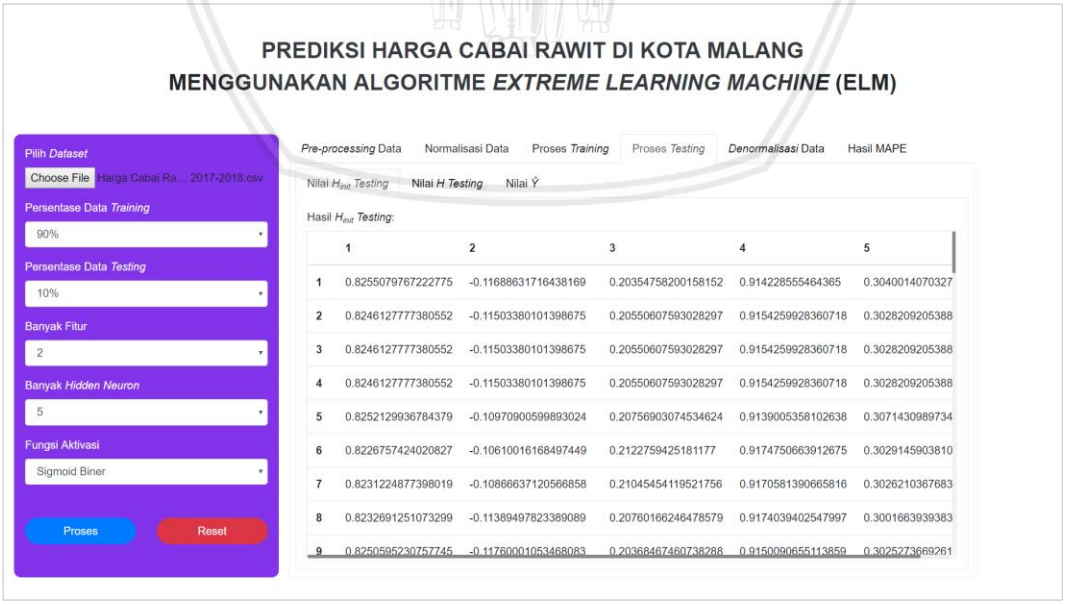
Pada bagian nilai $\hat{\beta}$ menampilkan hasil nilai akhir proses *training*. Hasil tampilan nilai *H training* terdapat pada bagian Gambar 5.8.

Gambar 5.8 Hasil Tampilan Nilai $\hat{\beta}$

5.2.4 Implementasi Proses *Testing*

5.2.4.1 Perancangan Antarmuka Nilai H_{init} *Testing*

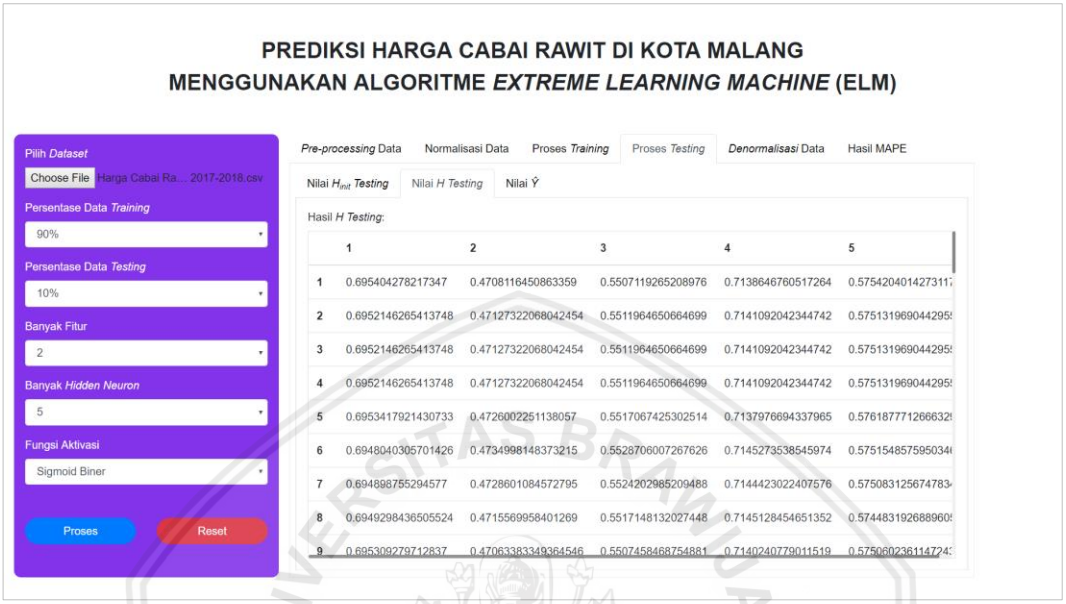
Pada bagian nilai H_{init} *testing* menampilkan hasil H_{init} *testing* dengan nilai data *testing*. Hasil tampilan nilai *H training* terdapat pada bagian Gambar 5.9.



Gambar 5.9 Hasil Tampilan Nilai H_{init} *Testing*

5.2.4.2 Perancangan Antarmuka Nilai *H Testing*

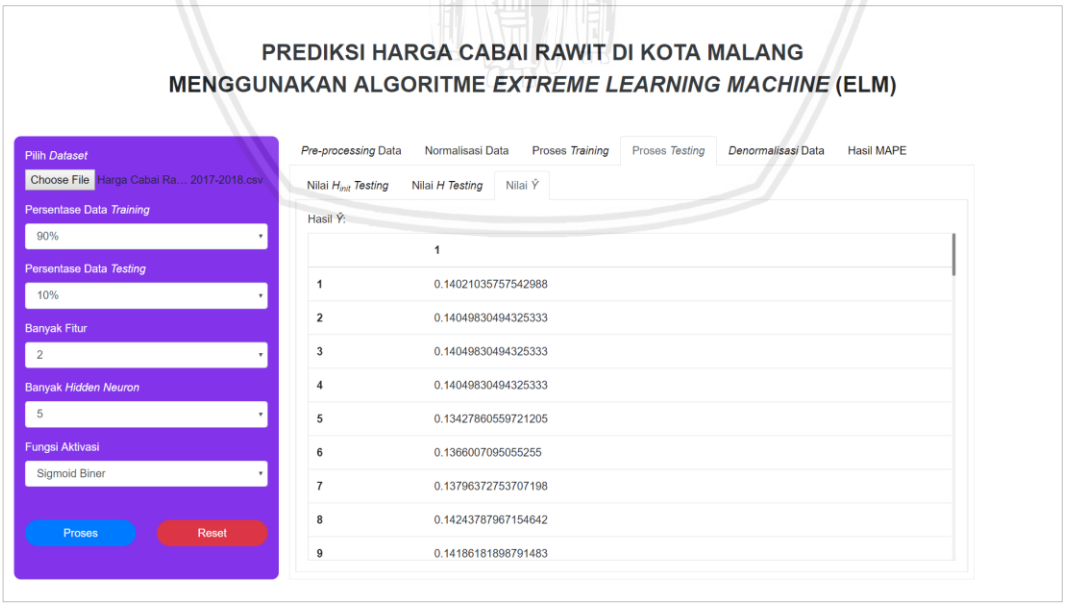
Pada bagian nilai *H testing* menampilkan hasil perhitungan fungsi aktivasi *Sigmoid Biner* dalam bentuk tabel. Hasil tampilan nilai *H testing* terdapat pada bagian Gambar 5.10.



Gambar 5.10 Hasil Tampilan *H Testing*

5.2.4.3 Perancangan Antarmuka Nilai $\hat{Y}$

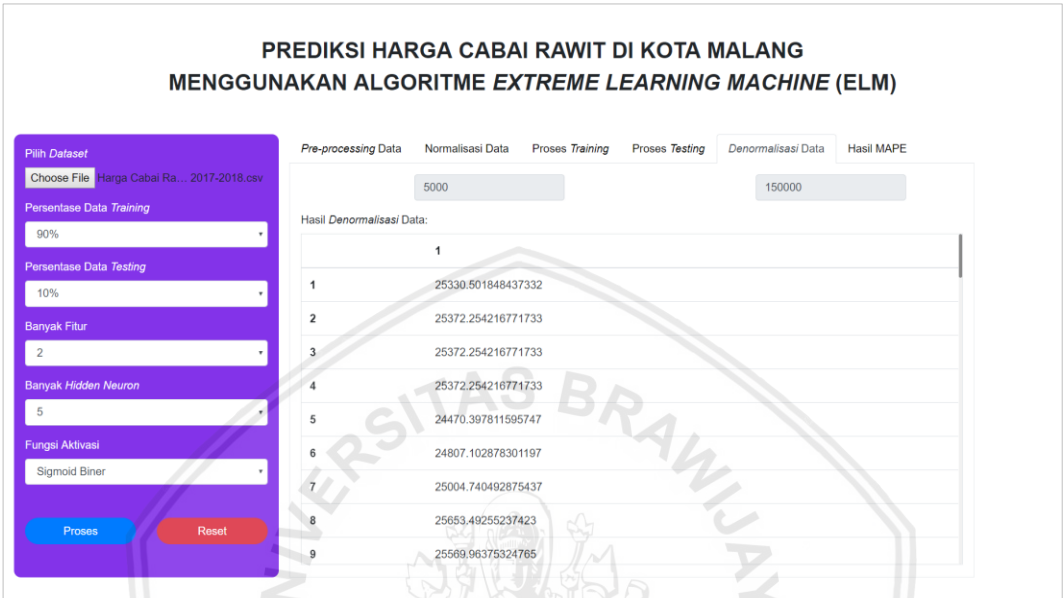
Pada bagian nilai $\hat{Y}$ menampilkan hasil prediksi dari perhitungan antara nilai $\hat{\beta}$ dengan nilai *H testing*. Tampilan nilai $\hat{Y}$ terdapat pada bagian Gambar 5.11.



Gambar 5.11 Hasil Tampilan Nilai $\hat{Y}$

5.2.5 Implementasi *Denormalisasi* Data

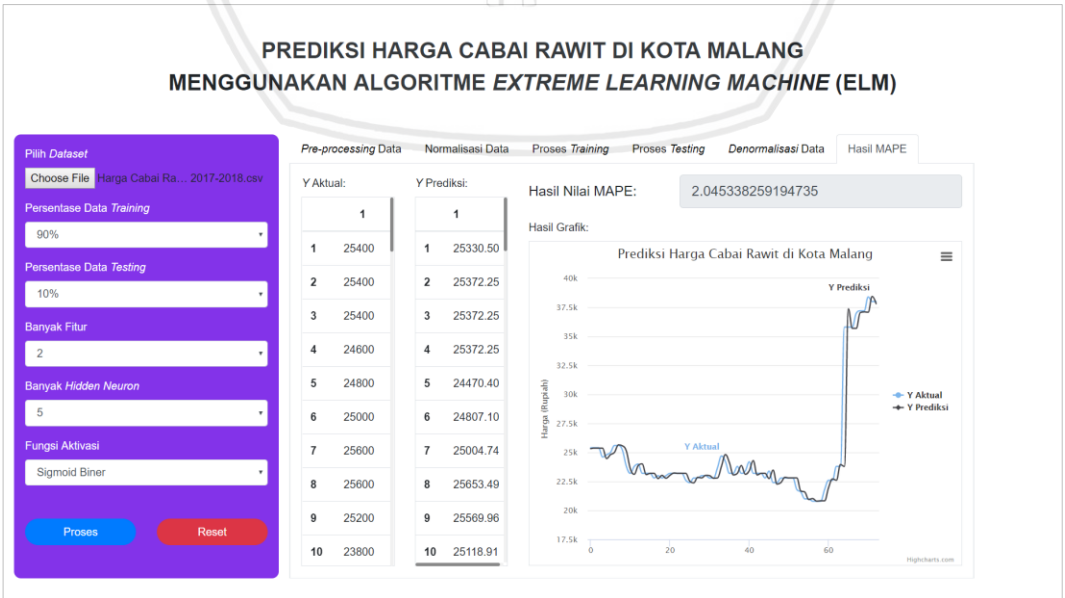
Pada bagian nilai yang dihasilkan dari proses ini, akan ditampilkan data dari hasil perhitungan *denormalisasi* untuk prediksi nilai $\hat{Y}$ menjadi nilai prediksi setelah *denormalisasi*. Hasil tampilan nilai *denormalisasi* data terdapat pada bagian Gambar 5.12.



Gambar 5.12 Hasil Tampilan *Denormalisasi* Data

5.2.6 Implementasi Hasil MAPE

Pada bagian hasil MAPE menampilkan hasil nilai aktual, nilai prediksi setelah *denormalisasi*, hasil grafik, dan hasil nilai MAPE. Hasil tampilan hasil MAPE terdapat pada bagian Gambar 5.13.



Gambar 5.13 Hasil Tampilan Nilai MAPE

BAB 6 HASIL DAN PEMBAHASAN

Pada bagian bab akan dilakukan penjelasan hasil dari pengujian yang telah dilakukan sebelumnya. Hasil pengujian dilakukan berdasarkan nilai MAPE yang diperoleh dari implementasi sistem. Hasil dan pembahasan mengenai pengujian yang akan dilakukan berupa pertama pengujian untuk banyak fitur, kedua pengujian untuk persentase banyak pada data *training* dan juga data *testing*, ketiga pengujian untuk banyak *neuron* pada *hidden layer*, dan keempat pengujian untuk fungsi aktivasi yang digunakan. Dari setiap hasil yang diperoleh setiap pengujian akan dilakukan pembahasan disetiap hasil pengujian.

6.1 Hasil Pengujian

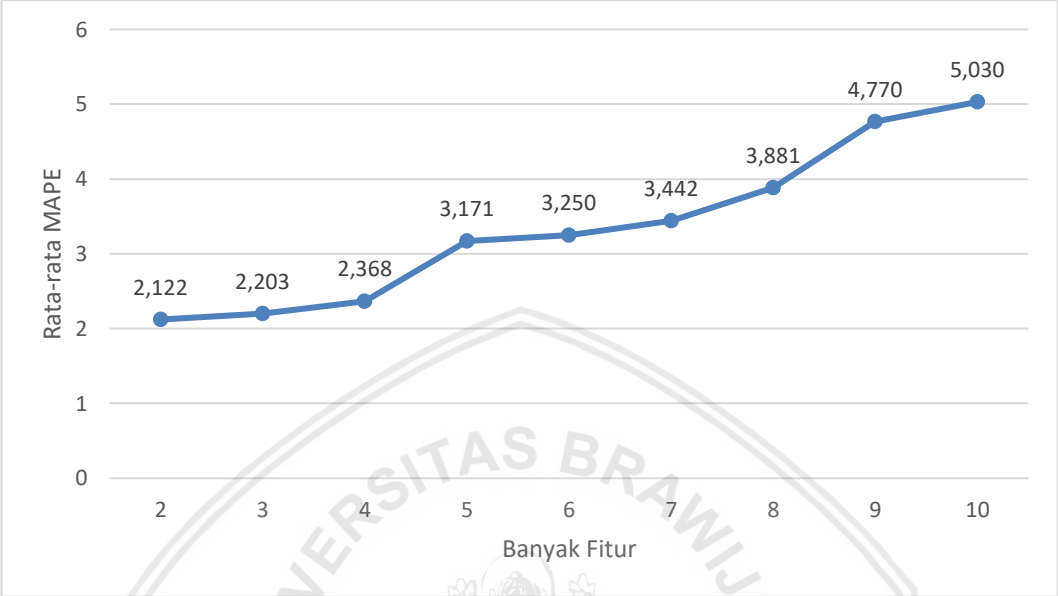
6.1.1 Pengujian Banyak Fitur

Pada bagian pengujian untuk banyak fitur akan dilakukan pengujian terhadap parameter ELM untuk mengetahui nilai fitur yang paling optimal berdasarkan banyak hari sebelumnya. Nilai parameter yang digunakan sebagai pengujian banyak fitur adalah 2 sampai 10. Dari nilai tersebut diambil berdasarkan hari sebelumnya, jika parameter nilai banyak fitur yang ditentukan sebanyak 2 fitur maka data yang diambil untuk dijadikan fitur adalah 2 hari sebelumnya. Dengan adanya pengujian banyak fitur akan dapat diketahui apakah pengujian pada banyak fitur bisa membuat pengaruh terhadap nilai MAPE. Setiap parameter nilai yang diuji dilakukan sebanyak 10 kali percobaan dengan nilai parameter lain yang tetap untuk persentase data *training* bernilai 90%, persentase data *testing* bernilai 10%, banyak *neuron* pada *hidden layer* sebesar 5, dan fungsi aktivasi yang digunakan menggunakan *Sigmoid Biner*. Berikut nilai dari pengujian ini terdapat pada bagian Tabel 6.1 dan hasil tabel secara detail pada Lampiran E.

Tabel 6.1 Hasil Pengujian Banyak Fitur

Banyak Fitur	MAPE Percobaan ke-						Rata-rata MAPE
	1	2	3	…	9	10	
2	**2,036**	**2,140**	**2,123**	…	**2,042**	**2,272**	**2,122**
3	2,012	2,337	2,116	…	2,282	2,200	2,203
4	2,910	2,712	2,378	…	2,367	2,678	2,368
5	3,804	2,810	3,078	…	2,369	2,918	3,171
6	3,078	4,050	2,825	…	4,464	3,880	3,250
7	3,642	3,737	2,987	…	4,401	4,398	3,442
8	3,619	4,484	3,173	…	3,437	4,815	3,881
9	5,193	5,892	6,412	…	4,576	5,051	4,770
10	5,139	4,491	3,605	…	7,598	5,656	5,030

Dari hasil Tabel 6.1 diperoleh hasil ditampilkan nilai dari rata-rata MAPE yang dihasilkan dari 10 percobaan pada pengujian banyak fitur yang ditampilkan dalam bentuk diagram garis. Berikut diagram garis untuk rata-rata hasil pengujian banyak fitur terdapat pada bagian Gambar 6.1.



Gambar 6.1 Hasil Pengujian Banyak Fitur

Berdasarkan hasil pengujian pada bagian Tabel 6.1 dan hasil grafik pengujian pada bagian Gambar 6.1 menunjukkan hasil pengujian nilai banyak fitur terhadap nilai MAPE yang dihasilkan. Hasil nilai dari rata-rata MAPE yang menghasilkan nilai terkecil dihasilkan pada banyak fitur yang bernilai 2 dengan nilai 2,122% sedangkan nilai rata-rata MAPE dengan nilai terbesar terjadi pada banyak fitur 10 dengan nilai 5,030%. Dari hasil pengujian banyak fitur pada bagian Gambar 6.1 dapat diambil kesimpulan bahwa untuk setiap perubahan nilai dari banyak fitur jika fitur yang digunakan semakin banyak maka hasil nilai dari rata-rata MAPE juga akan semakin besar dan jika fitur yang digunakan semakin sedikit maka hasil nilai dari rata-rata MAPE juga akan semakin kecil. Hal tersebut disebabkan karena jika banyak fitur semakin sedikit maka jumlah data yang akan dihasilkan semakin banyak, sehingga memengaruhi dalam proses pembelajaran karena jika jumlah data latih yang dihasilkan semakin banyak maka hasil kesalahan prediksi yang dihasilkan juga akan semakin kecil dan pengaruh perubahan data harga cabai rawit di Kota Malang yang signifikan cenderung berlangsung hanya dalam waktu beberapa hari. Hal tersebut membuktikan bahwa pengujian banyak fitur dapat memengaruhi nilai MAPE yang dihasilkan serta dari seluruh banyak fitur yang diujikan termasuk dalam hasil MAPE yang sangat baik.

6.1.2 Pengujian Persentase Banyak Data *Training* dan Data *Testing*

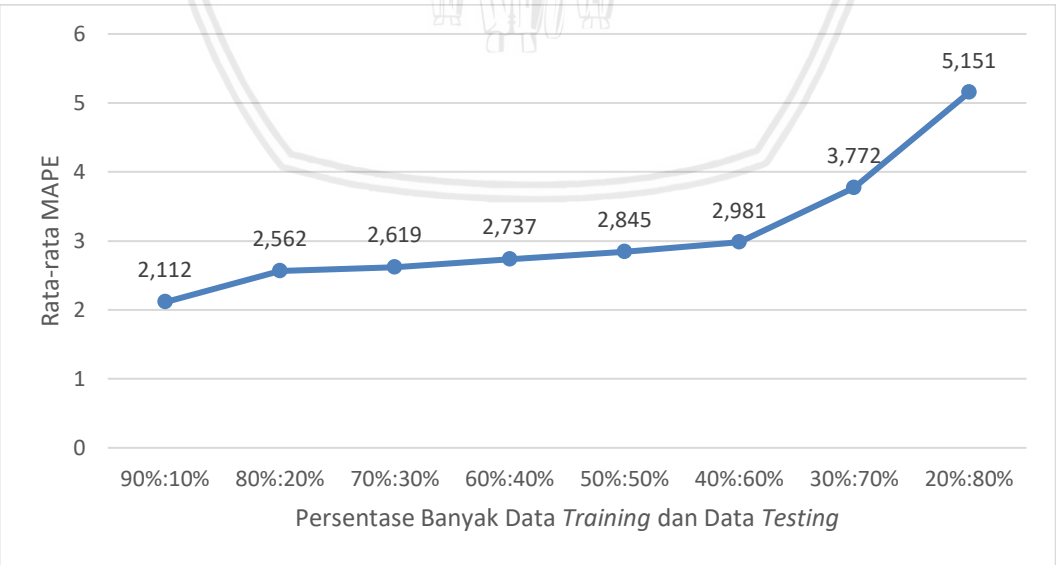
Pada pengujian ini dilakukan perbandingan persentase antara banyak data *training* dan data *testing*. Pengujian ini diharapkan dapat mengetahui nilai perbandingan data yang menghasilkan MAPE terkecil. Pengujian perbandingan banyak data dilakukan sebanyak 8 variasi yaitu dengan perbandingan 90%:10%,

80%:20%, 70%:30%, 60%:40%, 50%:50%, 40%:60%, 30%:70%, dan 20%:80%. Nilai parameter untuk banyak fitur sebesar 2, banyak *neuron* pada *hidden layer* yaitu 5, dan fungsi aktivasi yang dipakai adalah *Sigmoid Biner*. Hasil yang diperoleh dari pengujian persentase banyak data *training* dan data *testing* terdapat pada bagian Tabel 6.2 dan hasil tabel secara detail terdapat pada Lampiran F.

Tabel 6.2 Hasil Pengujian Persentase Banyak Data *Training* dan Data *Testing*

Data *Training* : Data *Testing*	MAPE Percobaan ke-						Rata-rata MAPE
	1	2	3	…	9	10	
90%:10%	**2,096**	**2,149**	**2,065**	…	**2,154**	**2,106**	**2,112**
80%:20%	2,666	2,546	2,576	…	2,546	2,550	2,562
70%:30%	2,614	2,614	2,614	…	2,604	2,659	2,619
60%:40%	2,738	2,727	2,723	…	2,740	2,726	2,737
50%:50%	2,842	2,860	2,833	…	2,876	2,818	2,845
40%:60%	2,972	2,959	2,997	…	3,013	2,975	2,981
30%:70%	5,623	3,036	4,971	…	2,984	2,988	3,772
20%:80%	4,875	5,736	6,752	…	3,192	6,551	5,151

Dari hasil pada bagian Tabel 6.2 dapat ditampilkan rata-rata nilai MAPE yang diperoleh dari 10 percobaan yang ditampilkan dalam bentuk diagram garis. Berikut diagram garis untuk rata-rata hasil pengujian persentase banyak data *training* dan data *testing* dapat dilihat pada bagian Gambar 6.2.



Gambar 6.2 Hasil Pengujian Persentase Banyak Data *Training* dan Data *Testing*

Berdasarkan hasil pengujian pada bagian Tabel 6.2 dan hasil grafik pengujian pada bagian Gambar 6.2 menunjukkan hasil pengujian nilai perbandingan untuk persentase banyak pada data *training* dan data *testing* terhadap nilai MAPE yang dihasilkan. Hasil nilai dari rata-rata MAPE terkecil terlihat pada persentase data *training*: persentase data *testing* sebesar 90%:10% dengan nilai sebesar 2,112% sedangkan nilai dari rata-rata MAPE terbesar terjadi pada persentase data *training* dan persentase data *testing* sebesar 20%:80% dengan nilai 5,151%. Dari hasil pengujian pada bagian Gambar 6.2 dapat disimpulkan bahwa terjadi kenaikan nilai MAPE ketika persentase data *training* semakin besar dan persentase data *testing* semakin kecil. Hal tersebut dapat disebabkan karena saat banyak data *training* semakin besar maka proses pembelajaran yang dilakukan oleh algoritme akan semakin baik, jika banyak data *training* terlalu sedikit dibandingkan dengan banyak data *testing* menyebabkan algoritme kurang mampu mempelajari jumlah data yang diujikan dengan pengetahuan pembelajaran yang sedikit. Hal tersebut dapat membuktikan bahwa dalam pengujian persentase banyak data *training* dan data *testing* dapat memengaruhi nilai MAPE yang dihasilkan termasuk dalam hasil MAPE yang sangat baik.

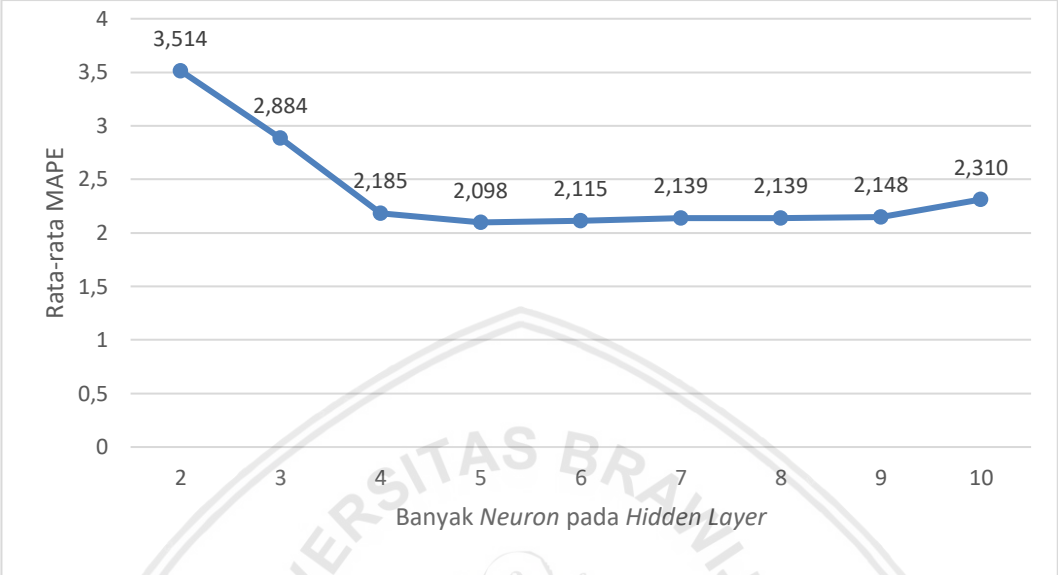
6.1.3 Pengujian Banyak *Neuron* pada *Hidden Layer*

Pada bagian pengujian banyak *neuron* akan dilakukan pengujian terhadap parameter ELM untuk mengetahui nilai banyak *neuron* yang paling optimal. Nilai parameter yang digunakan sebagai pengujian banyak *neuron* pada *hidden layer* adalah 2 sampai 10. Nilai tersebut merupakan nilai yang akan dijadikan input pada parameter *hidden layer* sedangkan untuk nilai persentase banyak data *training* sebesar 90%, persentase banyak data *testing* sebesar 10%, banyak fitur sebesar 2, dan fungsi aktivasi yang digunakan adalah *Sigmoid Biner*. Hasil yang diperoleh dari pengujian banyak *neuron* pada *hidden layer* terdapat pada bagian Tabel 6.3 dan hasil detail tabel terdapat pada Lampiran G.

Tabel 6.3 Hasil Pengujian Banyak *Neuron* pada *Hidden Layer*

Banyak *Neuron*	MAPE Percobaan ke-						Rata-rata MAPE
	1	2	3	…	9	10	
2	4,494	3,884	2,605	…	3,123	2,932	3,514
3	2,522	2,557	2,100	…	5,226	2,077	2,884
4	2,080	2,072	2,082	…	2,131	2,106	2,185
5	**2,091**	**2,081**	**2,072**	…	**2,091**	**2,104**	**2,098**
6	2,040	2,124	2,136	…	2,109	2,112	2,115
7	2,160	2,180	2,138	…	2,124	2,142	2,139
8	2,121	2,139	2,159	…	2,077	2,138	2,139
9	2,173	2,106	2,223	…	2,185	2,153	2,148
10	2,144	2,094	2,951	…	2,260	2,197	2,310

Dari hasil bagian Tabel 6.3 dapat ditampilkan rata-rata nilai MAPE dari 10 percobaan yang dilakukan dalam bentuk diagram garis. Berikut diagram garis untuk rata-rata hasil pengujian banyak *neuron* pada *hidden layer* dapat dilihat pada bagian Gambar 6.3.



Gambar 6.3 Hasil Pengujian Banyak *Neuron* pada *Hidden Layer*

Berdasarkan dari hasil pengujian pada bagian Tabel 6.3 dan grafik hasil pengujian pada bagian Gambar 6.3 menunjukkan bahwa nilai rata-rata hasil MAPE terkecil terlihat pada banyak *neuron* sebesar 5 dengan nilai sebesar 2,098% dan nilai dari rata-rata MAPE terbesar terjadi pada banyak *neuron* sebesar 2 dengan nilai 3,514%. Dari hasil penggujian pada Gambar 6.3 terjadi penurunan dan kenaikan nilai MAPE ketika banyak *neuron* semakin besar. Penurunan nilai MAPE terjadi saat banyak *neuron* antara 2-5. Kenaikan nilai MAPE terjadi antara banyak *neuron* 6-10 yang cenderung mengalami kenaikan yang signifikan. Hal tersebut disebabkan karena semakin besar dari nilai banyak *neuron* pada *hidden layer* yang diberikan maka hal tersebut dapat meningkatkan hasil ketepatan nilai prediksi mendekati nilai aktual dengan kesalahan yang kecil dan semakin besar banyak *neuron* maka hal tersebut juga mampu membuat ruang fitur yang terbentuk semakin besar sehingga membuat pencarian ruang fitur semakin luas. Hal tersebut membuktikan bahwa perlu dilakukan pengujian parameter banyak *neuron* pada *hidden layer* agar dapat mengetahui parameter optimal karena tidak selalu semakin besar nilai banyak *neuron* akan menghasilkan tingkat kesalahan yang semakin kecil.

6.1.4 Pengujian Fungsi Aktivasi

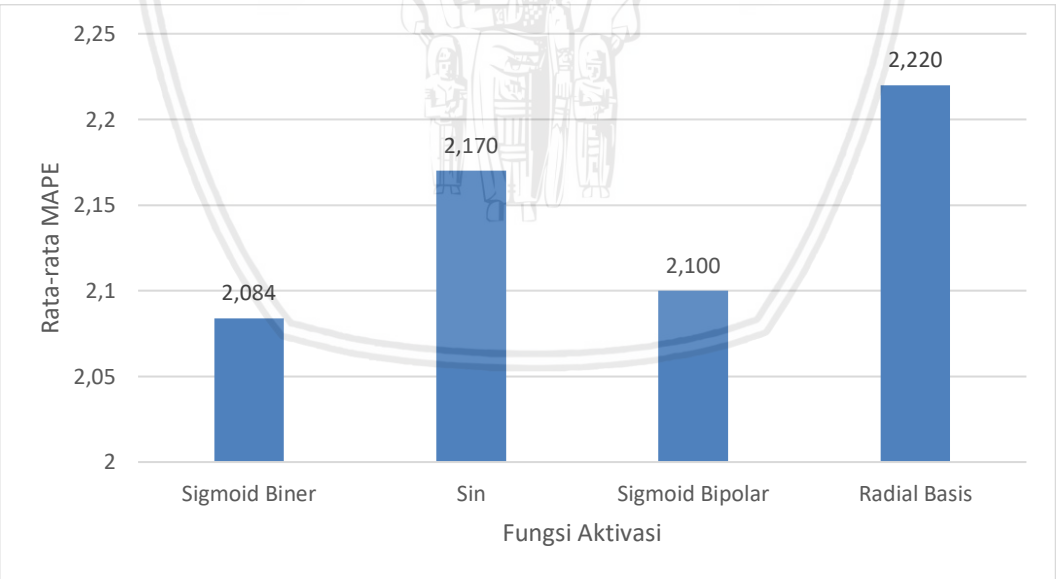
Pada pengujian fungsi aktivasi akan dilakukan uji untuk mengetahui parameter fungsi aktivasi yang paling optimal dengan nilai kesalahan paling kecil pada algoritme ELM. Dengan menggunakan fungsi aktivasi yang berbeda diharapkan dapat mengetahui setiap hasil yang diperoleh pada setiap fungsi. Fungsi yang digunakan dalam penelitian ini sebanyak 4 yaitu fungsi *Sigmoid*

Biner, fungsi *Sin*, fungsi *Sigmoid Bipolar*, dan fungsi *Radial Basis*. Sedangkan untuk nilai parameter persentase banyak data *training* sebesar 90%, persentase banyak data *testing* sebesar 10%, banyak fitur sebesar 2, dan banyak *neuron* pada *hidden layer* sebesar 5. Hasil yang diperoleh dari pengujian fungsi aktivasi terdapat pada Tabel 6.4 dan hasil detail tabel terdapat pada Lampiran H.

Tabel 6.4 Hasil Pengujian Fungsi Aktivasi

Fungsi Aktivasi	MAPE Percobaan ke-						Rata-rata MAPE
	1	2	3	…	9	10	
Sigmoid Biner	**2,104**	**2,077**	**2,137**	…	**2,116**	**2,075**	**2,084**
Sin	2,118	2,103	2,067	…	2,746	2,176	2,170
Sigmoid Bipolar	2,097	2,114	2,054	…	2,078	2,145	2,100
Radial Basis	2,104	2,489	2,028	…	2,238	2,062	2,220

Dari hasil Tabel 6.4 dapat ditampilkan rata-rata nilai MAPE yang dihasilkan dari 10 percobaan yang dilakukan dalam bentuk diagram batang. Berikut diagram batang untuk rata-rata MAPE dari hasil pengujian fungsi aktivasi dapat dilihat pada Gambar 6.4.



Gambar 6.4 Hasil Pengujian Fungsi Aktivasi

Berdasarkan hasil pengujian pada bagian Tabel 6.4 dan hasil grafik pengujian pada bagian Gambar 6.4 menunjukkan hasil pengujian fungsi aktivasi terhadap nilai MAPE yang dihasilkan. Hasil nilai dari rata-rata hasil MAPE terkecil terlihat pada fungsi aktivasi *Sigmoid Biner* sebesar 2,084% sedangkan hasil nilai dari rata-rata MAPE terbesar terjadi pada fungsi aktivasi *Radial Basis* sebesar 2,220%. Hal tersebut dapat disebabkan karena dalam penggunaan fungsi aktivasi berfungsi

untuk mengaktifkan agar hasil prediksi dapat mengikuti terhadap pola pergerakan data aktual sesuai dengan rentang nilai yang dihasilkan pada masing-masing fungsi aktivasi. Fungsi aktivasi *Sigmoid Biner* memberikan hasil yang paling optimal dikarenakan nilai yang terbentuk dapat menyesuaikan rentang data. Hal tersebut dapat dibuktikan bahwa fungsi aktivasi yang dimiliki oleh *Sigmoid Biner* terbukti dapat berjalan baik dengan menghasilkan nilai MAPE terendah. Dari keseluruhan hasil pengujian fungsi aktivasi dapat disimpulkan bahwa parameter pengujian untuk jenis fungsi aktivasi memiliki pengaruh terhadap hasil dari prediksi pada algoritme *Extreme Learning Machine*.

BAB 7 PENUTUP

Pada bab penutup akan menjelaskan dari kesimpulan dan saran dari penelitian prediksi harga cabai rawit di Kota Malang menggunakan algoritme *Extreme Learning Machine*. Bagian kesimpulan akan menjelaskan hasil jawaban dari rumusan masalah pada penelitian ini. Bagian saran akan menjelaskan kekurangan dalam penelitian ini yang dapat digunakan sebagai referensi untuk penelitian selanjutnya.

7.1 Kesimpulan

Berdasarkan hasil dan pembahasan pengujian pada bab sebelumnya tentang prediksi harga cabai rawit di Kota Malang menggunakan algoritme *Extreme Learning Machine*. Kesimpulan yang diperoleh dari penelitian ini sebagai berikut:

1. Algoritme *Extreme Learning Machine* untuk prediksi harga cabai rawit di Kota Malang menghasilkan nilai parameter yang optimal melalui pengujian parameter pada banyak fitur, persentase banyak data *training* dan data *testing*, banyak *neuron* pada *hidden layer*, dan fungsi aktivasi. Hasil rata-rata MAPE terkecil yang diperoleh pada pengujian banyak fitur menghasilkan nilai banyak fitur sebanyak 2 dengan hasil nilai rata-rata MAPE sebesar 2,122%. Pada pengujian persentase banyak data *training* dan data *testing* menghasilkan nilai perbandingan data 90%:10% menghasilkan rata-rata MAPE sebesar 2,112%. Pada pengujian banyak *neuron* pada hidden layer menghasilkan nilai banyak *neuron* sebanyak 5 menghasilkan rata-rata MAPE sebesar 2,098%. Sedangkan pada pengujian fungsi aktivasi yang digunakan menghasilkan rata-rata MAPE sebesar 2,084% untuk fungsi *Sigmoid Biner*.
2. Algoritme *Extreme Learning Machine* untuk prediksi harga cabai rawit di Kota Malang menghasilkan nilai *error rate* terkecil dengan menggunakan *Mean Absolute Percentage Error* (MAPE) pada pengujian nilai parameter. Parameter nilai yang digunakan pada persentase data *training* dan data *testing* sebesar 90%:10%, nilai banyak fitur sebesar 2, nilai banyak *neuron* pada *hidden layer* sebesar 5, dan fungsi aktivasi menggunakan *Sigmoid Biner* menghasilkan nilai MAPE sebesar 2,087%. Dari hasil nilai MAPE terebut diperoleh hasil rata-rata perbedaan harga prediksi dengan harga aktual yang akan dialami oleh konsumen saat melakukan pembelian harga cabai rawit di Kota Malang sebesar Rp570,36.

7.2 Saran

Saran yang digunakan untuk penelitian selanjutnya sebagai berikut:

1. Penambahan faktor terhadap prediksi harga cabai rawit di Kota Malang tidak hanya berdasarkan faktor harga sebelumnya tetapi faktor lain yang dapat memengaruhi antara lain yaitu faktor jumlah produksi.
2. Peningkatan dalam penentuan nilai bobot pada algoritme *Extreme Learning Machine* yang masih menggunakan hasil *random* pertama sebagai nilai bobot

dalam perhitungan, dengan dilakukan optimasi terhadap nilai bobot yang akan digunakan pada algoritme *Extreme Learning Machine* menggunakan algoritme *Particle Swarm Optimization* untuk mencari *random* bobot yang optimal dengan *error* terkecil.

DAFTAR REFERENSI

Anggraini, S.D., 2017. Prediksi Nilai Tukar Mata Uang Asing Menggunakan Extreme Learning Machine. *Jurnal Ilmiah Matematika*, 2(6), pp.110–115.

Bahiuddin, I., Mazlan, S.A., Shapiai, M.I., Imaduddin, F. & Ubaidillah, 2018. Study of Extreme Learning Machine Activation Functions for Magnetorheological Fluid Modelling in Medical Devices Application. *Proceeding of 2017 International Conference on Robotics, Automation and Sciences, ICORAS 2017*, 2018–March, pp.1–5.

BPS, 2018. Statistik Daerah Kota Malang 2018. [online] Tersedia di: <https://malangkota.bps.go.id/publication/2019/01/16/5660721147918ac671c37e4d/statistik-daerah-kota-malang-2018.html> [Diakses 16 Januari 2019].

Cao, W., Gao, J., Ming, Z. & Cai, S., 2017. Some Tricks in Parameter Selection for Extreme Learning Machine.

Cholissodin, I. & Sutrisno, 2018. Prediction of Rainfall using Simplified Deep Learning based Extreme Learning Machines. 3(2), pp.120–131.

Darmayanti, E.Y., Setiawan, B.D. & Bachtiar, F.A., 2018. Particle Swarm Optimization Untuk Optimasi Bobot Extreme Learning Machine Dalam Memprediksi Produksi Gula Kristal Putih Pabrik Gula Candi Baru-Sidoarjo. *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer*, 2(11), pp.5096–5104.

Ertuğrul, Ö.F. & Kaya, Y., 2014. A Detailed Analysis on Extreme Learning Machine and Novel Approaches Based on ELM. *American Journal of Computer Science and Engineering*, [online] 1(5), pp.43–50. Tersedia di: <http://www.openscienceonline.com/journal/ajcse> [Diakses 24 Januari 2019].

Fachrony, A., Cholissodin, I. & Santoso, E., 2018. Implementasi Algoritme Extreme Learning Machine (ELM) untuk Prediksi Beban Pemanasan dan Pendinginan Bangunan. *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer*, 2(9), pp.3043–3049.

Fikriya, Z.A., Irawan, M.I. & Soetrisno, 2017. Implementasi Extreme Learning Machine untuk Pengenalan Objek Citra Digital. 6(1).

Giusti, A., Widodo, A.W. & Adinugroho, S., 2018. Prediksi Penjualan Mi Menggunakan Metode Extreme Learning Machine (ELM) di Kober Mie Setan Cabang Soekarno Hatta. *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer*, 2(8), pp.2972–2978.

Gumelar, R.A., Setiawan, B.D. & Adikara, P.P., 2019. Peramalan Harga Cabai Menggunakan Metode High Order Fuzzy Times Series Multifactors. *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer*, 3(1), pp.1038–1046.

Hanifa, T.T., Adiwijaya & Al-faraby, S., 2017. Analisis Churn Prediction pada Data Pelanggan PT. Telekomunikasi dengan Logistic Regression dan Underbagging. 4(2), pp.3210–3225.

Hanke, J.E. & Wichern, D.W., 2009. *Business Forecasting*. 9th ed. New Jersey: Pearson Education.

Huang, G. Bin, Zhu, Q.Y. & Siew, C.K., 2004. Extreme Learning Machine: A New Learning Scheme of Feedforward Neural Networks. *IEEE International Conference on Neural Networks - Conference Proceedings*, 2(August 2004), pp.985–990.

Huang, G. Bin, Zhu, Q.Y. & Siew, C.K., 2006. Extreme Learning Machine: Theory and Applications. *Neurocomputing*, 70(1–3), pp.489–501.

Huixuan, F., Yuchao, W. & Zhang, H., 2015. Ship Rolling Motion Prediction Based on Extreme Learning Machine. *Chinese Control Conference*, pp.3468–3472.

Kartikasari, D.N., Purnamaningsih, S.L. & Soetopo, L., 2016. Penampilan Galur Generasi Pertama Hasil Seleksi Dari Cabai Rawit (Capsicum frutescens L.) Varietas Lokal. *Jurnal Produksi Tanaman*, 4(4), pp.320–324.

Mosabeth, C., Furqon, M.T. & Wihandika, R.C., 2018. Prediksi Harga Pasar Daging Sapi Di Kota Malang Dengan Menggunakan Metode Extreme Learning Machine (ELM). *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer*, 2(12), pp.6362–6369.

Mustaffa, Z. & Yusof, Y., 2011. A Comparison of Normalization Techniques in Predicting Dengue Outbreaik. *International Conference on Business and Economics Research*, 1, pp.345–349.

Rizaldy, D.Z., 2017. Pengaruh Harga Komoditas Pangan Terhadap Inflasi Di Kota Malang Tahun 2011-2016. *Jurnal Ekonomi Pembangunan*, 15.

Rosmainar, L., Ningsih, W., Ayu, N.P. & Nanda, H., 2018. Penentuan Kadar Vitamin C Beberapa Jenis Cabai (Capsicum sp.) Dengan Spektrofotometri UV-VIS. 3(1), pp.1–5.

Rusdiana, 2014. *Manajemen operasi*. 1st ed. Bandung: CV. Pustaka Setia.

Ukhra, A.U., 2014. Pemodelan dan Peramalan Data Deret Waktu dengan Metode Seasonal ARIMA. 3(3), pp.59–67.

Disperindag Jatim. 2018. Sistem Informasi Ketersediaan dan Perkembangan Harga Bahan Pokok di Jawa Timur. [online] Tersedia di: <http://siskaperbapo.com/harga/tabel> [Diakses 7 Januari 2019].

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

Suyanto. 2014. *Artificial Intelligence Searching, Reasoning, Planning dan Learning*. 2th ed. Yogyakarta: Informatika Bandung.