IMPLEMENTASI METODE EXTREME LEARNING MACHINE (ELM) UNTUK MEMPREDIKSI JUMLAH DEBIT AIR YANG LAYAK DIDISTRIBUSI (STUDI KASUS: PDAM KABUPATEN GOWA MAKASSAR)

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:
Putri Indhira Utami P
NIM: 165150201111288

PROGRAM STUDI TEKNIK INFORMATIKA
JURUSAN TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2020

PENGESAHAN

IMPLEMENTASI METODE *EXTREME LEARNING MACHINE* (ELM) UNTUK MEMPREDIKSI JUMLAH DEBIT AIR YANG LAYAK DIDISTRIBUSI (STUDI KASUS: PDAM KABUPATEN GOWA MAKASSAR)

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh :
Putri Indhira Utami P
NIM: 165502011112288

Skripsi ini telah diuji dan dinyatakan lulus pada
9 April 2020
Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I	Dosen Pembimbing II
Muhammad Tanzil Furqon, S.Kom., M.CompSc.	Ir. Sutrisno, M.T.
NIP. 19820930 200801 1 004	NIP. 19570325 198701 1 001

Mengetahui
Ketua Jurusan Teknik Informatika

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

PERNYATAAN ORISINALITAS

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, di dalam naskah skripsi ini tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik 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 didalam naskah skripsi ini dapat dibuktikan terdapat unsur-unsur plagiasi, saya bersedia skripsi ini digugurkan dan gelar akademik yang telah saya peroleh (sarjana) dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku (UU No. 20 Tahun 2003, Pasal 25 ayat 2 dan Pasal 70).

Malang, 3 Maret 2020

Putri Indhira Utami P

NIM: 165150201111288

ABSTRAK

Putri Indhira Utami P, Implementasi Metode *Extreme Learning Machine* (ELM) Untuk Memprediksi Jumlah Debit Air yang Layak Didistribusi (Studi Kasus: PDAM Kabupaten Gowa Makassar)

Pembimbing: Muhammad Tanzil Furqon, S.Kom., M.CompSc dan Ir. Sutrisno, M.T.

PDAM Kabupaten Gowa Kota Makassar merupakan perusahaan di bawah pemerintah yang melakukan proses produksi air dan dilanjutkan dengan pendistribusian air bersih ke rumah penduduk. Jika jumlah air yang diproduksi banyak, berarti ketersediaan air bagi PDAM juga banyak, sehingga bisa memenuhi kebutuhan masyarakat bahkan bisa untuk menambah pelanggan. Namun, faktor perubahan musim yang kadang terjadi sangat berpengaruh dalam jumlah air yang diproduksi. Sehingga masalah utama yang dihadapi adalah tidak menentunya jumlah produksi air yang tentunya berdampak pada jumlah distribusi air PDAM yang akan disalurkan ke rumah penduduk. Terlebih lagi dalam jumlah produksi air oleh PDAM, tidak semua dapat didistribusikan dikarenakan harus melewati beberapa tahap pengecekan kualitas air agar air yang didistribusikan sesuai dengan standar yang telah ditetapkan oleh pemerintah. Oleh karena itu, dibutunkan prediksi debit air yang layak didistribusi oleh PDAM dengan tujuan agar PDAM dapat menyesuaikan debit air yang layak didistribusi dengan kebutuhan pelanggan. Penelitian ini menggunakan metode *Extreme Learning Machine* (ELM) untuk memprediksi debit distribusi air menggunakan jenis data *single variable/time series* dan *multivariate*. Proses penerapan metode ELM yaitu dengan proses pelatihan, pengujian, denormalisasi, dan mengevaluasi hasil prediksi dengan menggunakan *Mean Percentage Absolute Error* (MAPE). Berdasarkan penerapan metode ELM dan pengujian yang dilakukan, diperoleh kondisi terbaik pada data *single variable* ketika menggunakan 7 *input neuron*, 4 *hidden neuron*, 20 data latih dan 5 data uji dengan menghasilkan rata-rata nilai MAPE 3.938%, sedangkan pada pengujian terhadap data *multivariate*, menghasilkan rata-rata nilai MAPE 13.081% dengan menggunakan 4 *hidden neuron*, 30 data latih dan 6 data uji.

Kata kunci: Prediksi, Distribusi PDAM, *Single Variable*, *Multivariate*, ELM, MAPE

ABSTRACT

Putri Indhira Utami P, Implementation of the Extreme Learning Machine (ELM) Method to Predict the Proper Debit of Water Distribution (Case Study: PDAM Gowa Regency, Makassar City)

Advisor: Muhammad Tanzil Furqon, S.Kom., M.CompSc and Ir. Sutrisno, M.T.

PDAM Gowa Regency, Makassar City is a company under the government that carries out the process of water production and continues to distribute the PDAM water to home residents. If there is a lot if water produced, it means theres is also a large amount of water that available for PDAM, so it can fulfill the public's requirement and can even to add customers. However, the seasonal change factor can take effect the discharge of water produced. So, the main problem is the uncertainty of water production which will certainly have an impact of the PDAM water distribution that will be distributed to home residents. But not all the water produced can be distributed because it has to go through several stages of water quality checking, so that the water that's distributed is in accordance with the standarts set by the government. Therefore, prediction of a proper flow of water distributed by PDAM is needed, with the aim that PDAM can adjust the proper flow of water distributed to customers. This research uses the Extreme Learning Machine (ELM) method to predict using single variable dan multivariate data types. The process of applying the ELM methods are normalizing, process of training and testing, denormalizing, and evaluating the prediction results using Mean Percentage Absolute Error (MAPE). Based on the application of the ELM method and the testing process, it produces the best conditions of single data variable when using 7 input neurons, 4 hidden neurons, 20 training data and 5 testing data to produced an average MAPE of 3.938%, while using the multivariate data, the average MAPE was 13.081% using 4 hidden neurons, 30 training data and 5 testing data.

Keywords: Prediction, PDAM Distribution, Single Variable, Multivariate, ELM, MAPE

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB 1 PENDAHULUAN

1.1 Latar Belakang

Sumber kehidupan bagi makhluk hidup di alam sangat banyak, salah satu sumber yang paling penting untuk kelangsungan makhuk hidup adalah air. Air berperan penting dalam proses kehidupan makhluk hidup, terutama bagi manusia. Selain itu, air juga tentunya sangat bermanfaat bagi makhluk hidup lainnya, seperti hewan dan tumbuhan yang sangat terbantu dalam proses perkembangan dan pertumbuhannya.

Air merupakan sumber kehidupan yang keberadaannya mudah ditemukan dan sangat berlimpah di permukaan bumi ini. Indonesia berada pada peringkat kelima di dunia yang menyatakan bahwa di Indonesia masih terdapat cukup besar cadangan air yang dapat digunakan untuk keberlangsungan hidup. Pada penelitian oleh (Prihatin, 2015) dijelaskan bahwa berdasarkan hasil riset Pusat Penelitian dan Pengembangan Sumber Daya Air Kementerian Pekerjaan Umum tahun 2009, disebutkan bahwa Indonesia masih memiliki cadangan air yang besar yaitu sebanyak 2.530 km^3. Dampak yang dirasakan terhadap besarnya cadangan air yaitu, penyebaran sumber daya air di Indonesia masih kurang merata, salah satu penyebabnya adalah pertumbuhan penduduk yang tidak merata pada beberapa daerah tertentu, seperti di Pulau Jawa yang memiliki penduduk sekitar 65% dari total seluruh penduduk di Indonesia namun potensi air yang dimiliki hanya 4,5%.

Secara umum, masyarakat Indonesia mempunyai dua pilihan untuk bisa mendapatkan air bersih, yaitu dengan menggunakan sumur atau berlangganan pada Perusahaan Daerah Air Minum (PDAM) di wilayah setempat. Seperti contoh pada penduduk yang bertempat tinggal di Kabupaten Gowa Kota Makassar umumnya berlangganan PDAM untuk memenuhi kebutuhan air bersih. Pada penelitian oleh (SUNARYO, 2015) dijelaskan bahwa PDAM Tirta Jeneberang Kabupaten Gowa merupakan perusahaan produksi air bersih yang memiliki visi untuk menyediakan air minum terbaik bagi masyarakat dengan melakukan pelayanan yang maksimal, dengan misi yang dimiliki yaitu meningkatkan mutu dan pelayanan demi kepuasan masyarakat.

Berdasarkan jenisnya, sumber air yang digunakan oleh PDAM dalam proses produksi terbagi atas 3 macam, yaitu sumber air yang berasal dari sungai, berasal dari air sumur, dan berasal dari sumber mata air. Berkaitan dengan hal tersebut, PDAM Kabupaten Gowa Makassar memperoleh sumber air yang berasal dari sungai, yaitu Sungai Jeneberang. Sebelum memenuhi kebutuhan air bersih pelanggan, terdapat dua tahap penting yang dilakukan oleh PDAM, yaitu tahap produksi kemudian dilanjutkan dengan tahap distribusi air bersih ke rumah penduduk.

Secara umum, definisi dari produksi air adalah banyaknya air bersih yang dihasilkan oleh PDAM, sedangkan definisi dari distribusi air adalah banyaknya air bersih yang tersalurkan ke masyarakat sekitar. Jika jumlah air yang diproduksi banyak, berarti ketersediaan air bagi PDAM banyak pula, sehingga bisa memenuhi

kebutuhan masyarakat bankan bisa untuk menambah pelanggan. Dalam tiap bulannya, PDAM Kabupaten Gowa Makassar melakukan proses produksi air yang kemudian akan disalurkan ke beberapa unit PDAM yang beraca di dalam kabupaten yang sama. Namun, faktor perubahan musim yang kadang terjadi sangat berpengaruh dalam jumlah air yang diproduksi. Sehingga masalah utama yang dihadapi oleh PDAM Kabupaten Gowa Makassar adalah tidak menentunya jumlah produksi air dikarenakan beberapa faktor yang tentunya berdampak pada jumlah distribusi PDAM yang akan disalurkan ke rumah penduduk. Terlebih lagi dalam jumlah produksi air oleh PDAM, tidak semua dapat didistribusikan dikarenakan harus melewati beberapa tahap pengecekan kualitas air agar air yang didistribusikan sesuai dengan standar yang telah ditetapkan oleh pemerintah. Berkaitan dengan hal tersebut, pihak PDAM juga akan mengalami masalah jika jumlah debit yang dapat didistribusikan kurang, permasalahan tersebut yakni pihak PDAM akan mengalami kerugian dikarenakan tidak dapat memenuhi kebutuhan pelanggan dan juga pihak PDAM harus mengeluarkan banyak biaya untuk melakukan pengolahan yang lebih optimal agar memaksimalkan debit air yang dapat didistribusi berdasarkan debit air yang sebelumnya telah diproduksi.

Sebelumnya terdapat penelitian yang membandingkan metode *Holt Winter's Exponential Smoothing* dan metode *Extreme Learning Machine* (ELM) terhadap peramalan penjualan semen. Berdasarkan penelitian yang dilakukan oleh (Dewi, 2018) terhadap penjualan semen domestik PT Semen Indonesia untuk tahun 2018 dengan data penjualan semen tahun 2006-2017, didapatkan perbandingan antara kedua metode, yaitu ketika menggunakan metode *Holt Winter's Exponential Smoothing* memperoleh tingkat kesalahan peramalan (*error*) untuk PT Semen Gresik sebesar 9,49 %, PT Semen Padang sebesar 8,04 %, dan PT Semen Tonasa sebesar 10,44 %, sedangkan ketika menggunakan metode *Extreme Learning Machine* (ELM) memperoleh tingkat kesalahan peramalan (*error*) untuk PT Semen Gresik sebesar 5,41%, PT Semen Padang sebesar 5,95%, dan PT Semen Tonasa sebesar 9,19%. Sehingga dapat disimpulkan bahwa penggunaan metode *Extreme Learning Machine* (ELM) dalam kasus peramalan penjualan semen pada PT Semen Indonesia lebih tepat digunakan karena memiliki nilai *error* yang lebih kecil daripada metode *Holt Winter's Exponential Smoothing*.

Terdapat penelitian lain yang membahas perbandingan antara metode *Extreme Learning Machine* (ELM) dan metode *Backpropagation* pada kasus peramalan laju inflasi di Indonesia. Pada penelitian oleh (Alfiyatin *et al.*, 2019) dikatakan bahwa metode *Extreme Learning Machine* (ELM) memiliki kelebihan, yaitu cepat dalam proses *training*. Pada pengujian penelitian ini dilakukan perbandingan antara kedua metode, yaitu ketika menggunakan metode *Extreme Learning Machine* (ELM) memiliki nilai kesalahan sebesar 0,0202008, sedangkan ketika menggunakan metode *Backpropagation* memiliki nilai kesalahan lebih besar yaitu 1,16035821. Dengan demikian dapat disimpulkan bahwa metode *Extreme Learning Machine* (ELM) cocok digunakan untuk kasus peramalan.

Berdasarkan latar belakang yang telah dijelaskan di atas, maka penulis akan melakukan penelitian terhadap prediksi jumlah debit air bersih yang layak

didistribusi menggunakan metode *Extreme Learning Machine* (ELM). Selain memprediksi distribusi air secara *time series*, juga dapat dilakukan prediksi distribusi air berdasarkan faktor-faktor yang mempengaruhi, seperti data produksi air, data pengecekan kualitas air, dan data distribusi air pada tahun-tahun sebelumnya. Sehingga data yang dimiliki pada kasus penelitian pada PDAM ini dapat digunakan untuk membandingkan hasil prediksi distribusi air ketika menggunakan *single variable* atau *time series* dan menggunakan *multivariable* atau asosiatif, yaitu dengan melibatkan fitur-fitur yang saling terikat. Prediksi distribusi air ini bermanfaat sebagai parameter untuk menentukan ketersediaan suplai air bersih dan juga untuk mengetahui berapa realisasi penerimaan apakah sebanding atau tidak dengan biaya yang dikeluarkan ketika memproduksi. Selain itu, dengan memprediksi jumlah air yang dapat didistribusi, PDAM juga dapat memprediksi jumlah pelanggan PDAM di masa yang akan datang.

1.2 Rumusan Masalah

Berdasarkan latar belakang pada penjelasan sebelumnya, didapatkan beberapa permasalahan yang dapat dirumuskan pada penelitian ini, yaitu:

1. Bagaimana mengimplementasikan metode *Extreme Learning Maching* (ELM) untuk memprediksi debit air yang layak didistribusi dengan menggunakan data *single variable* dan *multivariate*?
2. Bagaimana perbandingan tingkat kesalahan yang diperoleh ketika memprediksi debit air yang layak didistribusi dengan metode ELM pada jenis data *single variable* dan *multivariate*?

1.3 Tujuan

Berdasarkan permasalahan pada penjelasan sebelumnya, disimpulkan tujuan dari penelitian ini, yaitu:

1. Mengetahui penerapan metode *Extreme Learning Maching* (ELM) dalam memprediksi debit air yang layak didistribusi menggunakan data *single variable* dan *multivariate*.
2. Mengetahui perbandingan tingkat kesalahan prediksi metode *Extreme Learning Maching* (ELM) yang diperoleh pada saat menggunakan data jenis *single variable* dan *multivariate* dalam memprediksi debit air yang layak didistribusi dengan menggunakan metode *Mean Absolute Percent Error* (MAPE).

1.4 Manfaat

Manfaat yang didapatkan dari penelitian ini sebagai berikut:

1. Bagi Penulis

   Memahami dan menerapkan metode *Extreme Learning Maching* (ELM) dalam memprediksi debit air yang layak didistribusi dengan menggunakan data *single variable* dan *multivariate*.

2. Bagi Pembaca

   Sebagai bahan referensi dalam pembelajaran Jaringan Svaraf Tiruan dengan metode *Extreme Learning Maching* (ELM) yang menggunakan dua jenis data, yaitu data *single variable* dan data *multivariate*.

3. Bagi PDAM

   Memberikan kemudahan kepada petugas PDAM Kabupaten Gowa Makassar dalam memprediksi jumlah debit air yang layak didistribusi berdasarkan data.

1.5 Batasan Masalah

Batasan-batasan masalah pada penelitian ini sebagai berikut:

1. Permasalahan yang diteliti adalah jumlah debit air yang layak didistribusi oleh PDAM Kabupaten Gowa Makassar.
2. Penentuan jumlah debit air yang layak didistribusi dapat digunakan dengan prediksi *time series* dan juga dapat mempertimbangkan beberapa parameter lainnya, yaitu jumlah debit air yang diproduksi dan pengukuran pengecekan kelayakan kualitas air yang dilakukan dua tahap.
3. Data yang digunakan untuk penelitian merupakan data dari Laboratorium PDAM Kabupaten Gowa Makassar pada tahun 2016-2018.

1.6 Sistematika Pembahasan

Pada bagian ini bertujuan untuk memberikan gambaran berupa uraian dari penyusunan penelitian secara garis besar yang meliputi beberapa bab sebagai berikut:

BAB I PENDAHULUAN

Bab ini berisi tentang latar belakang dilaksanakannya penelitian, rumusan masalah yang akan diteliti, tujuan dan manfaat yang diharapkan akan tercapai, serta batasan masalah yang telah ditentukan dan sistematika pembahasan.

BAB II LANDASAN KEPUSTAKAAN

Bab ini berisi tentang teori-teori penunjang dan konsep dasar yang berkaitan dengan penelitian yang akan dilakukan. Pada bab ini, lebih terkhusus dalam membahas debit air yang diproduksi dan didistribusi oleh PDAM serta membahas

metode yang akan digunakan dalam penerapan komputasinya, yaitu metode *Extreme Learning Machine* (ELM).

BAB III METODOLOGI PENELITIAN

Bab ini berisi tentang metode dan langkah kerja yang dilakukan selama penelitian mulai dari awal hingga akhir dalam memprediksi jumlah debit air yang layak didistribusi menggunakan metode *Extreme Learning Machine* (ELM).

BAB IV PERANCANGAN

Bab ini berisi tentang proses perancangan yang dilakukan selama penelitian.

BAB V IMPLEMENTASI

Bab ini berisi tentang penyajian dan penjelasan implementasi metode dalam bentuk kode program untuk memprediksi jumlah debit air yang layak didistribusi menggunakan metode *Extreme Learning Machine* (ELM).

BAB VI PENGUJIAN DAN HASIL ANALISIS

Bab ini berisi tentang penyajian dan analisis terhadap hasil pengujian yang diperoleh setelah menerapkan metode *Extreme Learning Machine* (ELM) pada data *single variable* dan *multivariate*.

BAB VII PENUTUP

Bab ini berisi tentang penutup dari penelitian yang terdiri dari kesimpulan dan saran terhadap penelitian yang telah dikerjakan agar dapat bermanfaat bagi pengembangan untuk penelitian selanjutnya.

BAB 2 LANDASAN KEPUSTAKAAN

Bab ini menjelaskan mengenai teori-teori penunjang berupa kajian pustaka dan konsep dasar yang berkaitan dengan penelitian yang akan dilakukan, yaitu memprediksi debit air yang layak didistribusi oleh PDAM. Pada bab ini, pembahasan mengenai prediksi debit air yang layak didistribusi lebih difokuskan dalam penerapan dengan menggunakan metode *Extreme Learning Machine* (ELM). Selain itu pada bab ini terdapat pembahasan mengenai kajian pustaka yang merupakan hasil dari penelitian-penelitian sebelumnya yang tentunya berkaitan dengan penelitian yang akan dikerjakan pada studi kasus ini. Untuk menyusun dan melaksanakan penelitian ini juga dibutuhkan dasar teori, seperti konsep dasar dari air dan manajemen kualitas air. Kemudian terkait metode yang digunakan dalam penelitian, akan dijelaskan mengenai dasar teori dari Jaringan Syaraf Tiruan (JST), terkhususnya pada penjelasan mengenai metode *Extreme Learning Machine* (ELM) dan proses normalisasi data pada ELM. Penjelasan selanjutnya juga membahas mengenai teknik dalam memproses *data training* dan *data testing*, kemudian melakukan proses denormalisasi data, dan terakhir melakukan evaluasi terhadap metode ELM dengan melakukan perhitungan dengan metode *Mean Absolute Percentage Error* (MAPE).

2.1 Kajian Pustaka

Pada kajian pustaka ini membahas mengenai beberapa penelitian yang telah dilakukan sebelumnya dengan mengeluarkan hasil serta kesimpulan yang dapat dijadikan penunjang untuk penelitian yang akan dilakukan oleh penulis. Penjelasan singkat mengenai kajian pustaka dijelaskan pada Tabel 2.1.

Tabel 2.1 Kajian Pustaka

No.	Penulis	Objek (Masukan)	Metode (Proses)	Hasil (Keluaran)
1.	(Dewi, 2018)	- Hasil penjualan domestik PT Semen Gresik Januari 2006-Agustus 2017 - Hasil penjualan domestik PT Semen Padang Januari 2006-Agustus 2017 - Hasil penjualan domestic PT Semen Tonasa Januari 2006-Agustus 2017	Metode *Extreme Learning Machine* (ELM)	Penelitian ini menghasilkan sistem dengan keluaran prediksi berupa data penjualan semen pada tahun 2018 di tiga perusahaan semen Indonesia dengan tingkat kesalahan prediksi untuk PT Semen Gresik sebesar 5.41%, PT Semen Padang sebesar 5,95%,

				dan PT Semen Tonasa sebesar 9,19%.
2.	(Alfiyatin *et al.*, 2019)	- Data historis dengan analisis deret waktu mewakili parameter bulan sebelumnya, dua bulan sebelumnya dan tiga bulan sebelumnya. - Faktor eksternal, seperti IHK (Indeks Harga Konsumen), tingkat suku bunga, jumlah uang beredar, nilai tukar rupiah, nilai kredit dan nilai asset.	Metode *Extreme Learning Machine* (ELM)	Penelitian ini menghasilkan sistem dengan keluaran prediksi berupa tingkat inflasi di Indonesia dengan tingkat kesalahan prediksi sebesar 0,0202008.
3.	(Ashar, Cholissodin and Dewi, 2018)	- Permintaan konsumen dalam bentuk diameter produk, ketebalan produk, dan panjang produk, dan bobot atau *quantity request*. Modal yang berupa bobot bahan baku utama	Metode *Extreme Learning Machine* (ELM)	Penelitian ini menghasilkan sistem dengan keluaran prediksi berupa bobot produk pipa yang layak diproduksi dengan nilai *error* terkecil dengan rata-rata 0,00372.

4.	(Vong *et al.*, 2014)	- Air pollution data: PM10, SO2, NO2, and O3. - Meteorological data: AtmP, TEMP, RH, WS, RF, and SH.	Metode *Extreme Learning Machine* (ELM)	Penelitian ini menghasilkan sistem dengan keluaran prediksi berupa kelas atau level (*Good, Moderate, Severe*) dari *particular matters* (PM_{10}). Pada kasus ini, metode Metode *Extreme Learning Machine* (ELM) memiliki tingkat akurasi di atas 53,2% (untuk kasus level 2) dan 45,8% (untuk kasus level 3).

Berdasarkan penjelasan singkat mengenai kajian pustaka di atas, hasil yang diperoleh pada penelitian pertama yang dilakukan oleh Dewi (2018) mengenai prediksi data penjualan semen pada tahun 2018 di tiga perusahaan semen Indonesia dengan masukan yaitu hasil penjualan domestik oleh ketiga perusahaan pada Januari 2006 - Agustus 2017. Penelitian ini membandingkan hasil kerja metode *Extreme Learning Machine* (ELM) dengan metode *Holt Winter's Exponential Smoothing*, dengan hasil kesimpulan yaitu metode *Extreme Learning Machine* (ELM) memiliki nilai *error* yang lebih kecil sehingga metode ini lebih tepat digunakan dalam penelitian prediksi jumlah penjualan semen.

Pada penelitian kedua yang dilakukan oleh Adyan Nur Alfiyatin, Wayan Firdaus Mahmudy, Candra Fajri Ananda, dan Yusuf Priyo Anggodo (2018) mengenai peramalan laju inflas di Indonesia dengan masukan yaitu data historis dengan analisis deret waktu mewakili parameter bulan sebelumnya, dua bulan sebelumnya dan tiga bulan sebelumnya serta faktor eksternal, seperti IHK (Indeks Harga Konsumen), jumlah uang beredar, tingkat suku bunga, nilai kredit, nilai tukar rupiah dan nilai asset. Penelitian ini membandingkan hasil kerja metode *Extreme Learning Machine* (ELM) dengan metode *Backpropagation* dengan hasil pengujian yaitu metode ELM menghasilkan nilai *error* sebesar 0,0202008, sedangkan metode *Backpropagation* menghasilkan nilai *error* yang lebih besar daripada metode ELM yaitu sebesar 1,16035821. Berdasarkan hal tersebut dapat disimpulkan bahwa metode ELM sangat baik digunakan dalam kasus peramalan. Selain itu terdapat kelebihan lain ELM yaitu dapat melakukan proses pembelajaran (*training*) dengan cepat.

Pada penelitian ketiga yang dilakukan oleh Nirzha Maulidya Ashar, Imam Cholissodin, dan Candra Dewi (2018) mengenai prediksi jumlah produksi pipa yang layak dengan beberapa masukan seperti permintaan konsumen dalam bentuk diameter produk, ketebalan produk, dan panjang produk, dan bobot atau *quantity request* serta modal yang berupa bobot bahan baku utama. Penelitian ini melakukan pengujian dengan menggunakan 7 *hidden neuron*, 5 fitur, dan persentase perbandingan 80% dan *training* 20% data *testing* yang menghasilkan nilai *error* terkecil dengan rata-rata 0.00372 dengan menghasilkan selisih sekitar 1% dari data yang sebenarnya.

Pada penelitian keempat yang dilakukan oleh Chi-man Vong, Weng-fai Ip, Pak-kin Wong, dan Chi-chong Chiu (2014) mengenai prediksi kelas atau level dari *particular matters* (PM_{10}) dengan beberapa masukan seperti *Air pollution data and Meteorological data: particular matters* (PM_{10})*, sulfur dioxide* (SO_2)*, nitrogen dioxide* (NO_2)*, ozone* (O_3)*, atmospheric pressure (AtmP), temperature (TEMP), mean relative humidity (RH), wind speed (WS), rainfall (RF), and sunshine hour*. Penelitian ini membandingkan hasil kerja metode *Extreme Learning Machine* (ELM) dengan metode *Support Vector Machine* (SVM) dengan hasil pengujian yaitu metode SVM menghasilkan tingkat akurasi dalam memprediksi kasus level 1, tetapi pada prediksi kasus level 2 dan 3 metode SVM hanya mampu menghasilkan tingkat akurasi sebesar 28%, sedangkan metode ELM mampu menghasilkan tingkat akurasi di atas 53,2% pada prediksi kasus level 2 dan 45,8% pada prediksi kasus level 3. Perlu diketahui bahwa dalam kasus pada sistem ini, memprediksi level kedua dan ketiga lebih penting. Berdasarkan penelitian tersebut, disimpulkan bahwa metode SVM kurang tepat dalam menangani masalah keseimbangan data sehingga tidak bisa memprediksi secara akurat pada kasus level 2 dan 3, sedangkan metode ELM dapat menghasilkan tingkat akurasi yang lebih tinggi daripada SVM (dengan atau tanpa strategi keseimbangan data). Oleh karena itu, dalam penelitian ini metode ELM lebih unggul daripada metode SVM.

2.2 Air

Air merupakan sumber kehidupan yang keberadaannya sangat mudah ditemukan dan sangat berlimpah di permukaan bumi ini. Indonesia berada pada posisi kelima teratas di dunia yang menyatakan bahwa di Indonesia masih terdapat cukup besar cadangan air yang dapat digunakan untuk keberlangsungan hidup. Berdasarkan hasil riset Pusat Penelitian dan Pengembangan Sumber Daya Air Kementerian Pekerjaan Umum tahun 2009, dijelaskan oleh (Prihatin, 2015) bahwa Indonesia masih memiliki cadangan air yang besar yaitu sebanyak 2.530 km^3. Namun demikian, penyebaran sumber daya air di Indonesia masih kurang merata, salah satu penyebabnya yaitu pertumbuhan penduduk yang juga tidak merata pada beberapa daerah tertentu, seperti di Pulau Jawa yang memiliki penduduk sekitar 65% dari total seluruh penduduk di Indonesia namun potensi air yang dimiliki hanya 4,5%.

Secara umum, masyarakat Indonesia mempunyai dua pilihan untuk bisa mendapatkan air bersih, yaitu dengan menggunakan sumur atau berlangganan

pada PDAM (Perusahaan Daerah Air Minum) setempat. Berdasarkan jenisnya, sumber air yang digunakan oleh PDAM dalam proses produksi terbagi atas 3 macam, yaitu sumber air yang berasal dari sungai, berasal dari air sumur, dan berasal dari sumber mata air.

2.2.1 Debit Air

Debit merupakan volume air yang mengalir dalam satuan waktu tertentu yang dinyatakan dalam satuan meter kubik perdetik (m^3/detik) atau liter per detik (1/detik). Menurut (Sumani and Tadulako, 2018), memperoleh debit aliran sungai dapat dilakukan dengan cara mengukur kecepatan aliran sungai dari sungai terpilih dalam suatu Daerah Aliran Sungai (DAS), dengan melakukan perhitungan yaitu mengalikan kecepatan aliran sungai dengan luas penampang sungai dapat diperoleh hasil yaitu besanya debit air sungai yang diperoleh dari suatu DAS.

Terdapat 2 macam debit yang dapat diukur pada PDAM, yaitu debit air yang diproduksi dan debit air yang didistribusi. Secara umum, definisi dari produksi air adalah banyaknya air bersih yang dihasilkan oleh PDAM, sedangkan definisi dari distribusi air adalah banyaknya air bersih yang tersalurkan ke masyarakat sekitar. Setelah PDAM berhasil memproduksi air, maka langkah selanjutnya yang dilakukan sebelum didistribusi adalah pengecekan kualitas air. Air yang sebelumnya telah diproduksi tidak langsung didistribusikan ke masyarakat, melainkan dilakukan terlebih dahulu pengecekan kualitas airnya, kemudian diolah sehingga menjadi air bersih yang layak didistribusikan kepada masyarakat.

2.2.2 Manajemen Kualitas Air

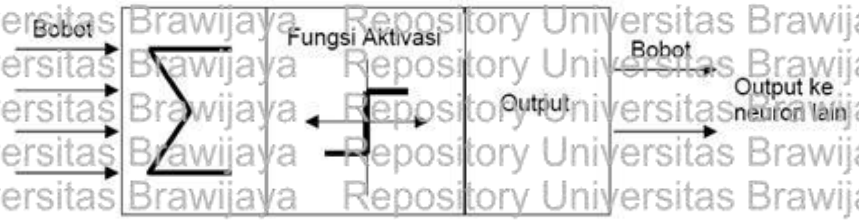
Dalam memproduksi air bersih, PDAM melakukan proses pengecekan kualitas air terlebih dahulu untuk memastikan apakah air tersebut benar layak didistribusi yang kemudian akan dikonsumsi oleh masyarakat sekitar. Dalam proses pengecekan kualitas air dilakukan dengan 2 tahap, yang pertama mengecek kulitas pada air baku dan yang keuda mengecek kualitas pada air bersih. Air yang baru saja diproduksi disebut dengan air baku, sedangkan hasil dari pengolahan air baku disebut dengan air bersih. Air bersih ini yang kemudian layak dikonsumsi oleh masyarakat karena status kelayakan kualitasnya sudah teruji.

Pengujian kualitas air pertama-tama dilakukan pada air baku, yaitu dengan mengukur tingkat kekeruhan, pH, dan alkalinitas. Setelah itu, air baku akan diolah menjadi air bersih dengan melakukan penjaringan dan menambahkan zat yang dapat membuat kualitas air menjadi baik. Setelah didapatkan air bersin, masin perlu lagi diuji kelayakannya dengan mengukur tingkat kekeruhan, pH, klor bebas, dan alkalinitas. Setelah proses pengujian kelayakan air bersih, dilakukan kembali pengecekan agar dapat menentukan apakah parameter tiap air bersih tersebut sudah sesuai dengan standar yang ditetapkan oleh pemerintah atau belum. Pengecekan dilakukan dengan mencocokkan data hasil uji kelayakan air bersih dengan peraturan persyaratan kualitas air yang telah dibuat oleh Menteri Kesehatan Republik Indonesia. Dalam peraturan tersebut terdapat jenis parameter kualitas air, satuan parameter, dan juga kadar maksimum yang

diperbolehkan untuk air tersebut dapat dikonsumsi oleh masyarakat. Ketika seluruh proses telah selesai dan parameter kualitas air terpenuhi, maka debit air yang tersisa sudah dikatakan layak untuk didistribusi yang kemudian akan dikonsumsi oleh masyarakat.

2.3 Jaringan Syaraf Tiruan

Salah satu representasi yang dapat digunakan dalam melakukan prediksi atau peramalan ialah dengan menggunakan representasi dari jaringan syaraf tiruan. Jaringan syaraf merupakan suatu representasi yang dibuat oleh manusia untuk melakukan proses pembelajaran pada otak manusia. Jaringan syaraf tiruan ini kemudian diharapkan mampu menyelesaikan sejumlah proses perhitungan selama melakukan proses pembelajaran yang diterapkan menggunakan implementasi dari program komputer. Menurut (Sel, 1940), dari awal ditemukan hingga sekarang, jaringan syaraf tiruan telah mengalami beberapa tahap perkembangan yang dimulai dari tahun 1940-an. Dalam struktur penyusunannya, jaringan syaraf terdiri dari beberapa neuron yang kemudian neuron-neuron itu akan mentransformasikan informasi menuju ke neuron-neuron yang lain. Dalam konsep jaringan syaraf, hubungan antara neuron-neuron yang mentransformasikan informasi kemudian diterima oleh neuron-neuron yang lain disebut dengan istilah bobot. Struktur neuron jaringan syaraf ditunjukkan pada gambar 2.1.



Gambar 2.1 Struktur Neuron Jaringan Syaraf

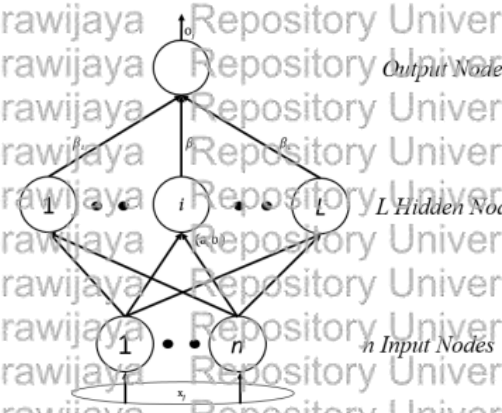
Sumber: (Sel, 1940)

Penjelasan mengenai gambar di atas yaitu terdapat informasi (disebut dengan: input) yang berasal dari neuron-neuron yang kemudian dikirim ke neuron lain dengan bobot awal tertentu. Setelah itu, input mengalami proses pada suatu fungsi perambatan dengan menjumlahkan seluruh nilai bobot yang telah masuk. Kemudian hasil dari penjumlahan tadi akan dibandingkan dengan nilai *threshold* tertentu menggunakan fungsi aktivasi dari setiap neuron. Terdapat dua kondisi yang memungkinkan setelah dilakukannya fungsi aktivasi yang dikemukakan oleh (Sel, 1940), yaitu jika nilai inputnya melewati nilai ambang tertentu, maka neuron tersebut akan diaktifkan, dan sebaliknya jika nilai inputnya kurang dari nilai ambang, maka neuron tersebut tidak akan diaktifkan.

2.3.1 Arsitektur Jaringan

Pada jaringan syaraf tiruan, terdapat kumpulan lapisan yang disebut dengan neuron. Jalur penyebaran informasi pada jaringan syaraf tiruan dimulai

dari lapisan *input* yang akan beraknir pada lapisan *output*, namun sebelumnya juga melewati suatu lapisan tersembunyi yang disebut dengan *hidden layer* (Sel, 1940). Fungsi dari *hidden layer* yaitu membantu jaringan agar dapat lebih banyak mengenali pola inputan dibandingkan ketika terdapat jaringan namun tidak memiliki *hidden layer* (Monika *et al.*, 2019). Salah satu arsitektur yang digunakan dalam proses penyelesaian masalah pada studi kasus penelitian ini yaitu jaringan syaraf tiruan dengan banyak lapisan atau disebut dengan *multilayer net*. Arsitektur jaringan syaraf tiruan *multilayer net* dapat dilihat pada Gambar 2.2.



Gambar 2.2 Arsitektur JST *Multilayer Net*

Sumber: (Riandri *et al.*, 2018)

Arsitektur jaringan ini disebut memiliki banyak lapisan karena memiliki 1 atau lebih lapisan tersembunyi (*hidden layer*) yang posisinya berada di antara lapisan masukan dan lapisan keluaran seperti yang terlihat pada Gambar 2.2 di atas. Menurut (Sel, 1940), arsitektur jenis ini memiliki perbedaan dibandingkan dengan arsitektur jaringan dengan lapisan tunggal, yaitu jaringan *multilayer net* ini dapat menyelesaikan masalah yang sulit dengan menggunakan teknik pembelajaran yang juga lebih rumit, namun terbukti lebih sukses untuk menyelesaikan permasalahan pada beberapa contoh kasus.

2.3.2 Fungsi Aktivasi

Penerapan fungsi aktivasi pada penelitian ini digunakan untuk menentukan hasil keluaran pada lapisan *output* neuron. Dalam penerapan jaringan syaraf tiruan pada penelitian ini menggunakan salah satu jenis fungsi aktivasi yang popular digunakan, yaitu fungsi sigmoid biner. Fungsi sigmoid biner ini memiliki ciri fungsi aktivasi yang baik karena telah memenuhi banyak kriteria, yaitu terdiferensial dengan mudah, kontinyu, dan merupakan fungsi yang tidak turun. Fungsi sigmoid biner ini memiliki *range* 0 hingga 1. Persamaan 2.1 merupakan persamaan fungsi sigmoid biner yang digunakan (Siang, 2005).

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

2.4 Metode Extreme Learning Machine (ELM)

Metode *Extreme Learning Machine* (ELM) merupakan salah satu metode pelatihan dari konsep representasi Jaringan Syaraf Tiruan (JST). Metode ELM merupakan JST *feedforward* yang memiliki *single hidden layer* atau yang biasa disebut dengan *Single Hidden Layer Feedforward Neural Networks* (SLFNs) (Sun *et al.*, 2008). Metode ELM dibuat dengan tujuan mengatasi masalan yang disebabkan oleh jaringan syaraf tiruan *feeaforward* terkhususnya dalam bidang *learning speed*. Salah satu pengaruh dari proses *learning speed* adalah terkait inputan bobot dan bias. Pada metode *Conventional gradient based learning algorithm*, seperti *backpropagration* (BP) dan *Lavenberg Marquadt* (LM) sebagai variannya, semua metode JST tersebut ditentukan secara manual. Sedangkan pada metode ELM parameter input bobot dan bias dipilih secara acak, sehingga dapat disimpulkan bahwa metode ELM mempunyai *learning speed* yang baik dan dapat memberikan performa yang baik (Huang, Zhu and Siew, 2006).

Dalam penerapan algoritma ELM pertama-tama akan melakukan proses pelatihan yang diakhiri dengan melakukan prediksi. Data yang digunakan pada proses pelatihan merupakan data yang bisa menggambarkan masalah dengan spesifik, yaitu kumpulan data yang terdiri dari hasil aktual dan faktor-faktor yang mempengaruhi suatu masalah. Selama proses pelatihan ELM berlangsung, (Ding *et al.*, 2015) menyebutkan bahwa kumpulan data tersebut melalui beberapa iterasi guna menyelesaikan proses pembelajaran. Kemudian, dengan diperolehnya kumpulan data yang sudah dilatih menggunakan algoritma ELM akan mampu untuk memprediksi dengan menggunakan inputan dan kumpulan data lain yang strukturnya mirip dengan faktor-faktor yang berkaitan pada masalah yang akan diselesaikan.

2.4.1 Normalisasi Data

Normalisasi data merupakan salah satu proses yang terdapat dalam konsep *Data Mining*. Normalisasi data dilakukan jika memiliki *range* yang jauh antara nilai atribut pada kumpulan sebuah data (*dataset*). *Dataset* yang telah dinormalisasi akan menghasilkan *output* yang kecil sesuai dengan skala yang diinginkan. Menurut (Jain and Bhandare, 2014), terdapat banyak metode yang dapat digunakan dalam melakukan normalisasi data, seperti *min-max normalization*, *z-score normalization*, dan *normalization by decimal scaling*.

2.4.1.1 Min Max Normalization

Salah satu metode yang bisa digunakan untuk melakukan proses normalisasi data adalah *Min Max Normalization*. Metode *Min-Max* merupakan sebuah teknik normalisasi data yang sederhana dengan melakukan proses transformasi linier terhadap data yang sebenarnya. Hasil dari proses *Min-Max Normalization* mengeluarkan perbandingan nilai, yaitu nilai sebelum dan sesudah dilakukannya proses normalisasi data yang menyebabkan keseimbangan data karena menghasilkan *output* dengan skala nilai yang kecil, yaitu pada penelitian ini menghasilkan *range* nilai mulai dari 0 hingga 1. Proses perhitungan normalisasi data dengan menggunakan metode *Min-Max* dengan *range* (0,1) tertera pada Persamaan 2.2 (Jain and Bhandare, 2014).

$$d' = \frac{d - \min(p)}{\max(p) - \min(p)} \quad (2.2)$$

Keterangan pada persamaan:

d' = Hasil transformasi suatu nilai yang telah dinormalisasi

d = Nilai data aktual yang akan dinormalisasi

$\min(p)$ = Nilai terkecil pada data fitur p

$\max(p)$ = Nilai terbesar pada data fitur p

2.4.2 Proses *Training*

Pada penelitian yang dikemukakan oleh (Cholissodin and Riyandani, 2018) menjelaskan langkah-langkah proses *training* dalam penerapan algoritme ELM seperti di bawah ini.

Langkah 1:

Menentukan nilai bobot masukan (W_{jk}) dan bias masukan (b) dengan ukutan matriks b yaitu $[1 \times j]$. Pada langkah pertama ini peneliti menggunakan nilai bobot dan bias masukan secara acak dengan *range* nilai 0 sampai 1. Menurut (Julpan, Nababan and Zarlis, 2015), penentuan *range* nilai yang digunakan dalam hal ini merupakan nilai objek yang akan diteliti, sehingga nilai masukan harus disesuaikan dengan fungsi aktivasi yang akan digunakan agar hasil yang diperoleh di akhir dapat disesuaikan dengan skala nilai fungsi aktivasi yang digunakan.

Langkah 2:

Melakukan perhitungan matriks *output hidden layer* menggunakan Persamaan 2.3.

$$H = 1/(1 + \exp(-(X.W^T + b)) \quad (2.3)$$

Keterangan pada persamaan:

H = Hasil perhitungan matriks *outout hidden layer*

exp = Nilai eksponensial

X = Data *training*

W^T = Nilai bobot yang di*transpose*

b = Nilai bias

Langkah 3:

Melakukan perhitungan matriks *Moore Penrose Pseudo-Inverse* menggunakan Persamaan 2.4.

$$H^+ = (H^T . H)^{-1} . H^T \quad (2.4)$$

Keterangan pada persamaan:

H^+ = Hasil perhitungan matriks *Moore Penrose Pseudo-Inverse*

H^T = Matriks H yang di*transpose*

H = Matriks H *output hidden layer*

Langkah 4:

Menghitung matriks *output weight* berdasarkan perhitungan sebelumnya, yaitu perhitungan matriks *Moore Penrose Pseudo-Inverse* dengan menggunakan Persamaan 2.5 yang dikemukakan oleh (Ding *et al.*, 2015).

$$\hat{\beta} = H^+ . T \quad (2.5)$$

Keterangan pada persamaan:

$\hat{\beta}$ = Hasil matriks *output weight*

H^+ = Hasil matriks *Moore Penrose Pseudo-Inverse*

T = Nilai target

2.4.3 Proses *Testing*

Setelah dilakukan proses *training*, selanjutnya dilakukan proses *testing* yang bertujuan untuk mengevaluasi kinerja dari metode ELM dengan melakukan perbandingan pada data dari hasil proses *training* sebelumnya. (Cholissodin and Riyandan, 2018) mengemukakan langkah-langkah dari proses *testing* pada metode ELM sebagai berikut.

Langkah 1:

Inisialisasi *input weight* (W_{ik}) dan bias (b) yang sebelumnya telah didapatkan dari proses *training*.

Langkah 2:

Melakukan perhitungan matriks *output hidden layer* pada data *testing* dengan menggunakan Persamaan 2.6.

$$H = 1/(1 + \exp(-(X_{test} . W^T + b)) \quad (2.6)$$

Keterangan pada persamaan:

H = Hasil perhitungan matriks *outout hidden layer*

exp = Nilai eksponensial

X_{test} = Data *testing*

W^T = Nilai bobot yang di*transpose*

b = Nilai bias

Langkah 3:

Menghitung nilai hasil prediksi dari target menggunakan Persamaan 2.7.

$$\check{T} = H.\hat{\beta} \quad (2.7)$$

Keterangan pada persamaan:

$\check{T}$ = Nilai keluaran yang merupakan hasil prediksi dari nilai target

H = Hasil matriks *output hidden layer* pada proses *testing*

$\hat{\beta}$ = Nilai matriks *output weight* yang telah diperoleh dari proses *training*

Langkah 4:

Menghitung nilai evaluasi untuk mengetahui tingkat kesalahan ketika memprediksi suatu target.

2.4.4 Proses Denormalisasi

Setelah seluruh tahap telah dilakukan, selanjutnya melakukan pembangkitan nilai yang telah dinormalisasi menjadi nilai asli yang disebut dengan proses denormalisasi. Untuk perhitungan proses denormalisasi dapat dilakukan dengan Persamaan 2.8.

$$a = a'(\max - \min) + min \quad (2.8)$$

Keterangan pada persamaan:

d' = hasil prediksi sebelum didenormalisasi

d = nilai asli setelah dilakukan proses denormalisasi

min = nilai terkecil pada data fitur P

max = nilai terbesar pada data firur P

2.4.5 *Mean Absolute Percentage Error* (MAPE)

Mean Absolute Percentage Error (MAPE) merupakan salah satu metode yang dilakukan untuk melakukan evaluasi pada algoritma ELM, yaitu dengan mengevaluasi hasil akhir dari prediksi dengan mengukur akurasi. Untuk melakukan perhitungan MAPE dapat menggunakan Persamaan 2.9 (Nugroho and Purqon, 2015).

$$MAPE = \frac{1}{n}\sum_{i=1}^{l}\left|\frac{y_i - \bar{y}_i}{y_i}\right| \times 100 \quad (2.9)$$

Keterangan pada persamaan:

n = Banyaknya data

i = [1,2,...,l] l merupakan keseluruhan jumlah data

y_i = Nilai aktual

$\bar{y}_i$ = Nilai hasil prediksi

BAB 3 METODOLOGI

Pada bab ini berisi tentang tahapan-tahapan dalam penerapan metode yang dilakukan oleh penulis selama penelitian mulai dari awal hingga akhir dalam memprediksi jumlah debit air yang layak didistribusi menggunakan metode *Extreme Learning Machine* (ELM). Adapun tahapan-tahapan pada Bab Metodologi akan dijelaskan di bawah ini.

3.1 Tipe Penelitian

Pada studi kasus kali ini peneliti menerapkan tipe penelitian yang bersifat non-implementatif analitik, yaitu penelitian yang dilakukan dengan cara menganalisis hubungan antar elemen dalam objek yang akan diteliti. Dalam penelitian ini, akan menganalisis hubungan antar elemen, meliputi data debit distribusi air dengan jenis data *single variable/time series* yang hanya memiliki 1 objek data dan dibandingkan dengan jenis data *multivariate* yang memiliki beberapa fitur yang dijadikan parameter dalam menghasilkan debit air yang didistrbusi.

3.2 Objek Penelitian

Objek yang dijadikan sebagai penelitian ini merupakan data dari Laboratorium PDAM Kabupaten Gowa Makassar. Data tersebut merupakan data bulanan selama 3 tahun yang dimulai dari tahun 2016-2018. Dalam data bulanan tersebut terdapat data debit produksi air, data hasil pengukuran dari pengecekan kualitas kelayakan air bersih, dan data debit air yang layak didistribusi.

3.3 Peralatan Pendukung

Spesifikasi peralatan pendukung dalam perangkat keras yang digunakan oleh penulis dalam melaksanakan penelitian ini sebagai berikut:

1. *Processor: Intel® Core™ i5-6200U CPU @ 2.30Ghz 2.40 Ghz*
2. *Installed RAM: 4.00 GB*
3. *System type: 64-bit operating system, x64-based processor*
4. *Monitor*
5. *Keyboard*

Spesifikasi peralatan pendukung dalam perangkat lunak yang digunakan oleh penulis dalam melaksanakan penelitian ini sebagai berikut:

1. *Spesification Windows: Windows 10 Home Single Language*
2. *Spyder* sebagai *Integrated Development Environment*
3. *Python* 3.6.4 sebagai Bahasa Pemrograman
4. *Numpy* dan *Pandas* sebagai *Library* yang digunakan

3.4 Lokasi Penelitian

Penelitian prediksi distribusi air yang layak didistribusi ini dilaksanakan di PDAM Kabupaten Gowa Makassar dan Fakultas Ilmu Komputer Universitas Brawijaya, Malang, Jawa Timur.

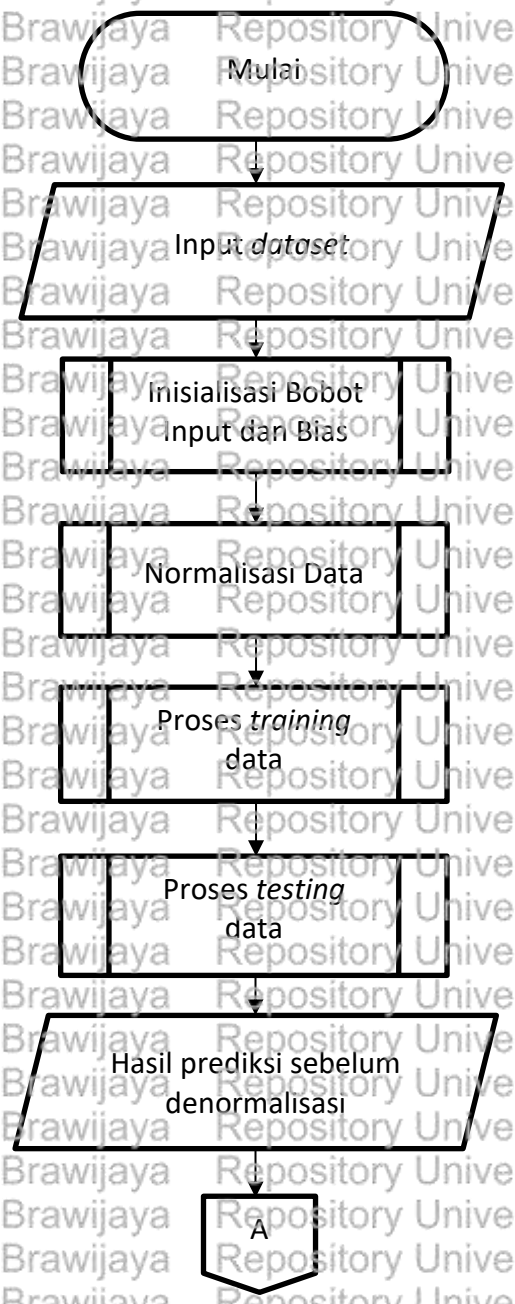
3.5 Teknik Pengumpulan Data

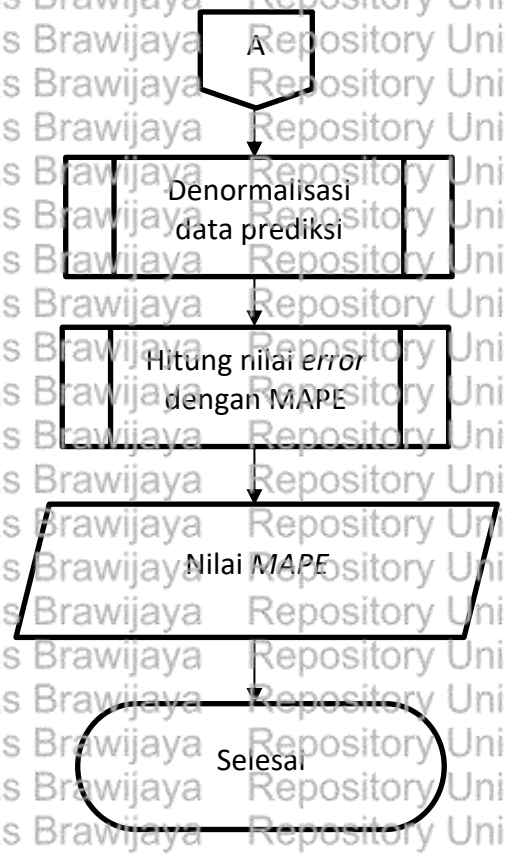
Pengumpulan data yang dilakukan yaitu melakukan riset ke Laboratorium PDAM Kabupaten Gowa Makassar. Data yang dikumpulkan merupakan data bulanan selama 3 tahun terakhir mulai dari tahun 2016-2018, yang meliputi data debit produksi air, data hasil pengukuran dari pengecekan kualitas kelayakan air bersih, dan data debit air yang layak didistribusi. Data terkait pengukuran kelayakan air bersih terdiri dari tingkat kekeruhan, pH, klor bebas, dan alkalinitas pada air keruh dan air bersih. Dengan demikian, data yang diperoleh sebanyak 36 data.

BAB 4 PERANCANGAN

Pada bab ini berisi tentang tahapan-tahapan dalam pengembangan sistem untuk memprediksi debit air yang layak didistribusi menggunakan *multivariate* dan *single variable* dengan metode ELM. Tahapan dalam pengembangan sistem yang dilakukan diijelaskan pada poin selanjutnya.

4.1 Diagram Alir Sistem Prediksi menggunakan Metode ELM





Gambar 4.1 Diagram Alir Sistem Prediksi menggunakan Metode ELM

Pada diagram alir di atas dijelaskan mengenai prediksi metode ELM. Adapun langkah-langkah berdasarkan Gambar 4.1 sebagai berikut:

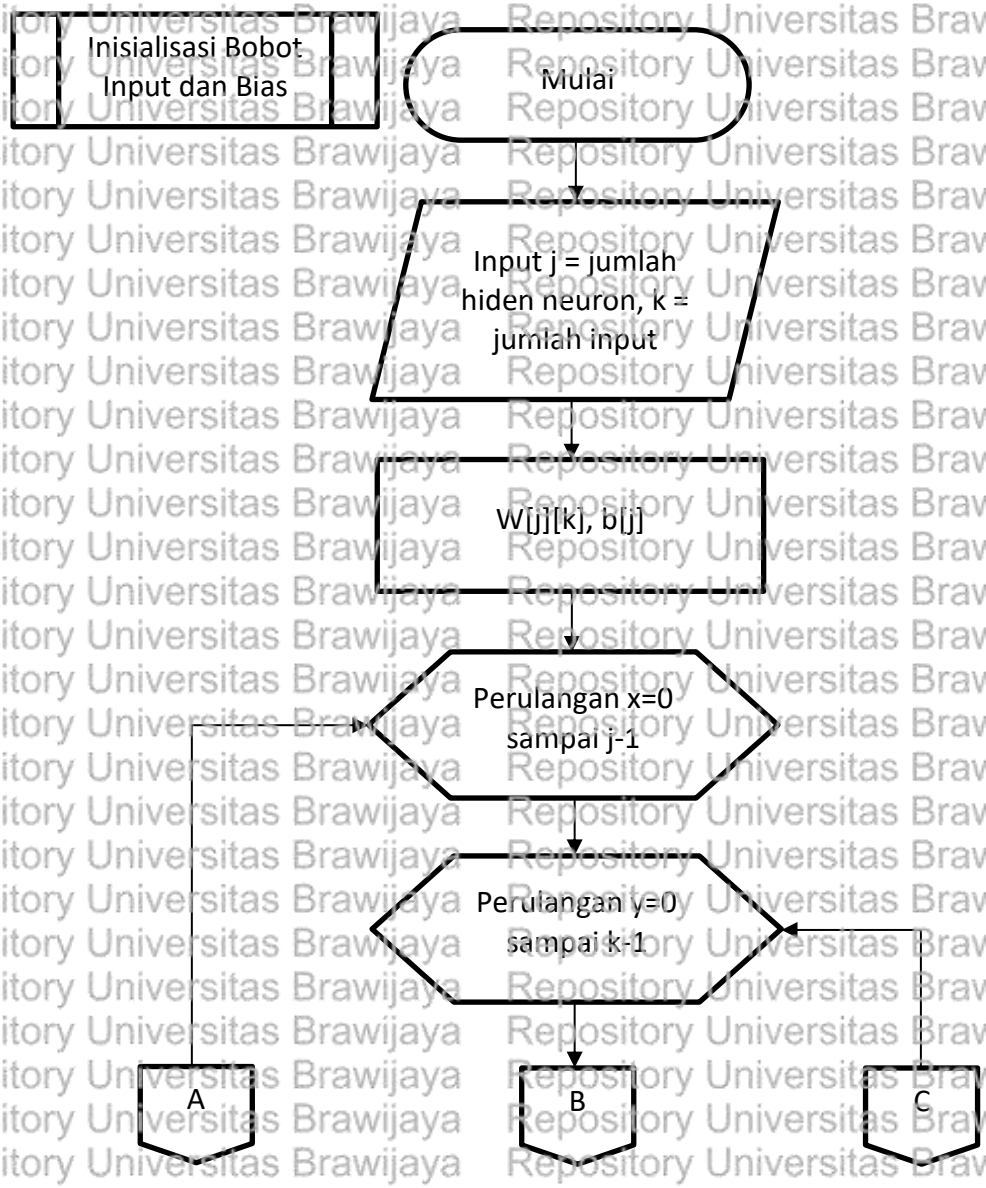
a. Langkah pertama pada penerapan metode ELM yaitu memasukkan *dataset*, yang dibagi menjadi data latih dan data uji. Dalam penggunaan multivariate akan dimasukkan seluruh fitur yang ada pada *dataset*, yaitu debit produksi air, kekeruhan air baku, pH air baku, alkalinitas air baku, kekeruhan air bersih, pH air bersih, klor bebas air bersih, dan alkalinitas air bersih. Untuk atribut debit distribusi air dijadikan sebagai target yang akan diprediksi.
b. Langkah kedua masuk ke tahap inisialisasi bobot *input* dan bias secara acak, namun sebelumnya ditentukan jumlah *input neuron* (k) dan jumlah *hidden neuron* (j), yang selanjutnya membentuk matriks bobot *input* [W] × k].
c. Langkah ketiga dilakukan normalisasi data pada tiap fitur dikarenakan rentang dari tiap nilai pada fitur yang tidak sama. Proses normalisasi dilakukan menggunakan metode *Min-Max Normalization* dengan rentang 0 sampai dengan 1.
d. Langkah keempat dilakukan proses *training*, yang meliputi perhitungan inisialisasi *output hidden layer*, fungsi aktivasi *output hidden layer*, matriks *Moore Penrose Pseudo-Inverse*, dan nilai *output weight*.
e. Langkah kelima dilakukan proses *testing* data, yang meliputi perhitungan matriks inisialisasi *output hidden layer*, fungsi aktivasi *output hidden* layer, dan

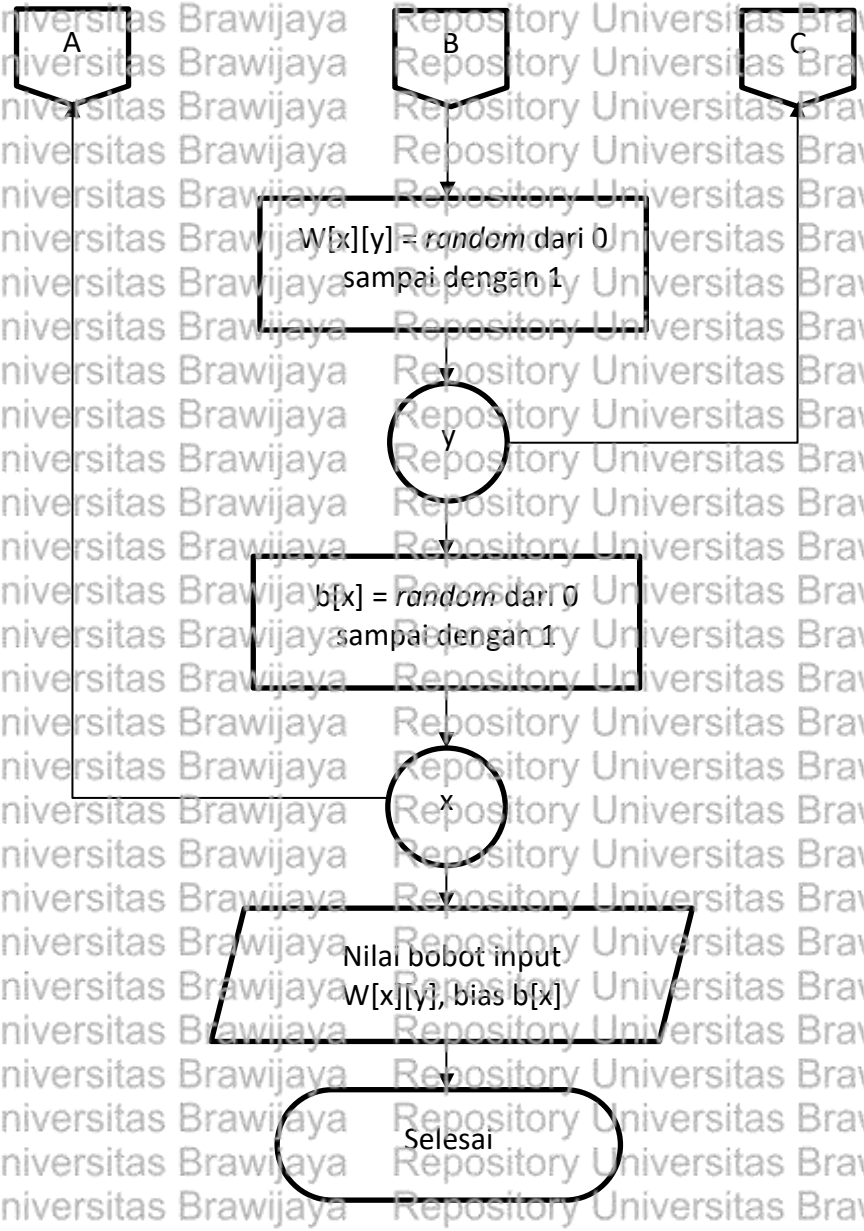
melakukan pernitungan prediksi dengan menggunakan nilai *output weight* yang sebelumnya telah didapatkan dari proses *training* data.

f. Setelah mendapatkan hasil prediksi, langkah keenam yang dilakukan yaitu denormalisasi data prediksi.

g. Langkah terakhir, yaitu menghitung nilai *error* dengan menggunakan MAPE. Representasi dari nilai *error* yang didapatkan digunakan untuk mengukur tingkat kesalahan dari perbandingan data prediksi dengan data aktual.

4.1.1 Diagram Alir Inisialisasi Bobot Input dan Bias





Gambar 4.2 Diagram Alir Inisialisasi Bobot Input dan Bias

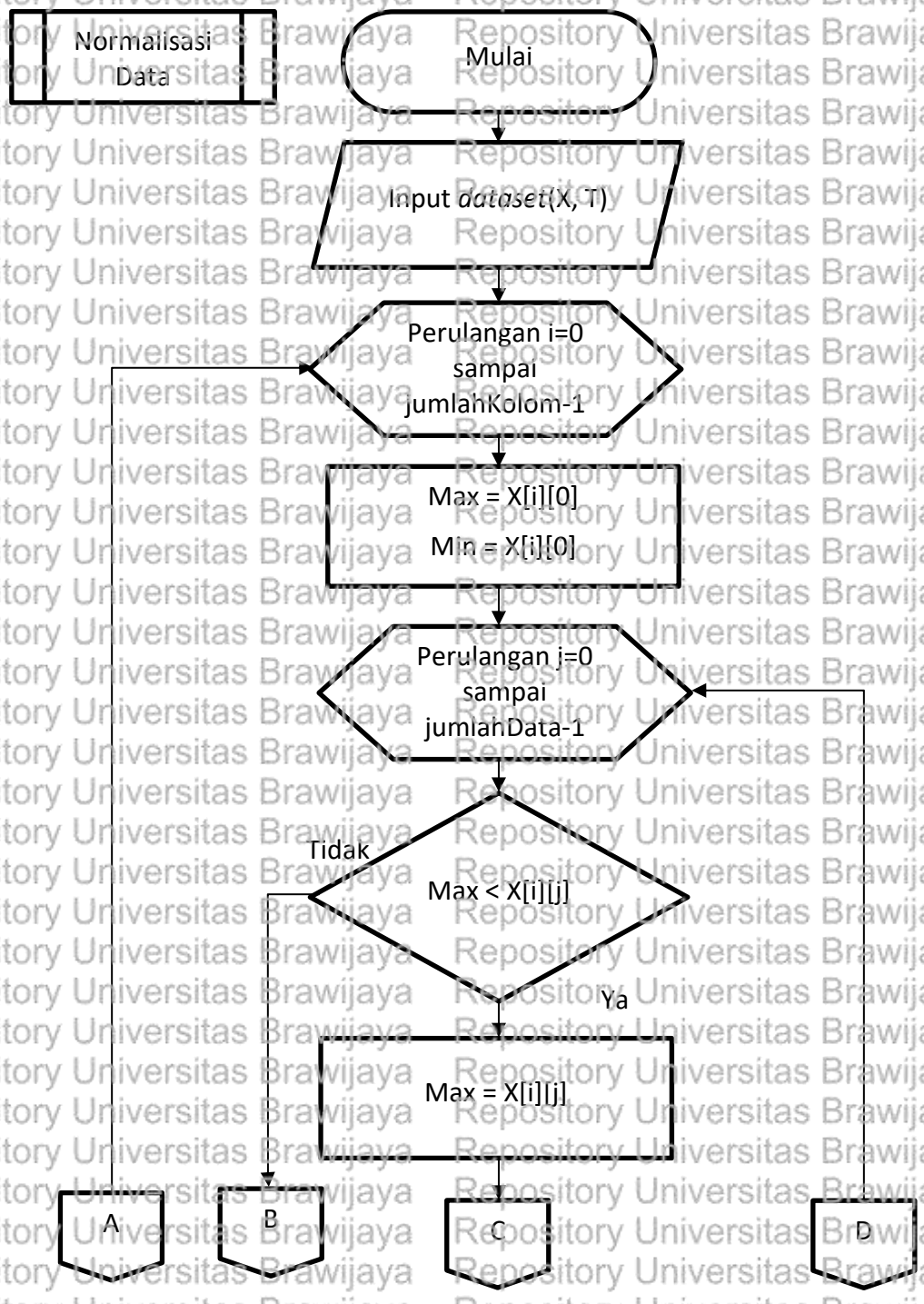
Pada diagram alir di atas dijelaskan mengenai alur dalam inisialisasi bobot input dan bias pada metode ELM. Adapun langkah-langkah inisialisasi bobot input dan bias berdasarkan Gambar 4.2 sebagai berikut:

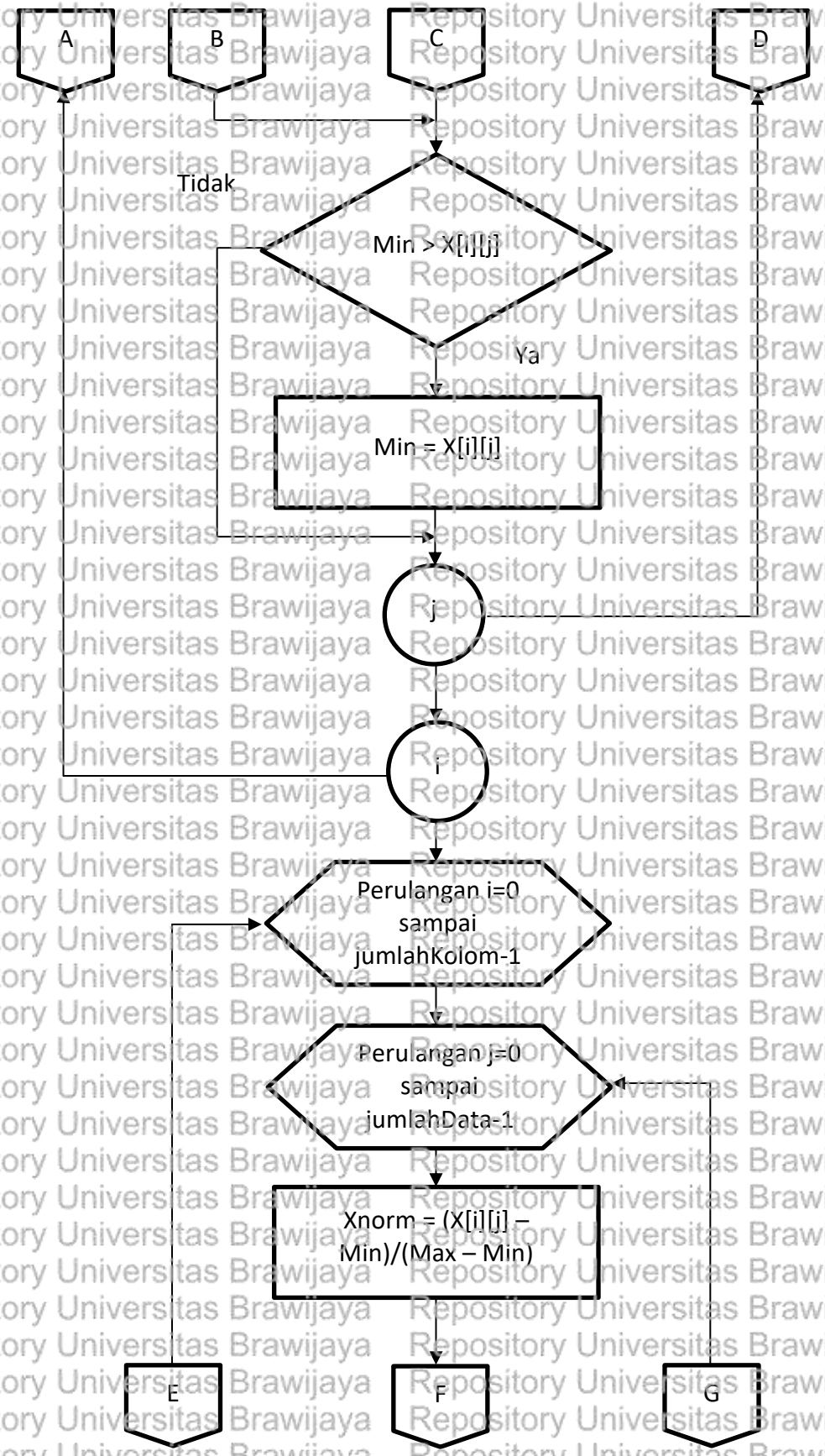
a. Langkah pertama dalam menentukan bobot input dan bias adalah menentukan jumlah *input neuron* (k) dan jumlah *hidden neuron* (j). yang selanjutnya membentuk matriks bobot input [Wj × k].
b. Langkah kedua dengan membentuk matriks berdasarkan *input neuron* (k) dan jumlah *hidden neuron* (j) yang menghasilkan matriks bobot input [Wj × k]
c. Langkah ketiga melakukan perulangan untuk x=0 sampai dengan j-1 untuk membuat baris (j) pada matriks bobot input, kemudian melakukan perulangan y=0 sampai dengan k-1 untuk membuat kolom (k) pada matriks bobot input.

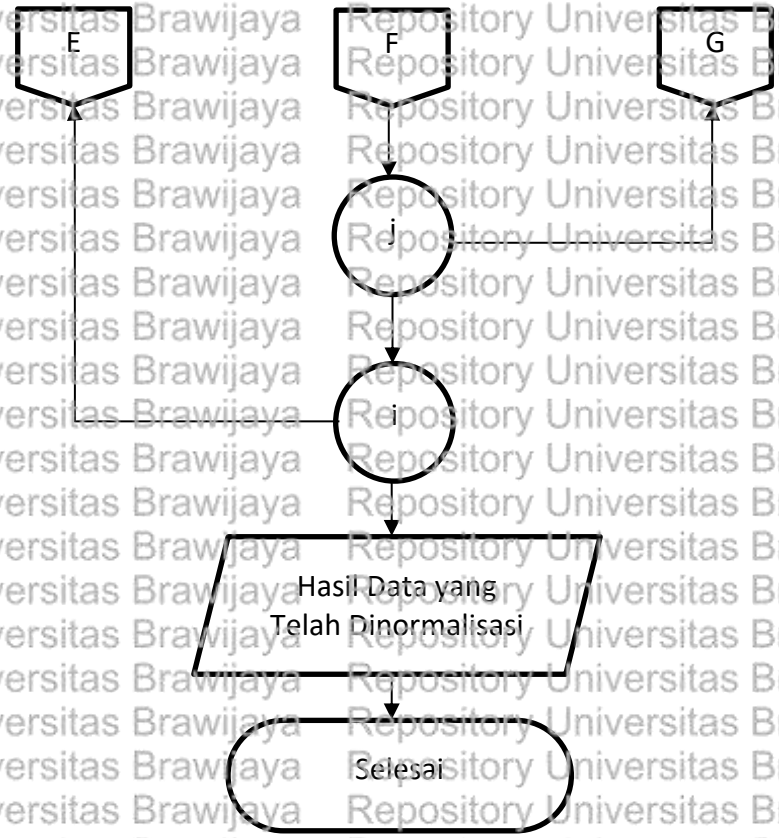
d. Langkah keempat dilakukan inisialisasi nilai *random* pada perulangan x dan y mulai dari 0 sampai dengan 1 untuk membentuk matriks bobot input [Wj × k] yang kemudian disimpan dalam matriks W[x][y].

e. Berdasarkan jumlah *hidden neuron* (k), maka terbentuklah nilai bias yaitu b[k], kemudian langkah kelima lakukan inisialisasi nilai *random* pada perulangan x mulai dari 0 sampai dengan 1, yang kemudian disimpan dalam matriks b[x].

4.1.2 Diagram Alir Normalisasi Data





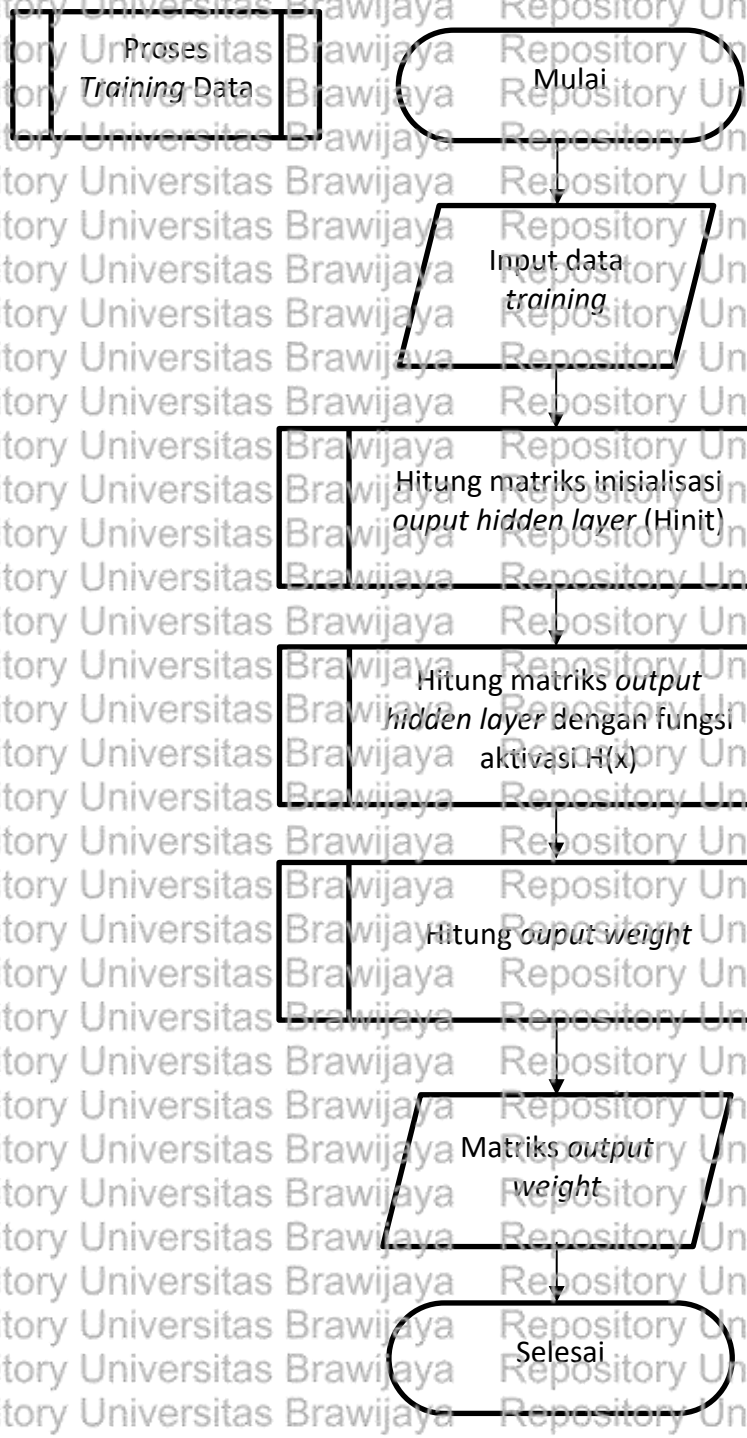


Gambar 4.3 Diagram Alir Normalisasi Data

Pada diagram aiir di atas dijelaskan mengenai aiur dalam proses normaiisasi data menggunakan *Min-Max Normalization*. Adapun langkah-langkah yang dilakukan berdasarkan Gambar 4.3 sebagai berikut.

1. Langkah pertama dengan memasukkan *dataset* yang terdiri dari parameter X sebagai fitur inputan yaitu X1 hingga X8 dan parameter T sebagai target yang akan diprediksi
2. Langkah kedua dengan melakukan perulangan i=0 sampai dengan jumlahKolom-1, kemudian dilakukan inisialisasi *array* untuk penentuan nilai maksimal dan minimal di awal menjadi Max = X[i][0] dan Min = X[i][0].
3. Langkah ketiga dengan melakukan j = 0 sampai dengan jumlahData-1, kemudian dilakukan seleksi kondisi ketika nilai Max lebih kecil dari nilai X[i][j] maka nilai X[i][j] menjadi nilai Maksimal yang baru. Jika nilai Max tidak lebih kecil dari nilai X[i][j] maka, masuk pada seleksi kondisi yang kedua, yaitu seleksi kondisi ketika nilai Min lebih besar dari nilai X[i][j] maka nilai X[i][j] menjadi nilai Minimal yang baru.
4. Setelah mendapatkan nilai Maksimal dan Minimal pada tiap kolom, langkah keempat yang dilakukan dengan membuat perulangan i = 0 sampai dengan jumlahKolom-1 dan perulangan j = 0 sampai dengan jumlahData-1, kemudian lakukan perhitungan normalisasi data dengan menggunakan rumus seperti pada Persamaan 2.1

4.1.3 Diagram Alir Proses *Training* Data



Gambar 4.4 Diagram Alir Proses *Training* Data

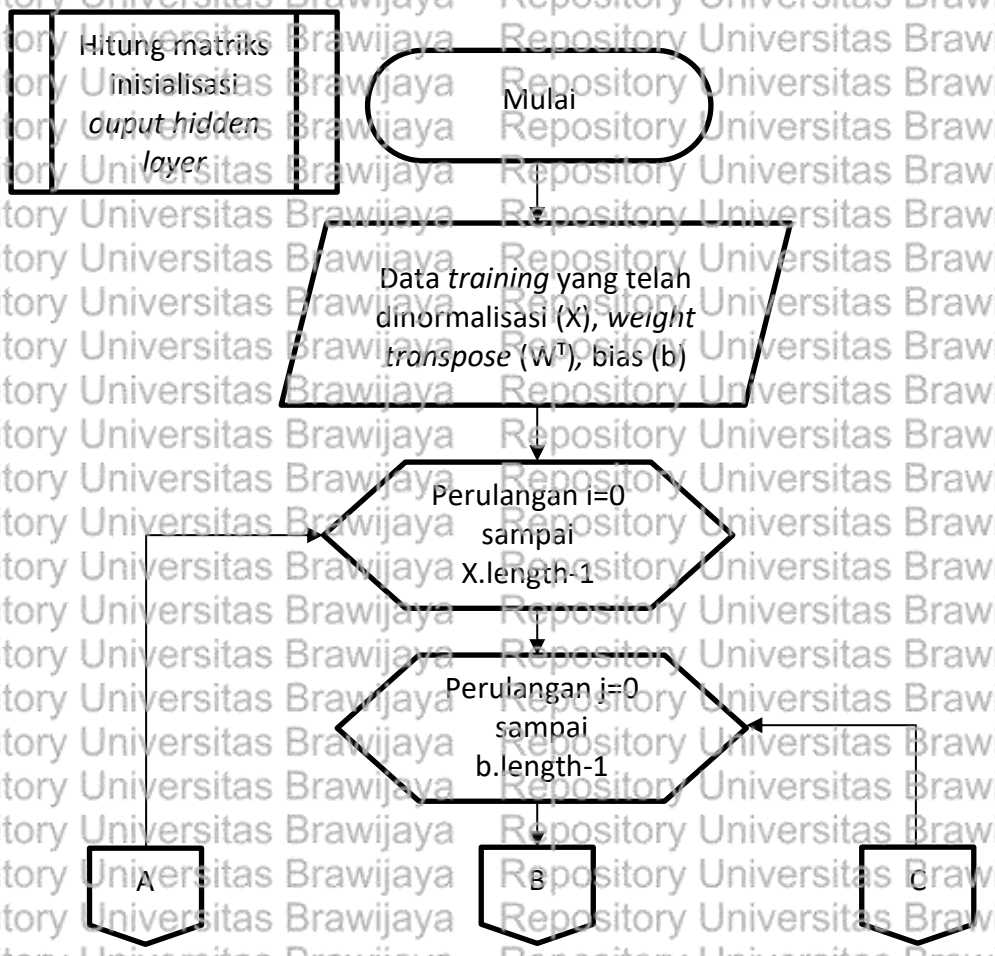
Pada diagram alir di atas dijelaskan mengenai alur dalam proses *training* data. Adapun langkah-langkah dalam proses *training* data berdasarkan Gambar 4.4 sebagai berikut:

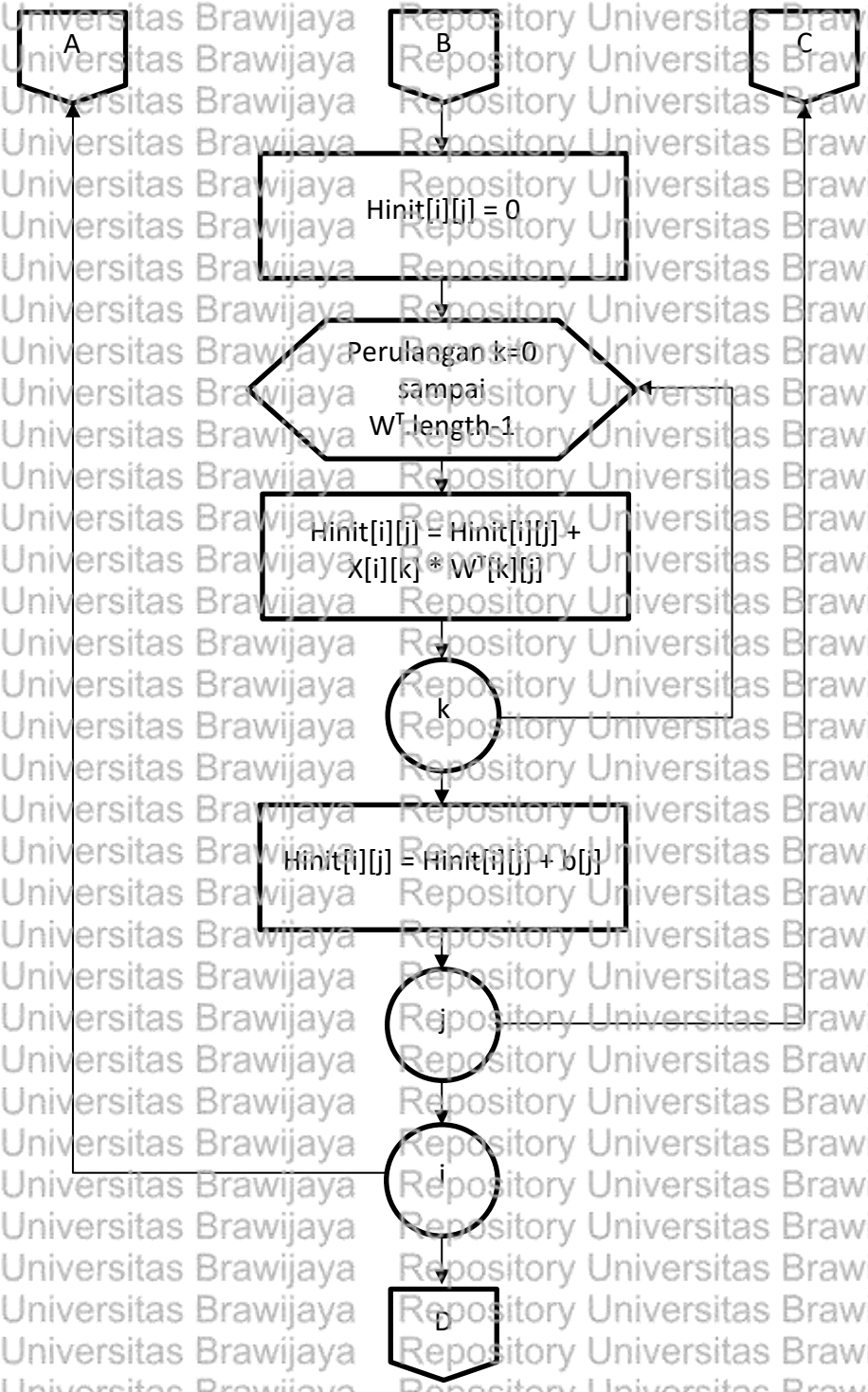
1. Langkah pertama yaitu memasukkan data latih yang telah dinormalisasi sebanyak 60% dari total *dataset*, yang meliputi inputan fitur produksi air

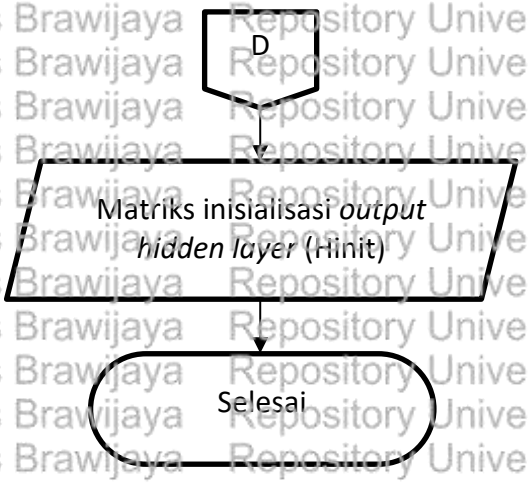
sebagai X1, kekeruhan air baku sebagai X2, pH air baku sebagai X3, alkalinitas air baku sebagai X4, kekeruhan air bersih sebagai X5, pH air bersih sebagai X6, klor bebas air bersih sebagai X7, dan alkalinitas air bersih sebagai X8.

2. Langkah kedua dengan melakukan perhitungan matriks inisialiasi *output hidden layer* (H_{init}) yang nantinya digunakan dalam perhitungan matriks *output hidden layer*.
3. Langkah ketiga dengan melakukan perhitungan matriks *ouput hidden layer* dengan menggunakan fungsi aktivasi H(x), kemudian keluaran matriks akan digunakan dalam proses perhitungan *output weight*.
4. Langkah keempat dengan melakukan perhitungan *output weight* yang dalam prosesnya dilakukan perhitungan Matriks H^+ (*Moore Penrose Pseudo-Inverse*) yang hasil akhirnya berupa matriks *output weight* yang akan digunakan dalam proses *testing* data.

4.1.3.1 Diagram Alir Perhitungan Matriks *Output Hidden Layer*




A
B
C
Hinit[i][j] = 0
Perulangan k=0
sampai
W^T length-1
Hinit[i][j] = Hinit[i][j] +
X[i][k] * W^T[k][j]
k
Hinit[i][j] = Hinit[i][j] + b[j]
j
i
D

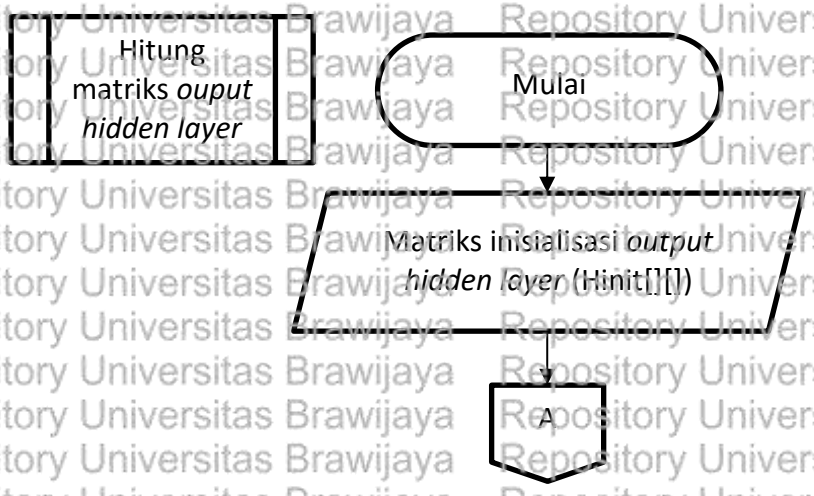


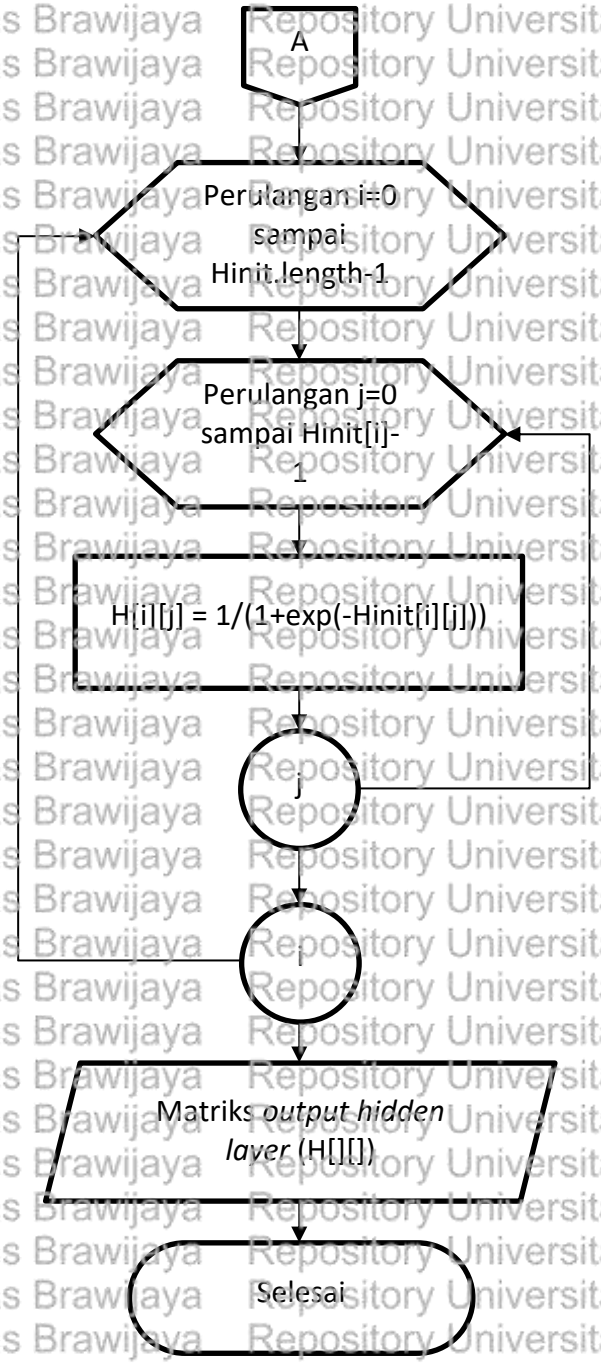
Gambar 4.5 Diagram Alir Perhitungan Matriks Inisialisasi *Output Hidden Layer*

Pada diagram alir di atas dijelaskan mengenai alur dalam perhitungan matriks inisialisasi *output hidden layer*. Adapun langkah-langkah dalam proses perhitungan pada Gambar 4.5 sebagai berikut:

a. Langkah pertama dilakukan penginputan data *training* yang telah dinormalisasi, *weight transpose*, dan bias.
b. Langkah kedua dilakukan perulangan i = 0 sampai dengan X.length-1, yaitu perulangan pada panjang data (X). Setelah itu dilakukan perulangan j = 0 sampai dengan b.length-1, yaitu perulangan pada nilai bias. Nilai matriks *output hidden layer* merupakan penjumlahan dari matriks $X.W^T$ dengan bias, sehingga untuk inisialisasi awal dinotasikan dengan $H_{init}[i][j] = 0$.
c. Untuk menghasilkan nilai matriks $H_{init}[i][j]$ dilakukan perulangan k = 0 sampai dengan W^T.length-1, yang hasil matriksnya ($W^T[k][j]$)
d. dikalikan dengan matriks data yang telah dinormalisasi (X[i][k]).
e. Setelah mendapatkan hasil dari poin c, selanjutnya dilakukan penjumlahan dengan matriks b[j]. Hingga akhirnya menghasilkan matriks inisialisasi *output hidden layer* (H_{init}).

4.1.3.2 Diagram Alir Perhitungan Matriks *Output Hidden Layer* dengan Fungsi Aktivasi H(x)





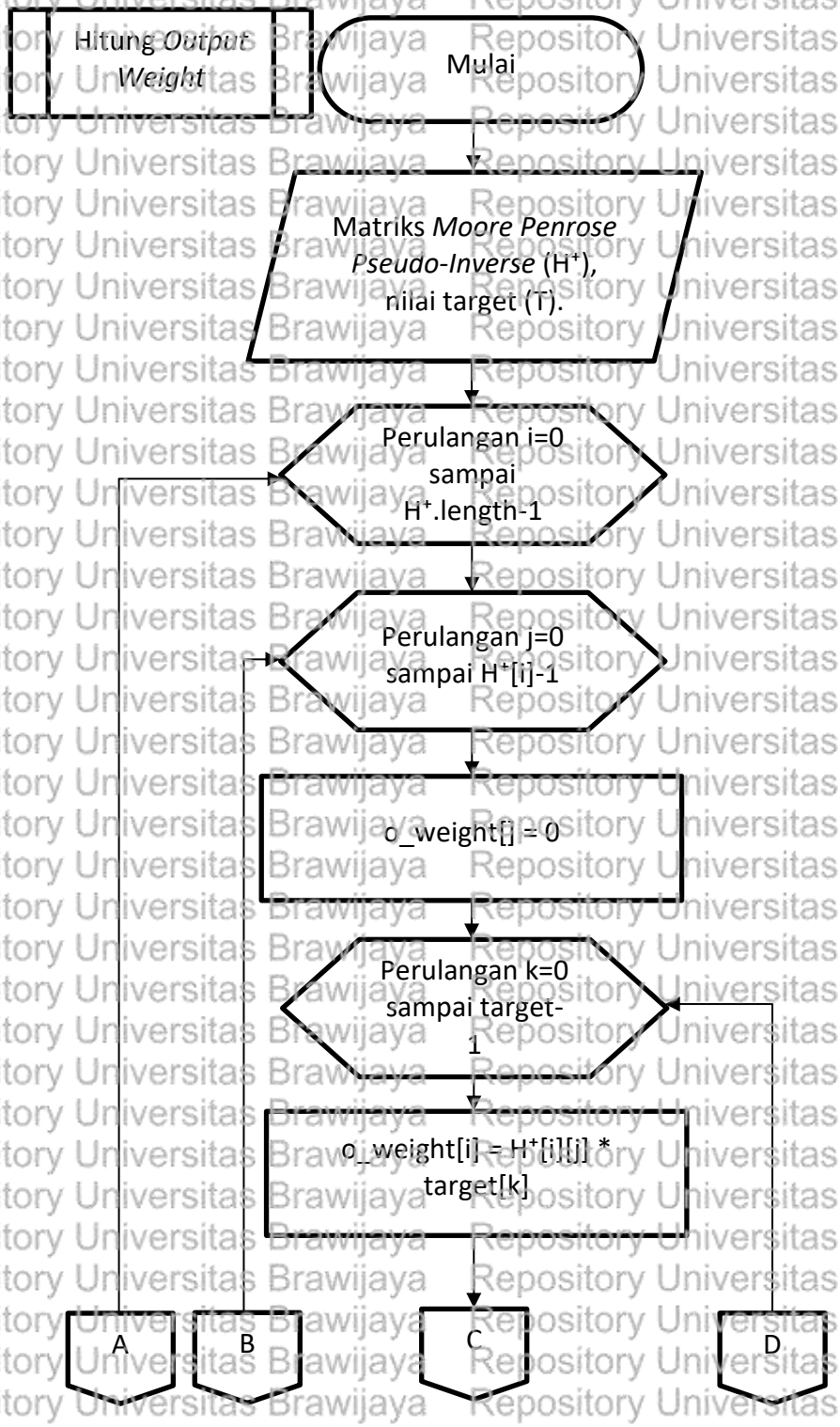
Gambar 4.6 Diagram Alir Perhitungan Matriks *Output Hidden Layer*

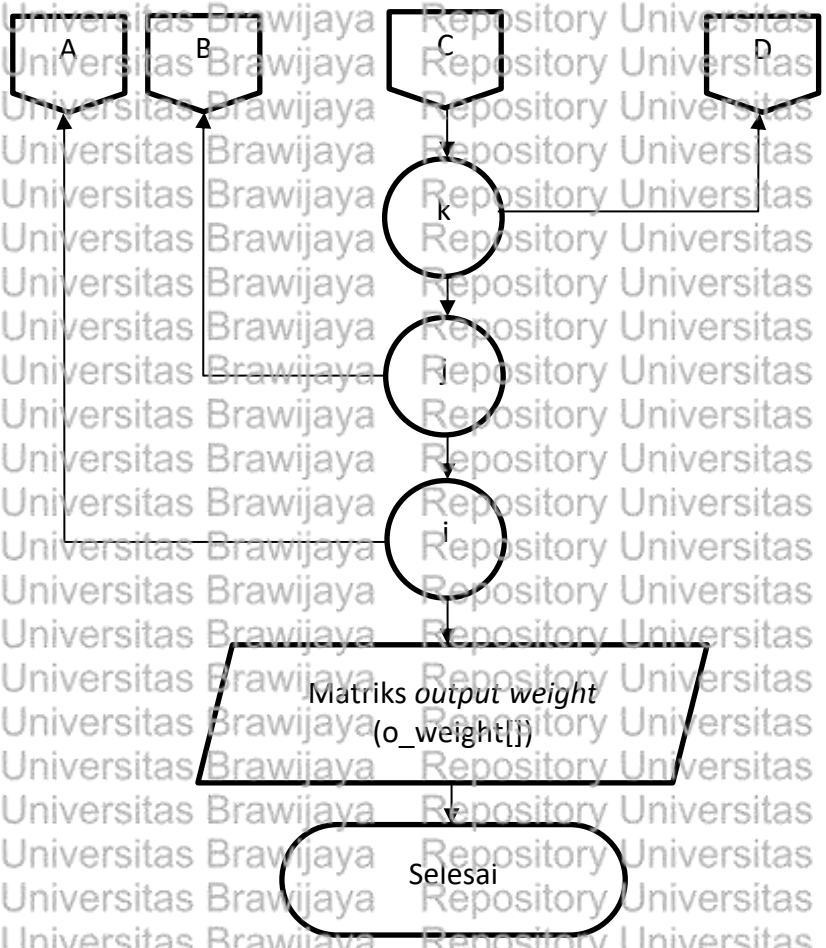
Pada diagram alir di atas dijelaskan mengenai alur dalam perhitungan matriks *output hidden layer* dengan fungsi aktivasi H(x). Adapun langkah-langkah dalam proses perhitungan pada Gambar 4.6 sebagai berikut:

1. Langkah pertama dilakukan penginputan matriks inisialiasi *output hidden layer* (H_{init}[][]), kemudian dilakukan perulangan i = 0 sampai dengan H H_{init}.length-1 untuk panjang matriks H_{init}, kemudian dilakukan perulangan j = 0 sampai dengan H_{init}[i].length-1 untuk isi kolom dari matriks H_{init}.

2. Setelah kondisi perulangan terpenuhi, dilakukan perhitungan *output hidden layer* dengan rumus H = 1/(1+exp(-H_{init})). Matriks yang terbentuk merupakan fungsi aktivasi dari matriks *output hidden layer*.

4.1.3.3 Diagram Alir Perhitungan *Output Weight*



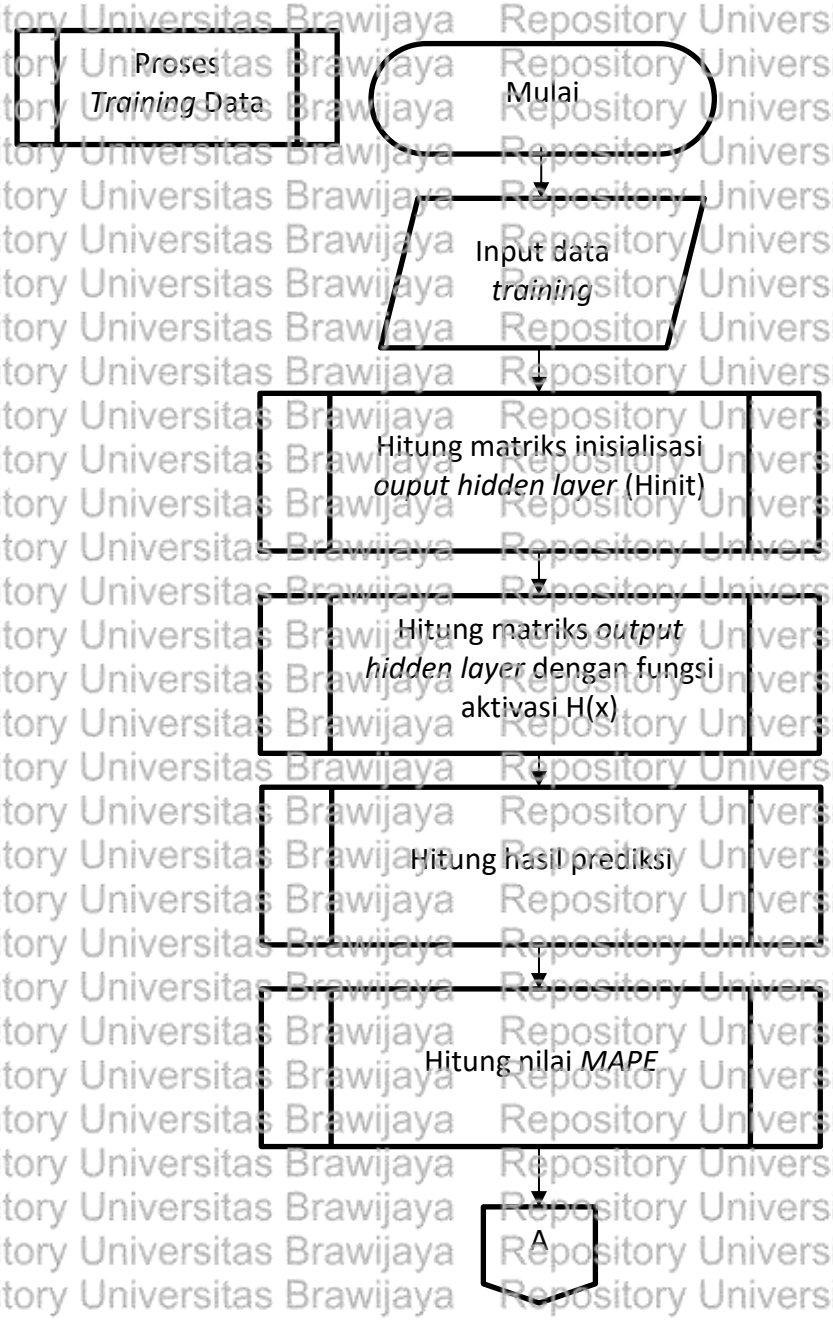


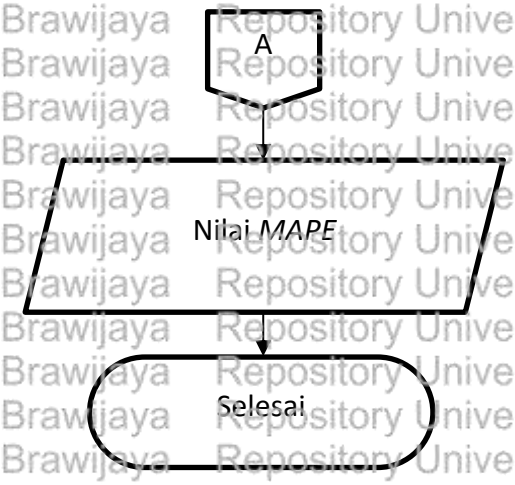
Gambar 4.7 Diagram Alir Perhitungan Matriks *Output Weight*

Pada diagram alir di atas dijelaskan mengenai alur dalam perhitungan hasil nilai *output weight* dengan menggunakan inputan nilai matriks H^+ dan nilai target (T). Adapun langkah-langkah dalam proses perhitungan pada Gambar 4.7 sebagai berikut:

1. Langkah pertama dilakukan penginputan matriks *Moore Penrose Pseudo-Inverse* (H^+) dan nilai target (T).
2. Langkah kedua dilakukan perulangan i = 0 sampai dengan nilai H^+-1 untuk mendapat nilai baris pada matriks, selanjutnya lakukan perulangan j = 0 sampai dengan nilai H^+ [i]-1 untuk mendapat nilai kolom pada matriks.
3. Langkah ketiga inisialisasi matriks untuk *output weight*, yaitu o_weight[] = 0. Selanjutnya dilakukan perulangan k = 0 sampai dengan nilai target-1.
4. Setelah kondisi perulangan terpenuhi, dilakukan perhitungan dengan mengalikan matriks (H][j]) dengan nilai T (target[k]). Kemudian, hasil dari perhitungan tersebut membentuk matriks *output weight* (o_weight[]).

4.1.4 Diagram Alir Proses *Training* Data



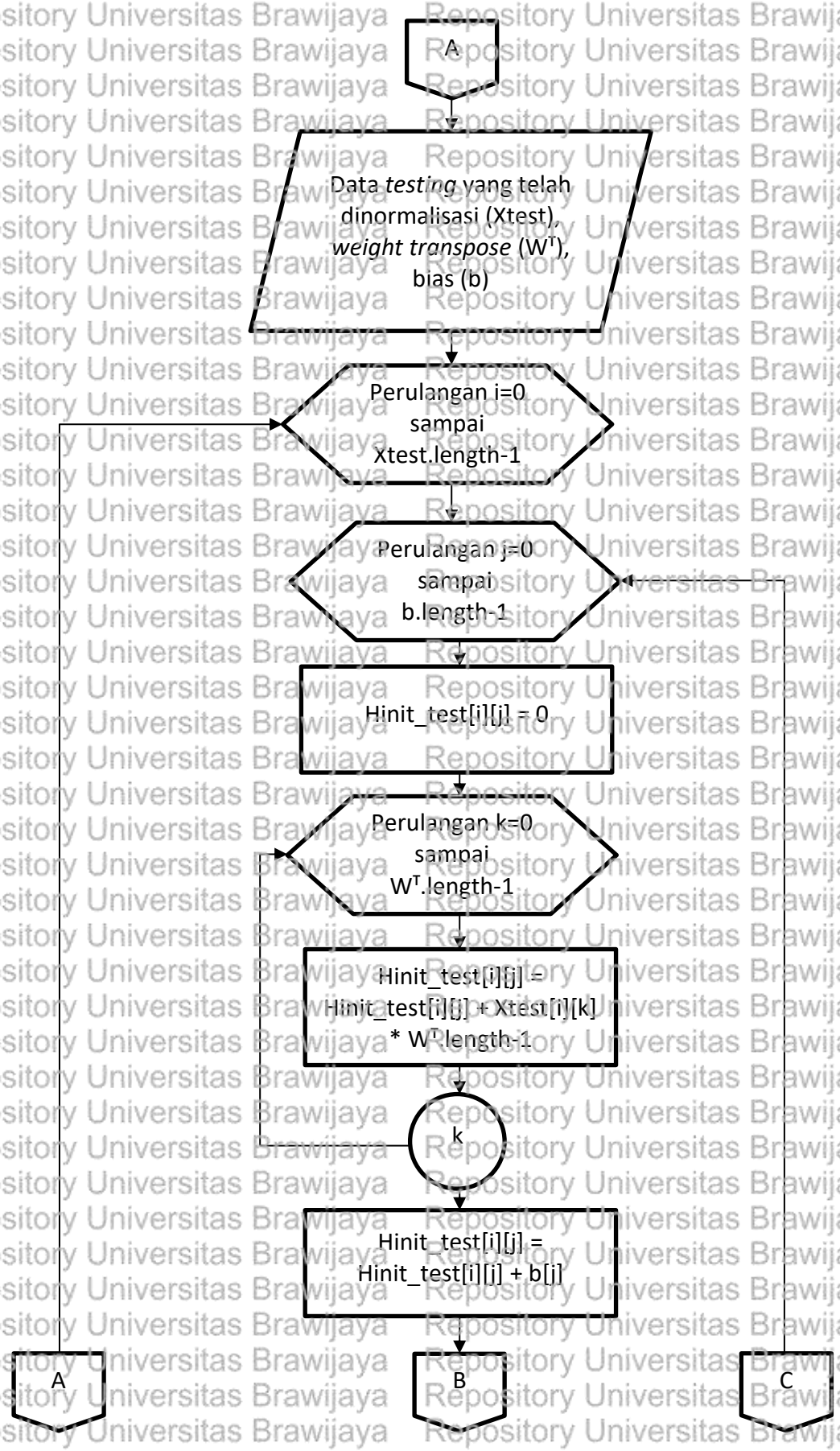


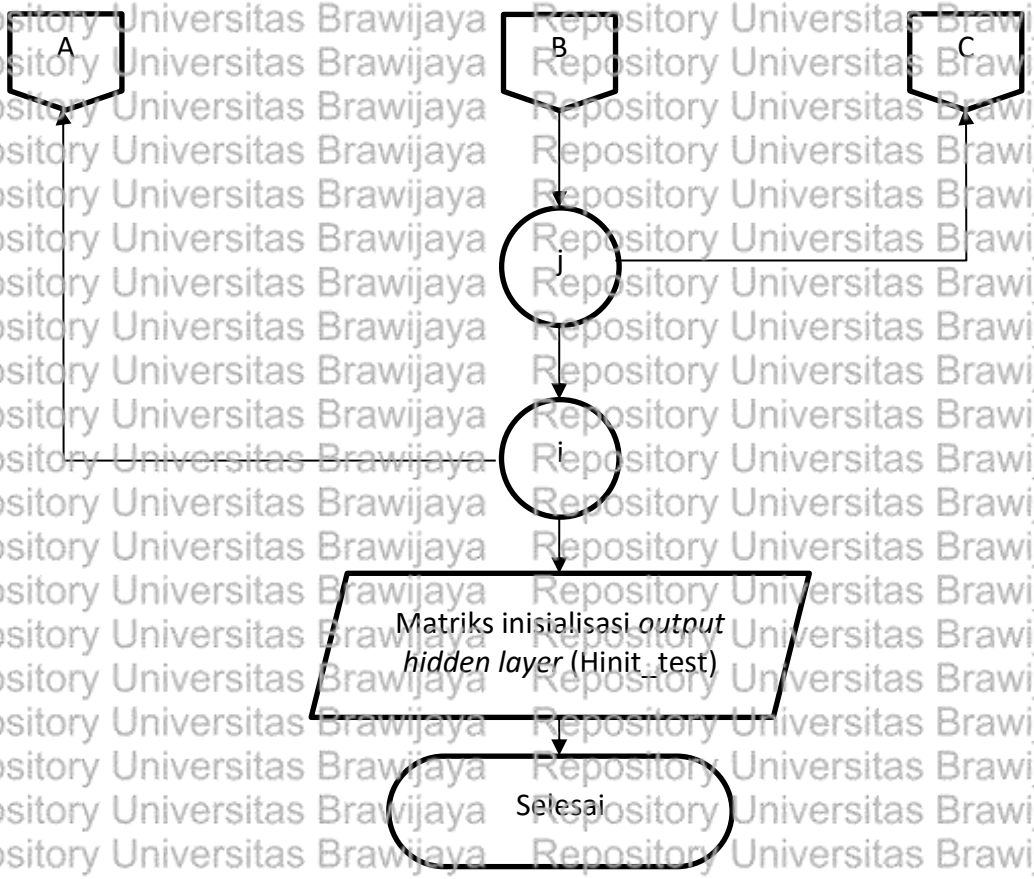
Gambar 4.8 Diagram Alir Proses *Testing* Data

Pada diagram alir di atas dijelaskan mengenai alur dalam proses *testing* data. Adapun langkah-langkah dalam proses *training* data berdasarkan Gambar 4.8 sebagai berikut:

1. Langkah pertama yaitu memasukkan sisa data yang belum digunakan pada proses *training* dan memasukkan matriks *output weight* yang sebelumnya telah didapatkan dari proses *training*. Data yang digunakan yaitu 40% data telah dinormalisasi yang selanjutnya disebut sebagai data uji, yang meliputi inputan fitur produksi air sebagai X1, kekeruhan air baku sebagai X2, pH air baku sebagai X3, alkalinitas air baku sebagai X4, kekeruhan air bersih sebagai X5, pH air bersih sebagai X6, klor bebas air bersih sebagai X7, dan alkalinitas air bersih sebagai X8.
2. Langkah kedua dengan melakukan perhitungan matriks inisialiasi *output hidden layer* (H_{init}) yang nantinya digunakan dalam perhitungan matriks *output hidden layer*.
3. Langkah ketiga dengan melakukan perhitungan matriks *ouput hidden layer* dengan menggunakan fungsi aktivasi H(x), kemudian keluaran matriks digunakan dalam proses perhitungan hasil prediksi.
4. Langkah keempat dengan melakukan perhitungan nilai *error*, namun sebelumnya dilakukan denormalisasi terhadap data hasil prediksi. Nilai *error* yang dihasilkan digunakan untuk melakukan perhitungan nilai *MAPE*.

4.1.4.1 Diagram Alir Perhitungan Matriks Inisialisasi *Output Hidden Layer*


A
Data *testing* yang telah dinormalisasi (Xtest), *weight transpose* (W^T), bias (b)
Perulangan i=0 sampai Xtest.length-1
Perulangan j=0 sampai b.length-1
Hinit_test[i][j] = 0
Perulangan k=0 sampai W^T.length-1
Hinit_test[i][j] = Hinit_test[i][j] + Xtest[i][k] * W^T.length-1
k
Hinit_test[i][j] = Hinit_test[i][j] + b[j]
A
B
C

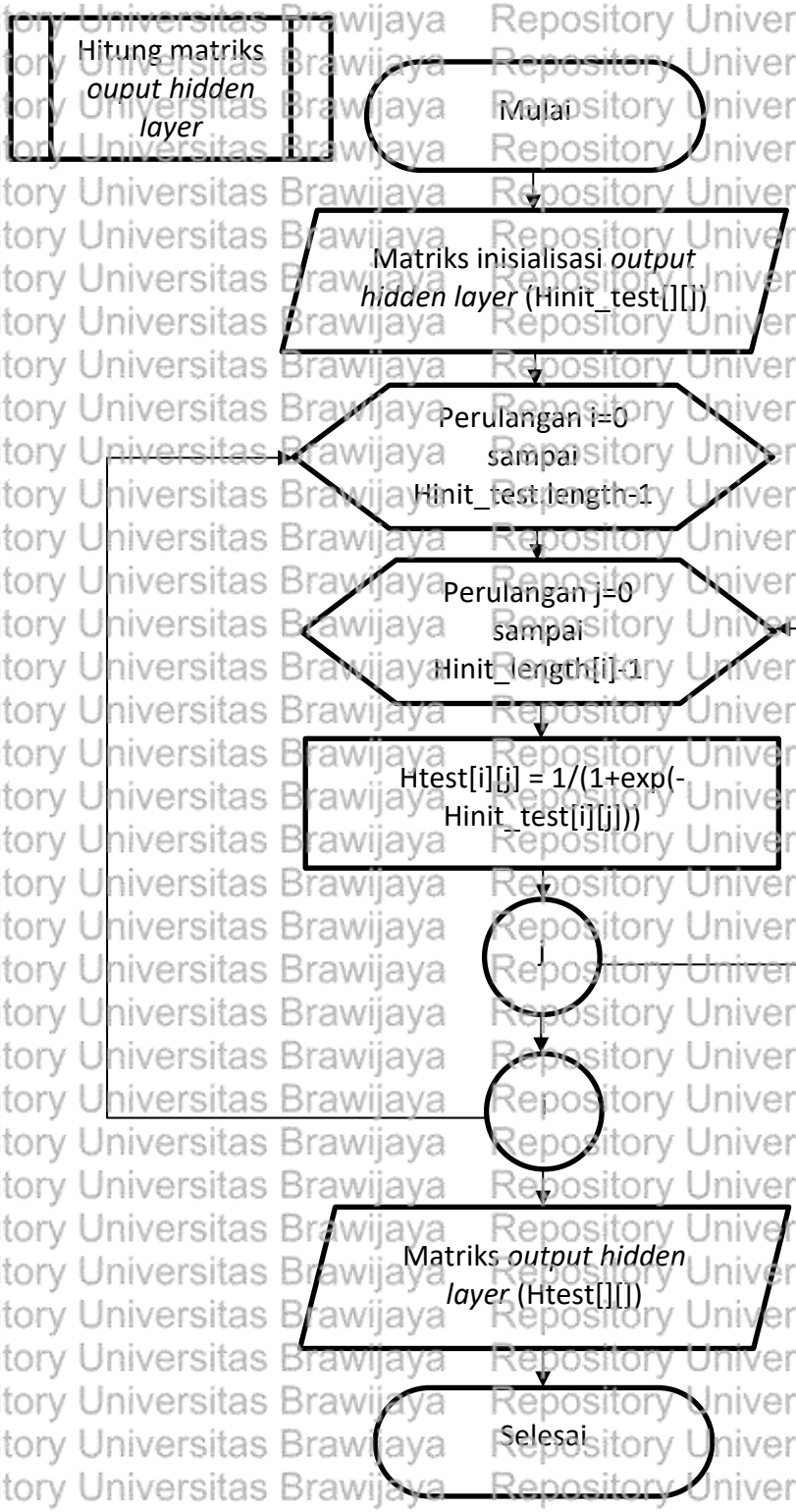


Gambar 4.9 Diagram Alir Pernitungan Matriks Inisialisasi *Output Hidden Layer* pada Proses *Testing* Data

Pada diagram alir di atas dijelaskan mengenai alur dalam perhitungan matriks inisialisasi *output hidden layer* pada proses *testing* data. Adapun langkah-langkah dalam proses perhitungan pada Gambar 4.9 sebagai berikut:

a. Langkah pertama dilakukan penginputan data *testing* yang telah dinormalisasi, *weight transpose*, dan bias.
b. Langkah kedua dilakukan perulangan i = 0 sampai dengan Xtest.length-1, yaitu perulangan pada panjang data (Xtest). Setelah itu dilakukan perulangan j = 0 sampai dengan b.length-1 yaitu perulangan pada nilai bias. Nilai matriks *output hidden layer* merupakan penjumlahan dari matriks Xtest.W^T dengan bias, sehingga untuk inisialisasi awal dinotasikan dengan H_{init}_test[i][j] = 0.
c. Untuk menghasilkan nilai matriks H_{init}_test[i][j] dilakukan perulangan k = 0 sampai dengan W^T.length-1, yang hasil matriksnya (W^T[k][j]) dikalikan dengan matriks data yang telah dinormalisasi (Xtest[i][k]).
d. Setelah mendapatkan hasil dari poin c, selanjutnya dilakukan penjumlahan dengan matriks b[j]. Hingga akhirnya menghasilkan matriks inisialisasi *output hidden layer* (H_{init}_test).

4.1.4.2 Diagram Alir Perhitungan Matriks *Output Hidden Layer* dengan Fungsi Aktivasi H(x) pada Proses *Testing*

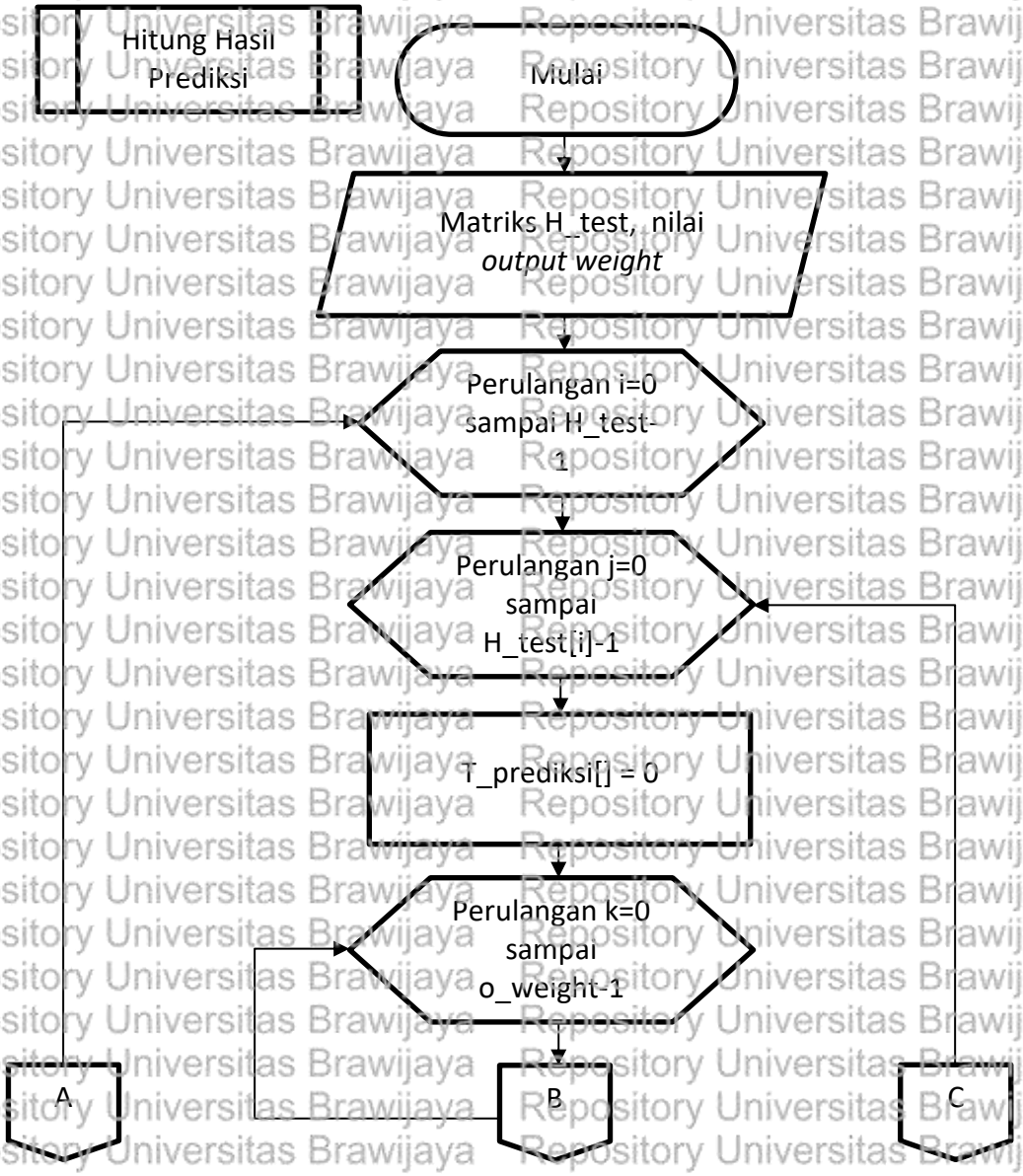


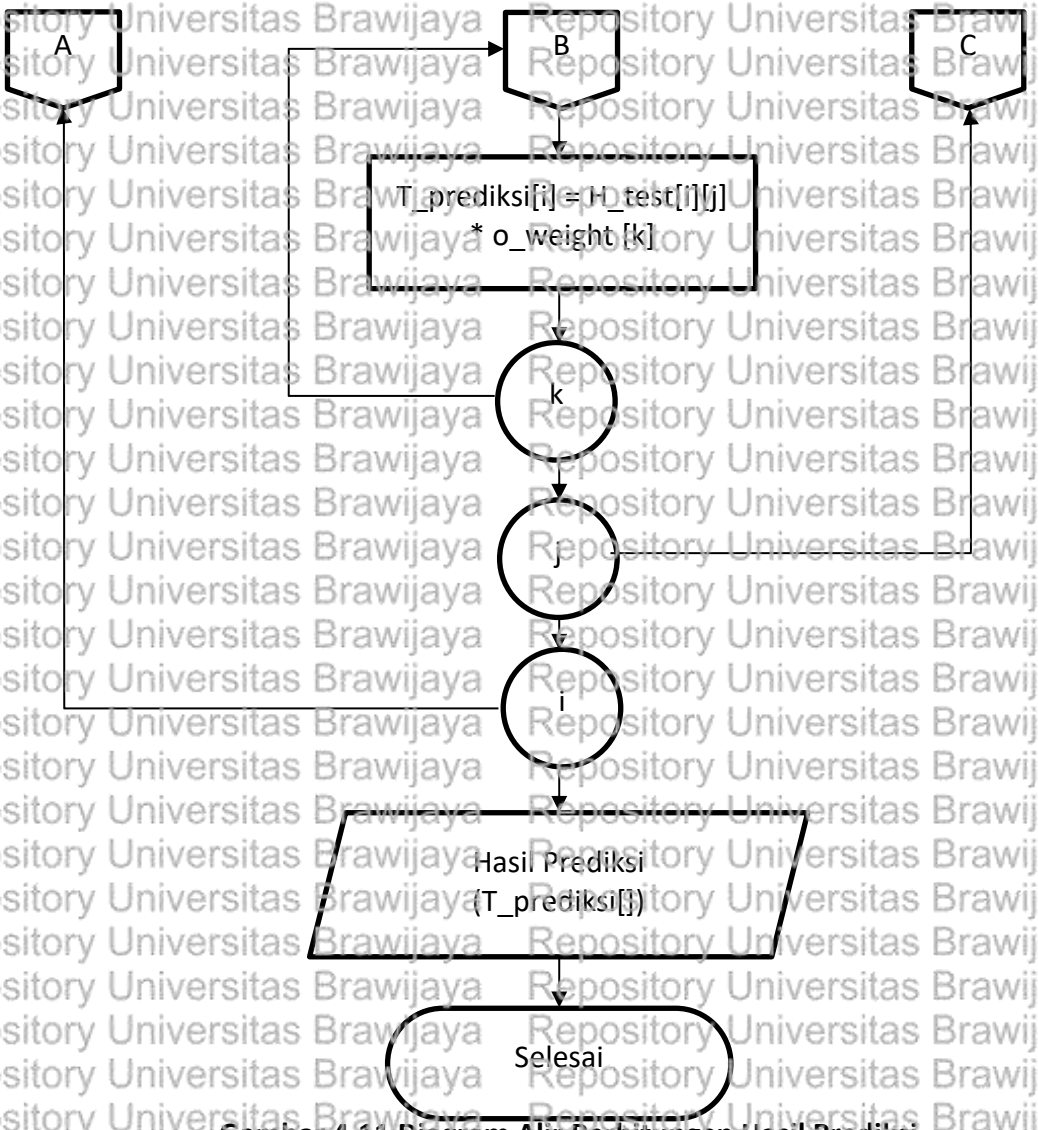
Gambar 4.10 Diagram Alir Perhitungan Matriks *Output Hidden Layer* pada Proses *Testing* Data

Pada diagram alir di atas dijelaskan mengenai alur dalam perhitungan matriks *output hidden layer* pada proses *testing* data dengan menggunakan fungsi aktivasi H(x). Adapun langkah-langkah dalam proses perhitungan pada Gambar 4.10 sebagai berikut:

1. Langkah pertama dilakukan penginputan matriks inisialiasi *output hidden layer* dari proses *testing* (H_{init}_testing[][]), kemudian dilakukan perulangan i = 0 sampai dengan H_{init}_test.length-1 untuk panjang matriks Hinit_test, kemudian dilakukan perulangan j = 0 sampai dengan H_{init}_test[i].length-1 untuk isi kolom dari matriks Hinit_test.
2. Setelah kondisi perulangan terpenuhi, dilakukan perhitungan *output hidden layer* dengan rumus H_test = 1/(1+exp(-H_{init_test})). Matriks yang terbentuk merupakan fungsi aktivasi dari matriks *output nidden layer*.

4.1.4.3 Diagram Alir Perhitungan Hasil Prediksi



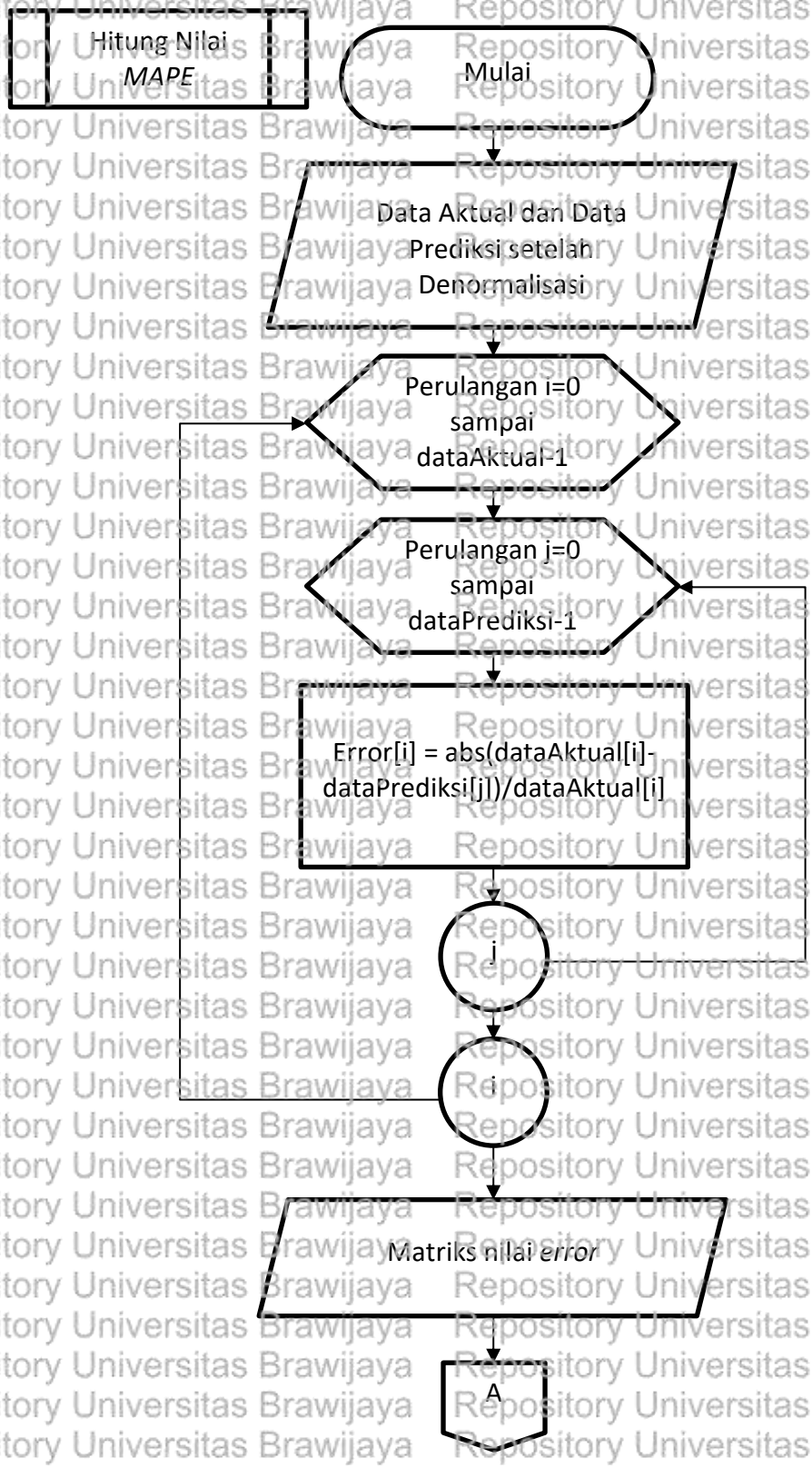


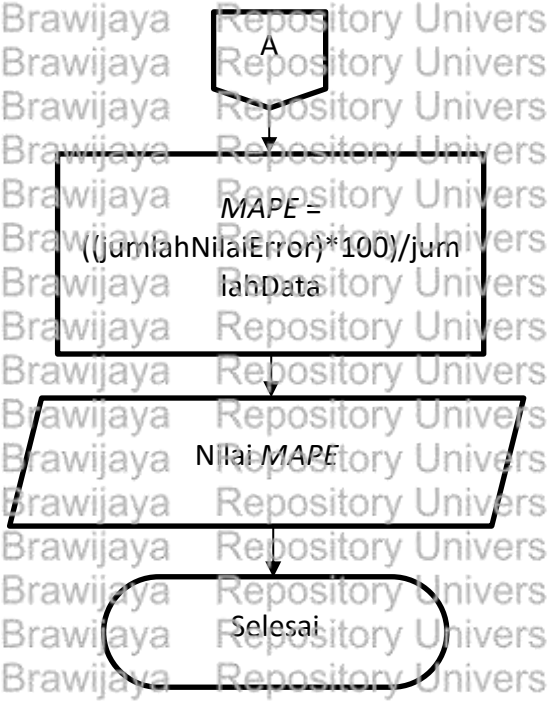
Gambar 4.11 Diagram Alir Perhitungan Hasil Prediksi

Pada diagram alir di atas dijelaskan mengenai alur dalam perhitungan hasil prediksi dengan menggunakan nilai matriks *output hidden layer* pada proses *testing* dan nilai *output weight*. Adapun langkah-langkah dalam proses perhitungan pada Gambar 4.11 sebagai berikut:

1. Langkah pertama dilakukan penginputan matriks *output hidden layer* (H_test[][]) dan *output weight*
2. Langkah kedua dilakukan perulangan i = 0 sampai dengan nilai H_test-1 untuk mendapat nilai baris pada matriks, selanjutnya lakukan perulangan j = 0 sampai dengan nilai H_test[i]-1 untuk mendapat nilai kolom pada matriks.
3. Langkah ketiga inisialisasi matriks untuk hasil prediksi, yaitu T_prediksi[] = 0. Selanjutnya dilakukan perulangan k = 0 sampai dengan nilai o_weight-1.
4. Setelah kondisi perulangan terpenuhi, dilakukan perhitungan dengan mengalikan matriks *output hidden layer* (H_test[i][j]) dengan nilai *output weight* (o_weight[k]). Kemudian, hasil dari perhitungan tersebut membentuk matriks dari hasil prediksi (T_prediksi[])

4.1.4.4 Diagram Alir Perhitungan Nilai *MAPE*





Gambar 4.12 Diagram Alir Perhitungan Nilai *MAPE*

Pada diagram alir di atas dijelaskan mengenai alur dalam perhitungan nilai *MAPE*. Adapun langkah-langkah dalam proses perhitungan pada Gambar 4.12 sebagai berikut:

a. Langkah pertama dilakukan penginputan nilai data aktual dan data hasil prediksi yang telah didenormalisasi. Setelah itu dilakukan perulangan i = 0 sampai dengan dataAktual-1 untuk mendapat nilai keseluruhan dari data aktual dan dilakukan perulangan j = 0 sampai dengan dataPrediksi-1 untuk mendapat nilai keseluruhan dari data hasil prediksi.
b. Langkah kedua dilakukan perulangan untuk perhitungan nilai *error*, yaitu dengan rumus nilai absolut dari pengurangan data aktual dengan data prediksi, kemudian hasilnya dibagi dengan nilai data aktual. Ketika kondisi perulangan telah terpenuhi, akan menghasilkan matriks yang merupakan nilai *error*.
c. Langkah ketiga dilakukan perhitungan nilai *MAPE* dengan rumus jumlah dari keseluruhan nilai *error* dikalikan dengan angka 100, kemudian hasilnya dibagi dengan jumlah data *testing*. Hasil yang diperoleh merupakan nilai dari *MAPE*, yang merepresentasikan tingkat kesalahan dari hasil prediksi dengan metode yang digunakan.

4.2 Perhitungan Manual Sistem Prediksi menggunakan Data *Multivariate* dengan Metode ELM

Perhitungan manual merupakan proses yang dijalankan pada tahap setelah pembuatan diagram alir. Proses perhitungan manual dilakukan berdasarkan langkah-langkah yang terdapat dalam setiap diagram alir dengan langsung

merepresentasikan perhitungan menggunakan *sample* angka yang terdapat pada *dataset*. Pada proses perhitungan manual dibutuhkan data *sample* yang selanjutnya akan dimanualisasi untuk proses *training* dan *testing* data.

Tabel 4.1 *Dataset Multivariate*

X1	X2	X3	X4	X5	X6	X7	X8	T
831.217	85,38	6,86	52,1	3,5	6,67	0,21	47,16	701.357
783.260	97,71	6,93	49,02	18,37	6,47	0,12	44,01	660.451
806.496	94,61	7,18	49,44	3,7	7,01	0,39	45,02	681.519
870.287	80,9	7	47,63	3,94	6,81	0,34	44,19	733.636
842.618	33,26	6,73	54,83	3,69	6,64	0,28	49,63	710.767
820.895	9,65	6,59	51,38	2,62	6,67	0,1	46,17	690.780
854.348	16,73	6,59	43,39	2,67	6,59	0,08	37,67	722.415
789.168	6,71	6,55	46,1	2,33	6,57	0,15	43,17	663.192
891.202	7,53	6,65	44,92	2,06	6,63	0,28	40,28	749.229
838.792	14,85	6,67	41,52	2,82	6,7	0,24	39,1	704.996
878.643	70,19	7,71	53,54	4,27	6,8	0,35	49,16	736.393
963.059	30,37	6,8	35,35	4,33	6,7	0,34	31,8	807.422

Tabel 4.1 merupakan beberapa contoh data *multivariate* yang selanjutnya digunakan dalam proses *training* dan *testing* data. Data yang terdapat pada Tabel 4.1 akan dibagi untuk data *training* yaitu data ke-1 sampai data ke-10, kemudian untuk data *testing* yaitu menggunakan data ke-11 sampai data ke-12. Adapun keterangan fitur-fitur yang terdapat pada Tabel 4.1 dijelaskan di bawah ini:

1. X1 merupakan atribut dari debit produksi air dengan satuan ukur m^3.
2. X2 merupakan atribut dari kekeruhan air baku dengan satuan ukur NTU (*Nephelometric Turbidity Unit*).
3. X3 merupakan atribut dari pH air baku.
4. X4 merupakan atribut dari alkalinitas air baku dengan satuan ukur ppm (*part per million*).
5. X5 merupakan atribut dari kekeruhan air bersih dengan satuan ukur NTU (*Nephelometric Turbidity Unit*).
6. X6 merupakan atribut dari pH air bersih.
7. X7 merupakan atribut dari klor bebas air bersih dengan satuan ukur mg/L.
8. X8 merupakan atribut dari alkalinitas air bersih dengan satuan ukur ppm (*part per million*).
9. T merupakan target dari debit air yang layak didistribusi dengan satuan ukur m^3.

4.2.1 Inisialisasi Bobot Input dan Bias

Dalam penerapan metode ELM dilakukan tahapan inisialisasi bobot input dan bias secara acak, yaitu dengan rentang nilai 0 sampai dengan 1. Sebelum melakukan inisialisasi bobot input dan bias, dilakukan penentuan jumlah *input neuron* (k) dan jumlah *hidden neuron* (j), yang selanjutnya membentuk matriks bobot input, yaitu [Wj × k]. Pada kasus ini menggunakan 8 *input neuron* sesuai dengan atribut-atribut yang terdapat pada *dataset* dan menggunakan 2 *hidden neuron*. Dalam penentuan jumlah *hidden neuron* akan digunakan dalam inisialisasi bias yang akan membentuk matriks bias, yaitu b[k].

Inisialisasi nilai bobot input yang diperoleh berdasarkan jumlah *input neuron* ditunjukkan pada Tabel 4.2.

Tabel 4.2 Inisialisasi Nilai Bobot Input

Matriks Bobot input (W)	1	2	3	4	5	6	7	8
1	-0,256	0,145	-0,662	0,319	0,411	-0,395	0,556	-0,565
2	0,337	0,498	-0,592	0,612	-0,388	0,215	0,715	-0,897

Inisialisasi nilai bias yang diperoleh berdasarkan jumlah *hidden neuron* ditunjukkan pada Tabel 4.3.

Tabel 4.3 Inisialisasi Nilai Bias

Matriks Bias (b)	1
1	0,135
2	0,286

4.2.2 Proses Normalisasi Data

Pada proses normalisasi data dengan *multivariate* dilakukan dengan pencarian nilai minimal dan maksimal terlebih dahulu dari setiap fitur yang ada pada *dataset*. Proses normalisasi pada tahap ini dilakukan berdasarkan kolom yang sama atau berdasarkan fitur yang sama. Hasil dari pencarian nilai minimal dan maksimal pada tiap fitur yang ada ditunjukkan pada Tabel 4.4.

Tabel 4.4 Nilai Minimal dan Maksimal Data

	X1	X2	X3	X4	X5	X6	X7	X8	T
Min	783.260	2,98	6,55	10,64	1,60	6,47	0,08	8,83	660.451

Max	1.302.233	97,71	7,71	82,75	18,37	7,01	1,10	81,72	866.024

Setelah mendapatkan nilai minimal dan maksimal pada tiap fitur, lakukan tahap normalisasi data dengan menggunakan perhitungan *Min-Max Normalization* seperti pada Persamaan 2.2. Berikut merupakan *sample* dalam pernitungan proses normalisasi data:

$$x_{1,1} = \frac{831.217 - 783.260}{1.302.233 - 783.260} = 0{,}092$$

$$x_{1,2} = \frac{85{,}38 - 2{,}98}{97{,}71 - 2{,}98} = 0{,}870$$

Hasil dari proses normalisasi data beberapa *sample* data ditunjukkan pada Tabel 4.5.

Tabel 4.5 Hasil Normalisasi Data

Data Ke-	X1	X2	X3	X4	X5	X6	X7	X8	T
1	0,092	0,870	0,267	0,575	0,113	0,370	0,127	0,526	0,199
2	0	1	0,328	0,532	1	0	0,039	0,483	0

4.2.3 Proses *Training* Data

Pada proses *training* data dalam implementasi metode ELM dilakukan beberapa langkah, yaitu perhitungan matriks inisialisasi *output hidden layer*, perhitungan matriks *output hidden layer* dengan fungsi aktivasi dengan *sigmoid biner*, perhitungan matriks *Moore Penrose Pseudo-Inverse*, dan perhitungan *output weight*. Hasil dari proses *training* ini berupa matriks *output weight*, yang selanjutnya digunakan pada proses *testing* data untuk menentukan hasil prediksi. Dalam mengimplementasikan metode ELM dibutuhkan data yang dijadikan sebagai data latih dan data uji. Dalam contoh perhitungan manual ini, digunakan 10 data sebagai data latih dan 2 data sebagai data uji. Representasi dari data latih yang digunakan pada proses manualisasi ini ditunjukkan pada Tabel 4.6.

Tabel 4.6 Data Latih

Data Latih	X1	X2	X3	X4	X5	X6	X7	X8
1	0,092	0,870	0,267	0,575	0,113	0,370	0,127	0,526
2	0	1	0,328	0,532	1	0	0,039	0,483
3	0,045	0,967	0,543	0,538	0,125	1	0,304	0,497

4	0,168	0,823	0,388	0,513	0,140	0,630	0,255	0,485
5	0,114	0,320	0,155	0,613	0,125	0,315	0,196	0,560
6	0,073	0,070	0,034	0,565	0,061	0,370	0,020	0,512
7	0,137	0,145	0,034	0,454	0,064	0,222	0	0,396
8	0,011	0,039	0	0,492	0,044	0,185	0,069	0,471
9	0,208	0,048	0,086	0,475	0,027	0,296	0,196	0,431
10	0,107	0,125	0,103	0,428	0,073	0,426	0,157	0,415

4.2.3.1 Perhitungan Manual Matriks Inisialisasi *Output Hidden Layer*

Pada manualiasasi untuk menentukan matriks inisialisasi *output hidden layer* (H_{init}), dibutuhkan masukan berupa data *training* yang telah dinormalisasi, data bobot input yang telah di*transpose*, dan bias. Dalam menentukan matriks H_{init} ini dilakukan perkalian terhadap data *training* yang telah dinormalisasi dengan data bobot input yang telah di*transpose*, kemudian hasilnya dijumlahkan dengan nilai bias yang sebelumnya telah diinisialisasi. Hasil dari proses ini akan menghasilkan matriks H_{init} yang selanjutnya digunakan dalam perhitungan matriks *output hidden layer*. Hasil dari proses *transpose* matriks bobot input ditunjukkan pada Tabel 4.7.

Tabel 4.7 Matriks Bobot input *Transpose*

Matriks W^T	1	2
1	-0,256	0,337
2	0,145	0,498
3	-0,662	-0,592
4	0,319	0,612
5	0,411	-0,388
6	-0,395	0,215
7	0,556	0,715
8	-0,565	-0,897

Berikut merupakan perhitungan manual matriks H_{init} terhadap beberapa *sample* pada data latih yang menggunakan Persamaan 2.3.

$$H_{init(1,1)} = [(0{,}092 \times (-0{,}256)) + (0{,}870 \times 0{,}145) + (0{,}267 \times (-0{,}662)) + (0{,}575 \times 0{,}319) + (0{,}113 \times 0{,}411) + (0{,}370 \times (-0{,}395)) + (0{,}127 \times 0{,}556) + (0{,}526 \times (-0{,}565)] + 0{,}135 = -0{,}081$$

$$H_{init(1,2)} = [(0{,}092 \times (0{,}337)) + (0{,}870 \times 0{,}498) + (0{,}267 \times (-0{,}592)) + (0{,}575 \times 0{,}612) + (0{,}113 \times (-0{,}388)) + (0{,}370 \times 0{,}215) + (0{,}127 \times 0{,}715) + (0{,}526 \times (-0{,}897)] + 0{,}286 = 0{,}599$$

Untuk hasil perhitungan manual matriks H_{init} terhadap seluruh data latih ditunjukkan pada Tabel 4.8.

Tabel 4.8 Matriks Bobot Inisialisasi *Output Hidden Layer*

Matriks H_{init}	1	2
1	-0,081	0,599
2	0,393	0,123
3	-0,378	0,729
4	-0,205	0,665
5	-0,035	0,424
6	-0,115	0,281
7	-0,041	0,330
8	0,011	0,260
9	-0,056	0,426
10	-0,091	0,388

4.2.3.2 Perhitungan Manual Matriks *Output Hidden Layer*

Pada manualiasasi untuk menentukan matriks *output hidden layer* H(x), dibutunkan masukan dari matriks inisialisasi *output hidden layer*. Dalam menentukan matriks *output hidden layer* dilakukan menggunakan fungsi aktivasi *sigmoid biner*. Hasil dari proses ini menghasilkan matriks H(x) yang selanjutnya digunakan pada perhitungan matriks *Moore Penrose Pseudo-Inverse*.

Berikut merupakan perhitungan manual matriks H(x) terhadap beberapa *sample* pada data latih yang menggunakan Persamaan 2.3:

$$H(x)_{(1,1)} = \frac{1}{1 + exp^{-(-0{,}031)}} = 0{,}479$$

$$H(x)_{(1,2)} = \frac{1}{1 + exp^{-0{,}599}} = 0{,}645$$

Untuk hasil perhitungan manual matriks H(x) terhadap seluruh data latih ditunjukkan pada Tabel 4.9.

Tabel 4.9 Matriks *Output Hidden Layer*

Matriks H(x)	1	2
1	0,479	0,645
2	0,597	0,530
3	0,406	0,674
4	0,448	0,660
5	0,491	0,604
6	0,471	0,569
7	0,489	0,581
8	0,502	0,564
9	0,485	0,605
10	0,477	0,595

4.2.3.3 Perhitungan Manual Matriks *Moore Penrose Pseudo-Inverse*

Pada manualiasasi untuk menentukan matriks *Moore Penrose Pseudo-Inverse* (H^+) dibutuhkan masukan berupa matriks *output hidden layer* H(x). Dalam menentukan matriks H^+ ini didapatkan dari perhitungan *inverse* matriks H(x) *transpose* yang dikalikan dengan matriks H(x), kemudian hasilnya dikalikan dengan matriks H(x) yang telah di*transpose*. Hasil dari proses ini menghasilkan matriks H^+ yang selanjutnya digunakan dalam perhitungan *output weight*.

Langkah pertama yang dilakukan ialah mengalikan matriks H(x) transpose dengan matriks H(x). Berikut merupakan perkalian matriks terhadap beberapa *sample* pada data latih yang menggunakan Persamaan 2.5:

$$H(x)^{\mathrm{T}}.H(x)_{(1,1)} = (0{,}479 \times 0{,}479) + (0{,}597 \times 0{,}597)$$
$$+(0{,}406 \times 0{,}406) + (0{,}448 \times 0{,}448) + (0{,}491 \times 0{,}491)$$
$$+(0{,}4712 \times 0{,}4712) + (0{,}489 \times 0{,}489) + (0{,}502 \times 0{,}509)$$
$$+(0{,}485 \times 0{,}485) + (0{,}477 \times 0{,}477) = 2{,}373$$

$$H(x)^{\mathrm{T}}.H(x)_{(1,2)} = (0{,}479 \times 0{,}645) + (0{,}597 \times 0{,}530)$$
$$+(0{,}406 \times 0{,}674) + (0{,}448 \times 0{,}660) + (0{,}491 \times 0{,}604)$$
$$+(0{,}472 \times 0{,}569) + (0{,}489 \times 0{,}581) + (0{,}502 \times 0{,}564)$$
$$+(0{,}485 \times 0{,}605) + (0{,}477 \times 0{,}595) = 2{,}910$$

Untuk hasil perhitungan dari seluruh *sample* data pada data latih ditunjukkan pada Tabel 4.10.

Tabel 4.10 Hasil Perkalian Matriks H(x) *Transpose* dengan Matriks H(x)

Matriks $(H(x)^T.H(x))$	1	2
1	2,373	2,910
2	2,910	3,659

Setelah mendapatkan hasil perkalian matriks ($H(x)^+.H(x)$), langkah selanjutnya yaitu mencari matriks *inverse* dari matriks tersebut.

Tabel 4.11 Hasil Matriks *Inverse* dari Perkalian Matriks H(x) *Transpose* dengan Matris H(x)

Matriks $(H(x)^T.H(x))^{-1}$	1	2
1	17,064	-13,571
2	-13,571	11,066

Setelah mendapatkan matriks *inverse*, langkah selanjutnya adalah mengalikan matriks *inverse* tersebut dengan matriks H(x) *transpose*. Berikut merupakan perhitungan manual pada perkalian kedua matriks untuk beberapa *sample* data pada data latih:

$$H^{+}_{(1,1)} = (17{,}064 \times 0{,}479) + ((-13{,}571) \times 0{,}645)) = -0{,}579$$

$$H^{+}_{(1,2)} = (17{,}064 \times 0{,}597) + ((-13{,}571) \times 0{,}530)) = 2{,}985$$

Hasil perhitungan manual matriks *Moore Penrose Pseudo*-Inverse (H^+) terhadap seluruh data latih dapat dilihat pada Tabel 4.12.

Tabel 4.12 Hasil Matriks *Moore Penrose Pseudo*-Inverse

Matriks $H^+(x)$	1	2	3	4	5	...	10
1	-0,576	2,985	-2,221	-1,303	0,177	...	0,054
2	0,635	-2,229	1,950	1,216	0,024	...	0,119

4.2.3.4 Perhitungan Manual Matriks *Output Weight*

Pada manualiasasi untuk menentukan matriks *output weight* (β) dibutuhkan masukan berupa matriks H^+ dan nilai target pada data latih. Dalam menentukan matriks β ini didapatkan dari perhitungan matriks H^+ yang dikalikan dengan nilai target pada data latih. Hasil dari proses ini menghasilkan matriks β yang selanjutnya digunakan dalam perhitungan pada proses *testing* data. Representasi dari nilai target pada data latih ditunjukkan pada Tabel 4.13.

Tabel 4.13 Nilai Target pada Data Latih

Nilai Target (T)	1
1	0,199
2	0
3	0,102
4	0,356
5	0,245
6	0,148
7	0,301
8	0,013
9	0,432
10	0,217

Langkah selanjutnya adalah mengalikan matriks H^+ dengan nilai target (T), berikut perhitungan manualnya dengan menggunakan Persamaan 2.5:

$$\beta_{(1,1)} = (-0{,}576 \times 0{,}199) + (2{,}985 \times 0) + ((-2{,}221) \times 0{,}102) + ((-1{,}303) \times 0{,}356) + (0{,}177 \times 0{,}245) + (0{,}305 \times 0{,}148) + (0{,}457 \times 0{,}301) + (0{,}919 \times 0{,}103) + (0{,}078 \times 0{,}432) + (0{,}054 \times 0{,}217) = -0{,}521$$

$$\beta_{(2,1)} = (-0{,}635 \times 0{,}199) + ((-2{,}229) \times 0) + (1{,}950 \times 0{,}102) + (1{,}216 \times 0{,}356) + (0{,}024 \times 0{,}245) + ((-0{,}087) \times 0{,}148) + ((-0{,}205) \times 0{,}301) + ((-0{,}576) \times 0{,}103) + (0{,}102 \times 0{,}432) + (0{,}119 \times 0{,}217) = 0{,}753$$

Hasil dari perhitungan nilai *output weight* ditunjukkan pada Tabel 4.14.

Tabel 4.14 Hasil *Output Weight*

Matriks β	1
1	-0,521
2	0,753

4.2.4 Proses *Testing* Data

Pada proses *testing* data dalam implementasi metode ELM dilakukan beberapa langkah, yaitu perhitungan matriks inisialisasi *output hidden layer*, perhitungan matriks *output hidden layer* dengan fungsi aktivasi *sigmoid biner*,

perhitungan hasil prediksi, dan perhitungan nilai *MAPE*. Hasil dari proses *testing* ini adalah nilai *MAPE* (%) yang merepresentasikan akurasi dari implementasi metode yang digunakan terhadap peramalan kasus pada penelitian ini. Pada proses *testing* ini menggunakan 2 *sample* data uji yang dapat dilihat pada Tabel 4.15.

Tabel 4.15 Data Uji

Data Uji	X1	X2	X3	X4	X5	X6	X7	X8
1	0,184	0,709	1	0,595	0,159	0,611	0,265	0,553
2	0,346	0,289	0,216	0,343	0,163	0,426	0,255	0,315

4.2.4.1 Perhitungan Manual Matriks Inisialisasi *Output Hidden Layer*

Pada manualiasasi untuk menentukan matriks inisialisasi *output hidden layer* (H_{init_test}), dibutuhkan masukan berupa data *testing* yang telah dinormalisasi, data bobot input yang telah di*transpose*, dan bias. Dalam menentukan matriks H_{init_test} ini dilakukan perkalian terhadap data *testing* yang telah dinormalisasi dengan data bobot input yang telah di*transpose*, kemudian hasilnya dijumlahkan dengan nilai bias yang sebelumnya telah diinisialisasi. Hasil dari proses ini menghasilkan matriks H_{init_test} yang selanjutnya digunakan dalam perhitungan matriks *output hidden layer*.

Berikut merupakan perhitungan manual matriks H_{init} terhadap beberapa *sample* pada data latih dengan menggunakan Persamaan 2.6:

$$H_{init\_test(1,1)} = [(0{,}184 \times (-0{,}256)) + (0{,}709 \times 0{,}145) + (1 \times (-0{,}662)) + (0{,}595 \times 0{,}319) + (0{,}159 \times 0{,}411) + (0{,}611 \times (-0{,}395) + (0{,}265 \times 0{,}556) + (0{,}553 \times (-0{,}565))] + 0{,}135 = -0{,}622$$

$$H_{init\_test(1,2)} = [(0{,}184 \times (0{,}337)) + (0{,}709 \times 0{,}498) + (1 \times (-0{,}592)) + (0{,}595 \times 0{,}612) + (0{,}159 \times (-0{,}388)) + (0{,}611 \times 0{,}215 + (0{,}265 \times 0{,}715) + (0{,}553 \times (-0{,}897))] + 0{,}286 = 0{,}237$$

Hasil perhitungan manual matriks H_{init} terhadap seluruh data latih dapat dilihat pada Tabel 4.16.

Tabel 4.16 Matriks Bobot Inisialisasi *Output Hidden Layer*

Matriks H_{init_test}	1	2
1	-0,622	0,237
2	-0,082	0,557

4.2.4.2 Perhitungan Manual Matriks *Output Hidden Layer*

Pada manualiasasi untuk menentukan matriks *output hidden layer* $H(x)_{test}$, dibutuhkan masukan dari matriks inisialisasi *output hidden layer* (H_{init_test}). Dalam menentukan matriks *output hidden layer* dilakukan menggunakan fungsi aktivasi *sigmoid biner*. Hasil dari proses ini menghasilkan matriks $H(x)_{test}$ yang selanjutnya digunakan pada perhitungan hasil prediksi.

Berikut merupakan perhitungan manual matriks H(x) terhadap beberapa *sample* pada data latih dengan menggunakan Persamaan 2.6:

$$H(x)\_test_{(1,1)} = \frac{1}{1 + exp^{-(-0,622)}} = 0,349$$

$$H(x)\_test_{(1,2)} = \frac{1}{1 + exp^{-0,237}} = 0,558$$

Hasil perhitungan manual matriks $H(x)_{test}$ terhadap seluruh data latih dapat dilihat pada Tabel 4.17.

Tabel 4.17 Matriks *Output Hidden Layer*

Matriks $H(x)_{test}$	1	2
1	0,349	0,558
2	0,479	0,635

4.2.4.3 Perhitungan Manual Hasil Prediksi

Pada manualiasasi untuk menentukan hasil prediksi dibutuhkan masukan berupa matriks *output hidden layer* dari proses *testing* ($H(x)_{test}$) dan matriks *output weight* (β) yang diperoleh dari proses *training*. Dalam menentukan hasil prediksi dilakukan dengan mengalikan matriks $H(x)_{test}$ dengan matriks β. Hasil dari proses ini berupa matriks hasil prediksi namun belum didenormalisasi.

Berikut merupakan pernitungan manual hasil prediksi terhadap *sample* pada data uji dengan menggunakan Persamaan 2.7:

$$\check{T}_{(1,1)} = \big(0,349 \times (-0,521)\big) + (0,558 \times 0,753) = 0,238$$

Hasil perhitungan manual matriks $\check{T}(x)$ terhadap seluruh data latih dapat dilihat pada Tabel 4.18.

Tabel 4.18 Matriks Hasil Prediksi

Matriks $\check{T}(x)$	1
1	0,238
2	0,228

4.2.4.4 Perhitungan Manual Denormalisasi

Perhitungan denormalisasi dilakukan pada data hasil prediksi yang berfungsi untuk mengembalikan nilai asli yang sebelumnya telah mengalami proses normalisasi. Dalam proses denormalisasi ini diperlukan masukan berupa data hasil prediksi, nilai minimal, dan nilai maksimal pada kolom target dari keseluruhan *dataset*. Proses perhitungannya adalah dengan mengalikan nilai prediksi yang belum didenormalisasi dengan selisih dari nilai minimal dan nilai maksimal pada kolom target, setelah itu hasilnya dijumlahkan dengan nilai minimal. Nilai minimal dan maksimal pada fitur T ditunjukkan pada Tabel 4.19.

Tabel 4.19 Nilai Terkecil dan Terbesar pada Kolom Target

	T
Min	660.451
Max	866.024

Berikut merupakan perhitungan manual denormalisasi terhadap *sample* hasil prediksi pada data uji dengan menggunakan Persamaan 2.8:

$$d_{(1,1)} = 0{,}238\ (866.024 - 660.451) + 660.451 = \ 709.536$$

Hasil proses denormalisasi terhadap seluruh data hasil prediksi ditunjukkan pada Tabel 4.20.

Tabel 4.20 Hasil Proses Denormalisasi Data Prediksi

Matriks $d(x)$	1
1	709.536
2	707.494

4.2.4.5 Perhitungan Manual Nilai *MAPE*

Hasil perhitungan nilai *MAPE* (%) merupakan representasi dari akurasi terhadap implementasi metode ELM yang digunakan terhadap peramalan kasus pada penelitian ini. Sebelum menentukan nilai *MAPE*, dilakukan perhitungan nilai *error* terlebih dahulu, yaitu selisih antara data aktual dan data prediksi. Hasil dari selisih data aktual dan data prediksi kemudian dibagi dengan data aktual, selanjutnya hasil yang diperoleh disebut dengan nilai *error*. Representasi data aktual yang selanjutnya akan dibandingkan dengan data prediksi dapat dilihat pada Tabel 4.21.

Tabel 4.21 Data Aktual

Data Aktual (T)	1
1	736.393
2	807.422

Berikut merupakan contoh perhitungan manual dari nilai selisih terhadap data aktual dan data prediksi dengan menggunakan Persamaan 2.9:

$$selisih_{(1,1)} = |736.393 - 709.536| \div 736.393 = 0{,}036$$

Hasil perhitungan nilai selisih terhadap seluruh data aktual dan data prediksi dapat dilihat pada Tabel 4.22

Tabel 4.22 Hasil Nilai Selisih

Nilai *Error*	1
1	0,036
2	0,123

Dalam menentukan nilai *MAPE*, digunakan total dari nilai *error* yang sebelumnya telah diperoleh, kemudian total tersebut dibagi dengan banyaknya data latih, yaitu 2 data kemudian hasilnya dikalikan dengan angka 100. Berikut merupakan perhitungan nilai *MAPE* menggunakan Persamaan 2.9:

$$\boldsymbol{MAPE}(\%) = [(0{,}036 + 0{,}123) \div 2] \times 100 = \mathbf{8{,}01}\ \%$$

4.3 Pengujian Algoritme Menggunakan Data *Multivariate* dengan Metode *ELM*

Tahapan pengujian algoritme dalam suatu penelitian berfungsi untuk mengetahui hasil kinerja sistem berdasarkan metode yang diimplementasikan terhadap objek penelitian. Pada pengujian algoritme ini, penulis akan melakukan pengujian dengan dua skenario yang berbeda-beda. Berikut merupakan penjelasan mengenai skenario pengujian yang akan dilakukan:

1. Skenario pengujian jumlah *neuron* pada *hidden layer*.
2. Skenario pengujian dalam perbandingan jumlah data *training* dan data *testing* yang digunakan.

4.3.1 Pengujian Jumlah *Neuron* pada *Hidden Layer*

Dalam melakukan pengujian terhadap jumlah *neuron* berfungsi untuk mencari nilai *MAPE* yang terbaik, yaitu nilai *MAPE* yang terkecil. Pada pengujian jumlah *neuron* ini menggunakan jumlah *neuron*, yaitu 1 sampai 10 *neuron*. Pengujian terhadap jumlah *neuron* ini dilakukan sebanyak 10 kali dengan berupa seluruh *dataset* yang telah dibagi menjadi data latih dan data uji.

Tabel 4.23 Rancangan Skenario Pengujian Jumlah *Neuron* pada *Hidden Layer*

Percobaan ke-	Nilai *MAPE* pada Jumlah *Hidden Neuron*									
	1	2	3	4	5	6	7	8	9	10
1										
2										
3										
4										
5										
6										
7										
8										
9										
10										
Rata-rata nilai *MAPE*										

4.3.2 Pengujian Perbandingan Jumlah Data *Training* dan Data *Testing*

Dalam melakukan pengujian terhadap perbandingan jumlah data *training* dan data *testing* berfungsi untuk mencari nilai *MAPE* yang terbaik, yaitu nilai *MAPE* terkecil. Pada proses pengujian ini digunakan masukan berupa seluruh *dataset* dan hasil dari pengujian sebelumnya yaitu jumlah *neuron* yang terbaik. Pengujian ini menggunakan perbandingan antara jumlah data *training* dan data *testing* yang berbeda, yaitu 6 : 30, 12 : 24, 18 : 18, 24 : 12, dan 30 : 6.

Tabel 4.24 Rancangan Skenario Pengujian Perbandingan Jumlah Data *Training* dan Data *Testing*

Perco-baan ke-	Nilai *MAPE* pada Perbandingan Jumlah Data *Training* dan Data *Testing*				
	6 : 30	12 : 24	18 : 18	24 : 12	30 : 6
1					
2					
3					
4					
5					
6					
7					
8					
9					
10					
Rata-rata nilai *MAPE*					

4.4 Perhitungan Manual Sistem Prediksi menggunakan Data *Single Variable* dengan Metode ELM

Pada contoh perhitungan manual dengan menggunakan data *single variable*, menggunakan enam jumlah *neuron* pada *input layer*. Proses perhitungan manual dilakukan berdasarkan langkah-langkah yang terdapat dalam setiap diagram alir dengan langsung merepresentasikan perhitungan menggunakan *sample* angka yang terdapat pada *dataset*. Pada proses perhitungan manual dibutuhkan data *sample* yang selanjutnya dimanualisasi untuk proses *training* dan *testing* data. Data yang digunakan pada proses manualisasi ini ditunjukkan pada Tabel 4.25.

Tabel 4.25 *Dataset Single Variable*

Waktu	X
Jan-16	701.357
Feb-16	660.451
Mar-16	681.519

Apr-16	733.636
May-16	710.767
Jun-16	690.780
Jul-16	722.415
Aug-16	663.192
Sep-16	749.229
...	...
Dec-18	826.278

4.4.1 Inisialisasi Bobot Input dan Bias

Dalam penerapan metode ELM dilakukan tahapan inisialisasi bobot input dan bias secara acak, yaitu dengan rentang nilai 0 sampai dengan 1. Sebelum melakukan inisialisasi bobot input dan bias, dilakukan penentuan jumlah *input neuron* (k) dan jumlah *hidden neuron* (j), yang selanjutnya akan membentuk matriks bobot input, yaitu [Wj × k]. Pada kasus ini menggunakan 6 *input neuron* dan menggunakan 2 *hidden neuron*. Dalam penentuan jumlah *hidden neuron* akan digunakan dalam inisialisasi bias yang akan membentuk matriks bias, yaitu b[k].

Inisialisasi nilai bobot input yang diperoleh berdasarkan jumlah *input neuron* ditunjukkan pada Tabel 4.26.

Tabel 4.26 Inisialisasi Nilai Bobot Input

Matriks Bobot Input (W)	**1**	**2**	**3**	**4**	**5**	**6**
1	-0,256	0,145	-0,662	0,319	0,411	-0,395
2	0,337	0,498	-0,592	0,612	-0,388	0,215

Inisialisasi nilai bias yang diperoleh berdasarkan jumlah *hidden neuron* ditunjukkan pada Tabel 4.27.

Tabel 4.27 Inisialisasi Nilai Bias

Matriks Bias (b)	**1**
1	0,135
2	0,286

4.4.2 Proses Normalisasi Data

Pada proses normalisasi dengan data *single variable* dilakukan dengan pencarian nilai minimal dan maksimal terlebih dahulu terhadap seluruh data yang ada pada *dataset*. Hasil dari pencarian nilai minimal dan maksimal pada *dataset* digunakan dalam proses normalisasi dengan menggunakan Persamaan 2.1. yaitu dengan metode *Min-Max Normalization*. Hasil dari pencarian nilai minimal dan maksimal ditunjukkan pada Tabel 4.28.

Tabel 4.28 Nilai Minimal dan Maksimal Data

	X
Min	649.792
Max	889.451

Setelah mendapatkan nilai minimal dan maksimal, dilakukan tahap normalisasi data dengan menggunakan perhitungan *Min-Max Normalization*. Berikut merupakan contoh perhitungan manual proses manualisasi pada *sample* data menggunakan Persamaan 2.2:

$$x_1 = \frac{701.357 - 649.792}{889.451 - 649.792} = 0{,}215$$

$$x_2 = \frac{660.451 - 649.792}{889.451 - 649.792} = 0{,}044$$

Hasil dari proses normalisasi data beberapa *sample* data ditunjukkan pada Tabel 4.29.

Tabel 4.29 Hasil Normalisasi Data *Single Variable*

Waktu	X1
Jan-16	0,215
Feb-16	0,044
Mar-16	0,132
Apr-16	0,349
May-16	0,254
Jun-16	0,171
Jul-16	0,303
Aug-16	0,055
Sep-16	0,414
...	...
Dec-18	0,736

4.4.3 Proses *Training* Data

Pada proses *training* data dalam implementasi metode ELM dilakukan beberapa langkah, yaitu perhitungan matriks inisialisasi *output hidden layer*, perhitungan matriks *output hidden layer* dengan fungsi aktivasi dengan *sigmoid biner*, perhitungan matriks *Moore Penrose Pseudo-Inverse*, dan perhitungan *output weight*. Hasil dari proses *training* ini berupa matriks *output weight*, yang selanjutnya digunakan pada proses *testing* data untuk menentukan hasil prediksi. Dalam mengimplementasikan metode ELM dibutuhkan data yang dijadikan sebagai data latih dan data uji. Dalam contoh perhitungan manual ini, digunakan 10 data sebagai data latih dan 2 data sebagai data uji. Pada contoh manualisasi di bawah ini menggunakan 6 *input neuron* yang berarti akan memprediksi distribusi air berdasarkan 6 data pada bulan sebelumnya. Representasi dari data latih yang digunakan pada proses manualisasi ini ditunjukkan pada Tabel 4.30.

Tabel 4.30 Data Latih

Data Latih	X1	X2	X3	X4	X5	X6	T
Jan-16	0,215	0,044	0,132	0,349	0,254	0,171	0,303
Feb-16	0,044	0,132	0,349	0,254	0,171	0,303	0,055
Mar-16	0,132	0,349	0,254	0,171	0,303	0,055	0,414
Apr-16	0,349	0,254	0,171	0,303	0,055	0,414	0,230
May-16	0,254	0,171	0,303	0,055	0,414	0,230	0,429
Jun-16	0,171	0,303	0,055	0,414	0,230	0,429	0,271
Jul-16	0,303	0,055	0,414	0,230	0,429	0,271	1
Aug-16	0,055	0,414	0,230	0,429	0,271	1	0,374
Sep-16	0,414	0,230	0,429	0,271	1	0,374	0
Okt-16	0,230	0,429	0,271	1	0,374	0	0,954

4.4.3.1 Perhitungan Manual Matriks Inisialisasi *Output Hidden Layer*

Pada manualiasasi untuk menentukan matriks inisialisasi *output hidden layer* (H_{init}), dibutuhkan masukan berupa data *training* yang telah dinormalisasi, data bobot input yang telah di*transpose*, dan bias. Dalam menentukan matriks Hinit ini dilakukan perkalian terhadap data *training* yang telah dinormalisasi dengan data bobot input yang telah di*transpose*, kemudian hasilnya dijumlahkan dengan nilai bias yang sebelumnya telah diinisialisasi. Hasil dari proses ini akan menghasilkan matriks H_{init} yang selanjutnya digunakan dalam perhitungan matriks

output hidden layer. Hasil dari proses *transpose* matriks bobot input ditunjukkan pada Tabel 4.31.

Tabel 4.31 Matriks Bobot Input *Transpose*

Matriks W^T	1	2
1	-0,256	0,337
2	0,145	0,498
3	-0,662	-0,592
4	0,319	0,612
5	0,411	-0,388
6	-0,395	0,215

Berikut merupakan perhitungan manual matriks H_{init} terhadap beberapa *sample* pada data latih menggunakan Persamaan 2.3:

$$H_{init(1,1)} = [(0{,}215 \times (-0{,}256)) + (0{,}0445 \times 0{,}145) + (0{,}132 \times (-0{,}662)) + (0{,}349 \times 0{,}319) + (0{,}254 \times 0{,}411) + (0{,}171 \times (-0{,}395))] + 0{,}135 = 0{,}147$$

$$H_{init(1,2)} = [(0{,}215 \times (0{,}337)) + (0{,}0445 \times 0{,}498) + (0{,}132 \times (-0{,}592)) + (0{,}349 \times 0{,}612) + (0{,}254 \times (-0{,}388)) + (0{,}171 \times 0{,}215)] + 0{,}286 = 0{,}455$$

Hasil perhitungan manual matriks H_{init} terhadap seluruh data latih dapat dilihat pada Tabel 4.32.

Tabel 4.32 Matriks Bobot Inisialisasi *Output Hidden Layer*

Matriks $H_{init}(x)$	1	2
1	0,147	0,455
2	-0,056	0,314
3	0,140	0,353
4	-0,074	0,683
5	-0,008	0,200
6	0,156	0,719
7	-0,066	0,203
8	-0,117	0,748

9	0,128	0,145
10	0,432	0,884

4.4.3.2 Perhitungan Manual Matriks *Output Hidden Layer*

Pada manualiasasi untuk menentukan matriks *output hidden layer* H(x), dibutuhkan masukan dari matriks inisialisasi *output hidden layer*. Dalam menentukan matriks *output hidden layer* dilakukan menggunakan fungsi aktivasi *sigmoid biner*. Hasil dari proses ini menghasilkan matriks H(x) yang selanjutnya digunakan pada perhitungan matriks *Moore Penrose Pseudo-Inverse*.

Berikut merupakan perhitungan manual matriks H(x) terhadap beberapa *sample* pada data latih menggunakan Persamaan 2.3:

$$H(x)_{(1,1)} = \frac{1}{1 + exp^{-0,147}} = 0,536$$

$$H(x)_{(1,2)} = \frac{1}{1 + exp^{-0,455}} = 0,611$$

Hasil perhitungan manual matriks H(x) terhadap seluruh data latih dapat dilihat pada Tabel 4.33.

Tabel 4.33 Matriks *Output Hidden Layer*

Matriks H(x)	1	2
1	0,536	0,611
2	0,485	0,578
3	0,535	0,587
4	0,481	0,664
5	0,497	0,549
6	0,538	0,672
7	0,483	0,550
8	0,470	0,678
9	0,531	0,536
10	0,606	0,707

4.4.3.3 Perhitungan Manual Matriks *Moore Penrose Pseudo-Inverse*

Pada manualiasasi untuk menentukan matriks *Moore Penrose Pseudo-Inverse* (H^+) dibutuhkan masukan berupa matriks *output hidden layer* H(x). Dalam menentukan matriks H^+ ini didapatkan dari perhitungan *inverse* matriks H(x) *transpose* yang dikalikan dengan matriks H(x), kemudian hasilnya dikalikan dengan

matriks H(x) yang telah di*transpose*. Hasil dari proses ini menghasilkan matriks H^+ yang selanjutnya digunakan dalam perhitungan *output weight*.

Langkah pertama yang dilakukan ialah mengalikan matriks H(x) transpose dengan matriks H(x). Berikut merupakan perkalian matriks ternadap beberapa *sample* pada data latih menggunakan Persamaan 2.4.

$$H(x)^{\mathrm{T}}.H(x)_{(1,1)} = (0{,}536 \times 0{,}536) + (0{,}485 \times 0{,}485)$$
$$+(0{,}535 \times 0{,}535) + (0{,}481 \times 0{,}481) + (0{,}497 \times 0{,}497)$$
$$+(0{,}538 \times 0{,}538) + (0{,}483 \times 0{,}483) + (0{,}470 \times 0{,}470)$$
$$+(0{,}531 \times 0{,}531) + (0{,}606 \times 0{,}606) = 2{,}687$$
$$H(x)^{\mathrm{T}}.H(x)_{(1,2)} = (0{,}536 \times 0{,}611) + (0{,}485 \times 0{,}578)$$
$$+(0{,}535 \times 0{,}587) + (0{,}481 \times 0{,}664) + (0{,}497 \times 0{,}549)$$
$$+(0{,}538 \times 0{,}672) + (0{,}483 \times 0{,}550) + (0{,}470 \times 0{,}678)$$
$$+(0{,}531 \times 0{,}536) + (0{,}606 \times 0{,}707) = 3{,}180$$

Hasil perhitungan dari seluruh *sample* data pada data latih ditunjukkan pada Tabel 4.34.

Tabel 4.34 Hasil Perkalian Matriks H(x) *Transpose* dengan Matriks H(x)

Matriks $(H(x)^T.H(x))$	1	2
1	2,687	3,180
2	3,180	3,802

Setelah mendapatkan hasil perkalian matriks $(H(x)^+ H(x))$, langkah selanjutnya yaitu mencari matriks *inverse* dari matriks tersebut yang hasilnya ditampilkan pada Tabel 4.35.

Tabel 4.35 Hasil Matriks *Inverse* dari Perkalian Matriks H(x) *Transpose* dengan Matris H(x)

Matriks $(H(x)^T.H(x))^{-1}$	1	2
1	36,753	-30,740
2	-30,740	25,973

Setelah mendapatkan matriks *inverse*, langkah selanjutnya adalah mengalikan matriks *inverse* tersebut dengan matriks H(x) *transpose*. Berikut merupakan perhitungan manual pada perkalian kedua matriks untuk beberapa *sample* data pada data latih:

$$H^+_{(1,1)} = (36{,}753 \times 0{,}536) + ((-30{,}740) \times 0{,}611)) = 0{,}925$$
$$H^+_{(1,2)} = (36{,}753 \times 0{,}485) + ((-30{,}740) \times 0{,}578)) = 0{,}087$$

Untuk hasil perhitungan manual matriks *Moore Penrose Pseudo*-Inverse (H^+) terhadap seluruh data latih dapat dilihat pada Tabel 4.36.

Tabel 4.36 Hasil Matriks *Moore Penrose Pseudo*-Inverse

Matriks $H^+(x)$	1	2	3	4	5	...	10
1	0,925	0,087	1,609	-2,734	1,394	...	0,536
2	-0,613	0,078	-1,191	2,462	-1,022	...	-0,262

4.4.3.4 Perhitungan Manual Matriks *Output Weight*

Pada manualiasasi untuk menentukan matriks *output weight* (β) dibutuhkan masukan berupa matriks H^+ dan nilai target pada data latih. Dalam menentukan matriks β ini didapatkan dari perhitungan matriks H^+ yang dikalikan dengan nilai target pada data latih. Hasil dari proses ini menghasilkan matriks β yang selanjutnya digunakan dalam perhitungan pada proses *testing* data. Representasi dari nilai target pada data latih ditunjukkan pada Tabel 4.37.

Tabel 4.37 Nilai Target pada Data Latih

Nilai Target (T)	1
1	0,303
2	0,055
3	0,414
4	0,230
5	0,429
6	0,271
7	1
8	0,374
9	0
10	0,954

Langkah selanjutnya adalah mengalikan matriks H^+ dengan nilai target (T), berikut contoh perhitungan manualnya menggunakan Persamaan 2.5:

$$\beta_{(1,1)} = (0{,}925 \times 0{,}303) + (0{,}087 \times 0{,}055) + (1{,}609 \times 0{,}414)$$
$$+\big((-2{,}734) \times 0{,}230\big) + (1{,}394 \times 0{,}429) + ((-0{,}860) \times 0{,}271)$$
$$+(0{,}840 \times 1) + \big((-3{,}356) \times 0{,}374\big) + (3{,}067 \times 0)$$
$$+(0{,}536 \times 0{,}954) = 0{,}704$$

Hasil dari perhitungan nilai *output weight* ditunjukkan pada Tabel 4.38.

Tabel 4.38 Hasil *Output Weight*

Matriks β	1
1	0,704
2	0,072

4.4.4 Proses *Testing* Data

Pada proses *testing* data dalam implementasi metode ELM dilakukan beberapa langkah, yaitu perhitungan matriks inisialisasi *output hidden layer*, pernitungan matriks *output hidden layer* dengan fungsi aktivasi *sigmoid biner*, perhitungan hasil prediksi, dan perhitungan nilai *MAPE*. Hasil dari proses *testing* ini adalah nilai *MAPE* (%) yang merepresentasikan akurasi dari implementasi metode yang digunakan terhadap peramalan kasus pada penelitian ini. Pada proses *testing* ini menggunakan *2 sample* data uji yang dapat dilihat pada Tabel 4.39.

Tabel 4.39 Data Uji

Data Uji	X1	X2	X3	X4	X5	X6
Nov-16	0,429	0,271	1	0,374	1	0,954
Dec-16	0,271	1	0,374	1	0,954	0,394

4.4.4.1 Perhitungan Manual Matriks Inisialisasi *Output Hidden Layer*

Pada manualiasasi untuk menentukan matriks inisialisasi *output hidden layer* (H_{init_test}), dibutuhkan masukan berupa data *testing* yang telah dinormalisasi, data bobot input yang telah di*transpose*. dan bias. Dalam menentukan matriks H_{init_test} ini dilakukan perkalian terhadap data *testing* yang telah dinormalisasi dengan data bobot input yang telah di*transpose*, kemudian hasilnya dijumlahkan dengan nilai bias yang sebelumnya telah diinisialisasi. Hasil dari proses ini akan menghasilkan matriks H_{init_test} yang selanjutnya digunakan dalam perhitungan matriks *output hidden layer*.

Berikut merupakan perhitungan manual matriks H_{init} ternadap data latih menggunakan Persamaan 2.6:

$$H_{init\_test(1,1)} = [(0{,}429 \times (-0{,}256)) + (0{,}271 \times 0{,}145) + (1 \times (-0.662)) + (0{,}374 \times 0{,}319) + (0 \times 0{,}411) + (0{,}954 \times (-0{,}395))] + 0{,}135 = -0{,}854$$

Hasil perhitungan manual matriks H_{init} terhadap seluruh data latih dapat dilihat pada Tabel 4.40.

Tabel 4.40 Matriks Bobot Inisialisasi *Output Hidden Layer*

Matriks $H_{init\_test}(x)$	1	2
1	-0,855	0,410
2	0,199	0,368

4.4.4.2 Perhitungan Manual Matriks *Output Hidden Layer*

Pada manualiasasi untuk menentukan matriks *output hidden layer* $H(x)_{\_test}$, dibutuhkan masukan dari matriks inisialisasi *output hidden layer* (H_{init_test}). Dalam menentukan matriks *output hidden layer* dilakukan menggunakan fungsi aktivasi *sigmoid biner*. Hasil dari proses ini menghasilkan matriks $H(x)_{\_test}$ yang selanjutnya digunakan pada perhitungan hasil prediksi.

Berikut merupakan perhitungan manual matriks H(x) terhadap data latih menggunakan Persamaan 2.6:

$$H(x)\_test_{(1,1)} = \frac{1}{1 + exp^{-(-0,854)}} = 0,298$$

Hasil perhitungan manual matriks $H(x)_{\_test}$ terhadap seluruh data latih dapat dilihat pada Tabel 4.41.

Tabel 4.41 Matriks *Output Hidden Layer*

Matriks $H(x)_{\_test}$	1	2
1	0,298	0,601
2	0,549	0,591

4.4.4.3 Perhitungan Manual Hasil Prediksi

Pada manualiasasi untuk menentukan hasil prediksi dibutuhkan masukan berupa matriks *output hidden layer* dari proses *testing* ($H(x)_{\_test}$) dan matriks *output weight* (β) yang diperoleh dari proses *training*. Dalam menentukan hasil prediksi dilakukan dengan mengalikan matriks $H(x)_{\_test}$ dengan matriks β. Hasil dari proses ini berupa matriks hasil prediksi namun belum didenormalisasi.

Berikut merupakan perhitungan manual hasil prediksi terhadap data uji menggunakan Persamaan 2.7:

$$\check{T}_{(1,1)} = \big(0,298 \times (0,704)\big) + (0,601 \times 0,072) = 0,253$$

Hasil perhitungan manual matriks $\check{T}(x)$ terhadap seluruh data latih dapat dilihat pada Tabel 4.42.

Tabel 4.42 Matriks Hasil Prediksi

Matriks $\hat{T}(x)$	1
1	0,253
2	0,429

4.4.4.4 Perhitungan Manual Denormalisasi

Perhitungan denormalisasi dilakukan pada data hasil prediksi yang berfungsi untuk mengembalikan nilai asli yang sebelumnya telah mengalami proses normalisasi. Dalam proses denormalisasi ini diperlukan masukan berupa data hasil prediksi, nilai minimal, dan nilai maksimal pada kolom target dari keseluruhan *dataset*. Proses perhitungannya adalah dengan mengalikan nilai prediksi yang belum didenormalisasi dengan selisih dari nilai minimal dan nilai maksimal pada *dataset*, setelah itu hasilnya dijumlahkan dengan nilai minimal.

Berikut merupakan perhitungan manual denormalisasi terhadap hasil prediksi pada data uji menggunakan Persamaan 2.8:

$$d_{(1,1)} = 0{,}253\ (889.451 - 649.792) + 649.792 = 710.543$$

Hasil proses denormalisasi terhadap seluruh data hasil prediksi dapat dilihat pada Tabel 4.43.

Tabel 4.43 Hasil Proses Denormalisasi Data Prediksi

Matriks $d(x)$	1
1	710.543
2	752.791

4.4.4.5 Perhitungan Manual Nilai *MAPE*

Hasil perhitungan nilai *MAPE* (%) merupakan representasi dari akurasi terhadap implementasi metode ELM yang digunakan terhadap permalan kasus pada penelitian ini. Sebelum menentukan nilai *MAPE*, dilakukan perhitungan nilai *error* terlebih dahulu, yaitu selisih antara data aktual dan data prediksi. Hasil dari selisih data aktual dan data prediksi kemudian dibagi dengan data aktual, selanjutnya hasil yang diperoleh disebut dengan nilai *error*. Representasi data aktual yang selanjutnya akan dibandingkan dengan data prediksi dapat dilihat pada Tabel 4.44.

Tabel 4.44 Data Aktual

Data Aktual (T)	1
1	744.442
2	756.324

Berikut merupakan perhitungan nilai selisih terhadap data aktual dan data prediksi menggunakan Persamaan 2.9:

$$selisih_{(1,1)} = |733{,}442 - 710{,}543| \div 733{,}442 = 0.045$$

Hasil perhitungan nilai selisih terhadap seluruh data aktual dan data prediksi dapat dilihat pada Tabel 4.45

Tabel 4.45 Hasil Nilai Selisih

Nilai *Error*	1
1	0,045
2	0,004

Dalam menentukan nilai *MAPE*, digunakan total dari nilai *error* yang sebelumnya telah diperoleh, kemudian total tersebut dibagi dengan banyaknya data latih, yaitu 2 data kemudian hasilnya dikalikan dengan angka 100. Berikut merupakan perhitungan nilai *MAPE* menggunakan Persamaan 2.9:

$$\boldsymbol{MAPE}(\%) = [(0{,}045 + 0{,}004) \div 2] \times 100 = \mathbf{2{,}51}\%$$

4.5 Pengujian Algoritme Menggunakan Data *Single Variable* dengan Metode *ELM*

Tahapan pengujian algoritme dalam suatu penelitian berfungsi untuk mengetahui hasil kinerja sistem berdasarkan metode yang diimplementasikan terhadap objek penelitian. Pada pengujian algoritme ini, penulis akan melakukan pengujian dengan tiga skenario yang berbeda. Berikut merupakan penjelasan mengenai skenario pengujian yang akan dilakukan:

1. Skenario pengujian jumlah *neuron* pada *input layer*.
2. Skenario pengujian jumlah *neuron* pada *hidden layer*.
3. Skenario pengujian dalam perbandingan jumlah data *training* dan data *testing* yang digunakan.

4.5.1 Pengujian Jumlah *Neuron* pada *Input Layer*

Pengujian terhadap jumlah *neuron* pada *input layer* berfungsi untuk mencari nilai *MAPE* terbaik. Pada pengujian tahap ini dilakukan pengujian terhadap jumlah *neuron* mulai dari 1 hingga 10. Pengujian terhadap jumlah *neuron* ini dilakukan perulangan sebanyak 10 kali untuk mendapatkan nilai rata-rata *MAPE* dengan masukan berupa seluruh *dataset* yang telah dibagi menjadi data latih dan data uji.

Tabel 4.46 Rancangan Skenario Pengujian Jumlah *Neuron* pada *Input Layer*

Percobaan ke-	Nilai *MAPE* pada Jumlah *Input Neuron*									
	1	2	3	4	5	6	7	8	9	10

1										
2										
3										
4										
5										
6										
7										
8										
9										
10										
Rata-rata nilai *MAPE*										

4.5.2 Pengujian Jumlah *Neuron* pada *Hidden Layer*

Dalam melakukan pengujian terhadap jumlah *neuron* berfungsi untuk mencari nilai *MAPE* yang terbaik, yaitu nilai *MAPE* yang terkecil. Pada pengujian jumlah *neuron* ini menggunakan jumlah *neuron*, yaitu 1 sampai 10 *neuron*. Pengujian terhadap jumlah *neuron* ini dilakukan sebanyak 10 kali dengan masukan berupa seluruh *dataset* yang telah dibagi menjadi data latih dan data uji, serta hasil dari pengujian sebelumnya yaitu jumlah *neuron* yang terbaik pada *input layer*.

Tabel 4.47 Rancangan Skenario Pengujian Jumlah *Neuron* pada *Hidden Layer*

Percobaan ke-	**Nilai *MAPE* pada Jumlah *Hidden Neuron***									
	1	2	3	4	5	6	7	8	9	10
1										
2										
3										
4										
5										
6										
7										
8										

9										
10										
Rata-rata nilai *MAPE*										

4.5.3 Pengujian Perbandingan Jumlah Data *Training* dan Data *Testing*

Dalam melakukan pengujian terhadap perbandingan jumlah data *training* dan data *testing* berfungsi untuk mencari nilai *MAPE* yang terbaik, yaitu nilai *MAPE* terkecil. Pada proses pengujian ini digunakan masukan berupa seluruh *dataset* dan hasil dari pengujian sebelumnya yaitu jumlah *neuron* yang terbaik pada *input layer* dan *hidden layer*. Pengujian ini akan menggunakan perbandingan antara data *training* dan data *testing* yang berbeda, yaitu 5 : 20, 8 : 17, 10 : 15, 15 : 10, 18 : 7, dan 20 : 5.

Tabel 4.48 Rancangan Skenario Pengujian Perbandingan Jumlah Data *Training* dan Data *Testing*

Percobaan ke-	**Nilai *MAPE* pada Perbandingan Jumlah Data *Training* dan Data *Testing***					
	5 : 20	8 : 17	10 : 15	15 : 10	18 : 7	20 : 5
1						
2						
3						
4						
5						
6						
7						
8						
9						
10						
Rata-rata nilai *MAPE*						

BAB 5 IMPLEMENTASI

Pada bab ini akan dibahas mengenai implementasi sistem berdasarkan perancangan yang telah dilakukan pada Bab 4 Perancangan. Implementasi pada bagian ini menggunakan metode *Extreme Learning Machine* (ELM) yang merupakan solusi dari permasalahan, yaitu dengan memprediksi jumlah debit air yang layak didistribusi dengan menggunakan data *multivariate* dan data *single variable*.

5.1 Implementasi Sistem Menggunakan Data *Multivariate*

5.1.1 Implementasi *Class* ELM

Pada implementasi *class* ELM dilakukan sebuah proses inisialisasi untuk fungsi ELM dengan menggunakan parameter yang telah ditentukan sebelumnya. Pada bagian ini juga dilakukan pembagian data *training* dan data *testing*, serta penentuan jumlah *neuron* pada *input layer*. Pada *class* ELM juga dilakukan inisialisasi dan pemanggilan seluruh fungsi yang terdapat pada metode ELM hingga pada tahap terakhir pada saat pemanggilan nila *MAPE*. Implementasi *class* ELM dapat dilihat pada Kode Sumber 5.1.

Function: Class ELM

```
1  import numpy as np
2  import pandas as pd
3  import random
4  class ELM:
5      def __init__(self, j, data, jumlahDataTraining = None):
6          self.j = j
7          if (pilih == "multi"):
8              self.min = np.zeros(len(dataMulti[0]))
9              self.max = np.zeros(len(dataMulti[0]))
10             if (jumlahDataTraining is None):
11                 self.dataTraining = dataMulti[range(0,26)]
12                 self.dataTesting = dataMulti[range(26,36)]
13             else:
14                 self.dataTraining = dataMulti[range(0,
   jumlahDataTraining)]
15                 self.dataTesting =
   dataMulti[range(jumlahDataTraining, 36)]
16             k = len(dataMulti[0])-1
17             dataNormMulti = self.normDataMulti(dataMulti)
18             if (jumlahDataTraining is None):
19                 self.dataTrainingNorm =
   dataNormMulti[range(0,26)]
20                 self.dataTestingNorm =
   dataNormMulti[range(26,36)]
21             else:
```

```
22                  self.dataTrainingNorm =
dataNormMulti[range(0, jumlahDataTraining)]
                    self.dataTestingNorm =
23  dataNormMulti[range(jumlahDataTraining, 36)]
24          elif pilih == "single":
25              self.min = np.zeros(dataSingle[0][0])
26              self.max = np.zeros(dataSingle[0][0])
27              if (jumlahDataTraining is None):
28                  self.dataTraining = dataSingle[range(0,20)]
29                  self.dataTesting = dataSingle[range(20,25)]
30              else:
31                  self.dataTraining = dataSingle[range(0,
jumlahDataTraining)]
32                  self.dataTesting =
dataSingle[range(jumlahDataTraining, 25)]
33              k = len(dataSingle[0])-1
34              dataNormSingle = self.normDataSingle(dataSingle)
35              if (jumlahDataTraining is None):
36                  self.dataTrainingNorm =
dataNormSingle[range(0,20)]
37                  self.dataTestingNorm =
dataNormSingle[range(20,25)]
38              else:
39                  self.dataTrainingNorm =
dataNormSingle[range(0, jumlahDataTraining)]
40                  self.dataTestingNorm =
dataNormSingle[range(jumlahDataTraining, 25)]
41          self.inputBobot = self.getBobotInput(j, k)
42          self.inputBias = self.getBias(j)
43          self.matriksOHL =
self.matriksOutputHiddenLayer(self.getBobotInput(j, k),
self.dataTrainingNorm, self.getBias(j))
44          self.matriksHT = self.matriksHT(self.matriksOHL)
45          self.matriksOWeight =
self.outputWeight(self.matriksHT)
46          self.matriksH_test =
self.matriksH_test(self.getBobotInput(j, k), self.getBias(j),
self.dataTestingNorm)
47          self.targetPrediksi =
self.targetPrediksi(self.matriksH_test, self.matriksOWeight)
48          self.denormPrediksi =
self.denormPrediksi(self.targetPrediksi)
49          self.nilaiError =
self.nilaiError(self.denormPrediksi)
50          self.nilaiMape = self.nilaiMape(self.nilaiError,
self.dataTesting)
```

Kode Sumber 5.1 Implementasi *Class* ELM

Penjelasan Kode Sumber 5.1 terhadap implementasi *class* ELM sebagai berikut:

1. Baris 1 – 3: Memasukkan beberapa *library* yang digunakan dalam proses implementasi metode ELM.
2. Baris 5 – 6: Inisialisasi *class* ELM dengan parameter j (jumlah *hidden layer*), jenis data yang digunakan, dan jumlah *data training*.
3. Baris 7: Kondisi dijalankan ketika memilih *String* "multi" yang menyatakan bahwa akan dilakukan perhitungan algoritme menggunakan data *multivariate*
4. Baris 10 – 12: Kondisi *if*, yaitu jika parameter jumlah data *training* dikosongkan maka akan otomatis mendeklarasikan data *training* yang digunakan yaitu data dengan *range* 0 hingga 26 dan data *testing* yang digunakan yaitu data dengan *range* 26 hingga 36.
5. Baris 13 – 15: Kondisi *else*, yaitu jika memasukkan parameter jumlah data *training*, maka pembagian data *training* dan data *testing* akan disesuaikan dengan parameter jumlah data *training* yang diinputkan.
6. Baris 16: Inisialisasi *variable* `k` (jumlah *input neuron*) dengan nilai sama dengan panjang dari jumlah kolom/fitur data dikurangi 1.
7. Baris 17: Memanggil *method* `normDataMulti` dengan parameter `dataMulti`, kemudian menyimpan hasilnya ke dalam *variable* `dataNormMulti`.
8. Baris 18 – 23: Seleksi kondisi untuk membagi data *training* dan data *testing* yang sudah dinormalisasi berdasarkan proses pada poin ke 4.
9. Baris 24: Kondisi dijalankan ketika memilih *String* "single" yang menyatakan bahwa akan dilakukan perhitungan algoritme menggunakan data *single variable*.
10. Baris 25 – 26: Inisialisasi nilai minimal dan maksimal terhadap data *single variable* yang digunakan pada fungsi selanjutnya.
11. Baris 27 – 32: Seleksi kondisi apabila di awal tidak ditentukan jumlah data *training* yang digunakan, maka otomatis akan menggunakan data *training* sebanyak 24 data dan data *testing* sebanyak 12 data. Apabila di awal ditentukan jumlah data *training* yang digunakan, maka sistem akan menggunakan data *training* sebanyak data yang ditentukan dan akan menggunakan sisa data sebagai data *testing*.
12. Baris 33: Inisialisasi nilai `k` (jumlah *input neuron*) yaitu sebanyak panjang kolom data dikurang 1.
13. Baris 34: Memanggil fungsi `normDataSingle` dengan parameter `dataSingle`.
14. Baris 35 – 40: Seleksi kondisi untuk membagi data *training* dan data *testing* yang sudah dinormalisasi berdasarkan proses pada poin ke 11.

15. Baris 41 – 50: inisialisasi dan pemanggilan seluruh fungsi yang terdapat pada metode ELM hingga menghasilkan nilai *MAPE*.

5.1.2 Implementasi Penentuan Bobot Input dan Bias

Penentuan bobot input dan bias dilakukan di awal karena digunakan dalam fungsi-fungsi yang ada dalam metode ELM. Inisialisasi bobot input dan bias dilakukan secara *random*, namun dengan Batasan nilai yang telah ditentukan. Implementasi penentuan bobot input dan bias dapat dilihat pada Kode Sumber 5.2.

Function: Inisialisasi Bobot Input dan Bias

```
1   def getBobotInput(self,j, k):
2       w = np.zeros([j,k])
3       for x in range(j):
4           for y in range(k):
5               w[x,y] = float(random.uniform(0, 1))
6       return w
7   def getBias(self,j):
8       b = np.zeros(j)
9       for x in range(j):
10          b[x] = float(random.uniform(0, 1))
11      return b
```

Kode Sumber 5.2 Implementasi Penentuan Bobot Input dan Bias

Penjelasan Kode Sumber 5.2 terhadap proses penentuan bobot input dan bias sebagai berikut:

1. Baris 1 – 6: Inisialisasi fungsi penentuan bobot input dengan parameter j (jumlah *input neuron*) dan k (jumlah *input neuron*). Bobot input yang dihasilkan merupakan nilai acak dari rentang 0 hingga 1.
2. Baris 7 – 11: Inisialisasi fungsi penentuan bias dengan parameter j (jumlah *input neuron*). Bias yang dihasilkan merupakan nilai acak dari rentang 0 hingga 1.

5.1.3 Implementasi Normalisasi Data

Proses normalisasi data dilakukan karena tiap data memiliki rentang yang berbeda dengan nilai yang cukup besar. Proses normalisasi akan sangat membantu terhadap proses penentuan fungsi aktivasi yang nantinya dilakukan pada fungsi tahap selanjutnya. Implementasi normalisasi data pada *data single variable* dan data *multivariate* dapat diilihat pada Kode Sumber 5.3.

Function: Normalisasi Data

```
1   def normDataSingle(self,data):
2       dataNormSingle =
            np.zeros((len(dataSingle),len(dataSingle[0])))
3       for i in range(len(dataSingle[0])):
4           self.min = dataSingle[0][0]
5           self.max = dataSingle[0][0]
6       for j in range(len(dataSingle)):
7           if self.max < dataSingle[j][i]:
8               self.max = dataSingle[j][i]
```

```
9           elif self.min > dataSingle[j][i]:
10              self.min = dataSingle[j][i]
11      for i in range(len(dataSingle[0])):
12          for j in range(len(dataSingle)):
13              dataNormSingle[j][i] = (dataSingle[j][i]-
                    self.min)/(self.max-self.min)
14      return dataNormSingle
15  def normDataMulti(self,data):
16      dataNormMulti =
    np.zeros((len(dataMulti),len(dataMulti[0])))
17      for i in range(len(dataMulti[0])):
18          self.min[i] = dataMulti[0][i]
19          self.max[i] = dataMulti[0][i]
20          for j in range(len(dataMulti)):
21              if self.max[i] < dataMulti[j][i]:
22                  self.max[i] = dataMulti[j][i]
23              elif self.min[i] > dataMulti[j][i]:
24                  self.min[i] = dataMulti[j][i]
25      for i in range(len(dataMulti)):
26          for j in range(len(dataMulti[0])):
27              dataNormMulti[i,j] = (dataMulti[i,j]-
    self.min[j])/(self.max[j]-self.min[j])
28      return dataNormMulti
```

Kode Sumber 5.3 Implementasi Normalisasi Data

Penjelasan Kode Sumber 5.3 terhadap proses normalisasi data sebagai berikut:

1. Baris 1 – 2: Inisialisasi fungsi normalisasi data *single variable* dengan parameter seluruh data yang digunakan, kemudian hasil normalisasi disimpan ke dalam *array* `dataNormSingle`.
2. Baris 3 – 10: Perulangan dan seleksi kondisi untuk menentukan nilai minimal dan nilai maksimal terhadap data yang digunakan.
3. Baris 11 – 14: Proses untuk normalisasi seluruh data *single variable* dengan menggunakan rumus *min-max normalization*.
4. Baris 15 – 16: Inisialisasi fungsi normalisasi data *multivariate* dengan parameter seluruh data yang digunakan, kemudian hasil normalisasi disimpan ke dalam *array* `dataNormMulti`.
5. Baris 17 – 24: Perulangan dan seleksi kondisi untuk penentuan nilai minimal dan nilai maksimal terhadap tiap fitur (tiap kolom) pada data.
6. Baris 25 – 28: Proses untuk normalisasi seluruh data *multivariate* dengan menggunakan rumus *min-max normalization*.

5.1.4 Implementasi Perhitungan Matriks *Output Hidden Layer*

Proses perhitungan matriks ini merupakan representasi dari perhitungan manual yang sebelumnya telah dijelaskan pada Bab 4 Perancangan. Terdapat dua kali proses perhitungan matriks *output hidden layer* ini pada implementasi metode ELM, yaitu pada proses *training* dan *testing*. Proses yang pertama dilakukan

terhadap proses *training* yang membutuhkan parameter bobot input, bias, dan data *training* yang telah dinormalisasi. Implementasi perhitungan matriks *output hidden layer* pada proses training dapat dilihat pada Kode Sumber 5.4.

Function: Matriks Output Hidden Layer

```
1  def matriksOutputHiddenLayer(self, w, dataTrainingNorm, b):
2      wTranspose = np.transpose(w)
3      hinit = []
4      for i in range(len(dataTrainingNorm)):
5          row = []
6          for j in range(len(wTranspose[0])):
7              total = 0
8              for k in range(0, len(dataTrainingNorm[0])-1):
9                  total = total + (dataTrainingNorm[i][k] * w
                       Transpose[k][j])
10             total = total + b[j]
11             row.append(total)
12         hinit.append(row)
13     hasil = 0
14     matriksH = []
15     for i in range(len(hinit)):
16         row = []
17         for j in range(len(hinit[i])):
18             hasil = 1/(1+np.exp(-hinit[i][j]))
19             row.append(hasil)
20         matriksH.append(row)
21     return matriksH
```

Kode Sumber 5.4 Implementasi Perhitungan Matriks *Output Hidden Layer*

Penjelasan Kode Sumber 5.4 terhadap proses perhitungan *output hidden layer* sebagai berikut:

1. Baris 1 – 12: Deklarasi fungsi matriks inisialisasi *output hidden layer* dengan parameter bobot input, bias, dan data *training* normalisasi. Proses perhitungan dilakukan dengan mengalikan seluruh data *training* normalisasi dengan bobot input yang di*transpose*. Kemudian hasilnya dijumlahkan dengan nilai bias.
2. Baris 13 – 21: Hasil perhitungan pada poin 1 digunakan pada proses perhitungan fungsi pada poin 2 (Matriks H), yaitu dengan memasukkan nilai pada persamaan fungsi aktivasi *sigmoid biner*.

5.1.5 Implementasi Perhitungan Matriks *Moore Penrose Pseudo-Inverse*

Proses perhitungan matriks ini merupakan representasi dari perhitungan manual yang sebelumnya telah dijelaskan pada Bab 4 Perancangan. Pada perhitungan ini dibutuhkan nilai yang sebelumnya telah dihasilkan pada proses implementasi matriks *output hidden layer*. Implementasi pernitungan matriks *Moore Penrose Pseudo-Inverse* dapat dilihat pada Kode Sumber 5.5.

Function: Matriks Moore Penrose Pseudo-Inverse

```
1  def matriksHT(self,matriksH):
2      matriksKali = []
3      hTranspose = np.transpose(matriksH)
```

```
4       for i in range(len(hTranspose)):
5           row = []
6           for j in range(len(matriksH[0])):
7               hasilKali = 0
8               for k in range(len(hTranspose[0])):
9                   hasilKali = hasilKali + (hTranspose[i][k] *
10                      matriksH[k][j])
11              row.append(hasilKali)
12          matriksKali.append(row)
13      matriksInverse = np.linalg.inv(matriksKali)
14      matriksHT = []
15      hasilMatriksHT = np.zeros((len(matriksInverse),
16      len(hTranspose)))
17      for x in range(len(matriksInverse)):
18          row1 = []
19          for y in range(len(hTranspose[0])):
20              hasilMatriksHT = 0
21              for z in range(len(matriksInverse[0])):
22                  hasilMatriksHT = hasilMatriksHT +
23                  (matriksInverse[x][z] * hTranspose[z][y])
24              row1.append(hasilMatriksHT)
25          matriksHT.append(row1)
26      return matriksHT
```

Kode Sumber 5.5 Implementasi Perhitungan *Moore Penrose Pseudo-Inverse*

Penjelasan Kode Sumber 5.5 terhadap proses perhitungan *Moore Penrose Pseudo-Inverse* sebagai berikut.

1. Baris 1 – 12: Proses perkalian matriks H *transpose* dengan matriks H. Kemudian mencari nilai invers dari hasil perkalian kedua matriks tersebut.
2. Baris 14 – 26: Proses perkalian hasil *inverse* matriks dengan matriks H *transpose*.

5.1.6 Implementasi Perhitungan Matriks *Output Weight*

Proses perhitungan matriks ini merupakan tahap terakhir pada proses *training* data. Selanjutnya hasil dari perhitungan matriks ini digunakan pada proses *testing* data. Pada proses perhitungan ini dibutuhkan hasil dari proses sebelumnya, yaitu matriks *Moore Penrose Pseudo-Inverse* dan nilai dari target pada data *training* yang telah dinormalisasi. Implementasi perhitungan matriks *Output Weight* dapat dilihat pada Kode Sumber 5.6.

Function: Matriks Output Weight

```
1   def outputWeight(self,matriksHT):
2       targetTraining = []
3       for target in self.dataTrainingNorm:
4           row = []
5           row.append(target[dataTrainingNorm.shape[1]-1])
6           targetTraining.append(row)
7       matriksOWeight = []
8       o_weigth = np.zeros((len(matriksHT),
        len(targetTraining)))
9       for i in range(len(matriksHT)):
10          row = []
```

```
11          for j in range(len(targetTraining[0])):
12              o_weight = 0
13              for k in range(len(matriksHT[0])):
14                  o_weight = o_weight + (matriksHT[i][k] *
                    targetTraining[k][j])
15              row.append(o_weight)
16          matriksOWeight.append(row)
17      return matriksOWeight
```

Kode Sumber 5.6 Implementasi Perhitungan Matriks *Output Weight*

Penjelasan Kode Sumber 5.6 terhadap proses perhitungan *output weight* sebagai berikut:

1. Baris 1 – 6: Proses pengambilan nilai target pada data *training*, yaitu mengambil kolom terakhir pada data *training* yang telah dinormalisasi.
2. Baris 7 – 17: Proses perhitungan matriks *output weight*, yaitu dengan mengalikan matriks *Moore Penrose Pseudo-Inverse* dengan nilai target pada data *training* yang telah dinormalisasi.

5.1.7 Implementasi Perhitungan Matriks *Output Hidden Layer* pada Proses *Testing*

Proses perhitungan matriks ini sama seperti pada proses *training*, bedanya pada bagian ini dilakukan dengan menggunakan data latih. Untuk parameter yang dilakukan sama seperti pada saat proses *training*, yaitu membutuhkan bobot input dan bias. Implementasi perhitungan matriks *output hidden layer* pada proses *testing* dapat dilihat pada Kode Sumber 5.7.

Function: Matriks Output Hidden Layer Testing

```
1   def matriksH_test(self,w, b, dataTestingNorm):
2       wTranspose = np.transpose(w)
3       hinit_test = []
4       for i in range(len(dataTestingNorm)):
5           row = []
6           for j in range(len(wTranspose[0])):
7               total = 0
8               for k in range(0, len(dataTestingNorm[0])-1):
9                   total = total + (dataTestingNorm[i][k] *
                    wTranspose[k][j])
10              total = total + b[j]
11              row.append(total)
12          hinit_test.append(row)
13      hasil = 0
14      matriksH_test = []
15      for i in range(len(hinit_test)):
16          row = []
17          for j in range(len(hinit_test[i])):
18              hasil = 1/(1+np.exp(-hinit_test[i][j]))
19              row.append(hasil)
20          matriksH_test.append(row)
21      return matriksH_test
```

Kode Sumber 5.7 Implementasi Perhitungan Matriks *Output Hidden Layer* pada Proses *Testing*

Penjelasan Kode Sumber 5.7 terhadap proses perhitungan *output hidden layer* sebagai berikut:

1. Baris 1 – 12: Deklarasi fungsi matriks inisialisasi *output hidden layer* dengan parameter bobot input, bias, dan data *testing* normalisasi. Proses perhitungan dilakukan dengan mengalikan seluruh data *testing* normalisasi dengan bobot input yang di*transpose*. Kemudian hasilnya dijumlahkan dengan nilai bias.
2. Baris 13 – 21: Hasil pernitungan pada poin 1 digunakan pada proses perhitungan fungsi pada poin 2 (Matriks H), yaitu dengan memasukkan nilai pada persamaan fungsi aktivasi *sigmoid biner*.

5.1.8 Implementasi Perhitungan Target Prediksi

Pada proses perhitungan target prediksi dilakukan dengan mengalikan hasil dari perhitungan proses sebelumnya (Matriks H *test*) dan hasil dari *output weight* yang sebelumnya telah didapatkan dari proses *training* data. Implementasi perhitungan target prediksi dapat dilihat pada Kode Sumber 5.8.

Function: Perhitungan Target Prediksi

```
1   def targetPrediksi(self,matriksH_test, matriksOWeight):
2       targetPrediksi = []
3       for i in range(len(matriksH_test)):
4           row = []
5           for j in range(len(matriksOWeight[0])):
6               hasilPrediksi = 0
```

```
7                 for k in range(len(matriksH_test[0])):
8                     hasilPrediksi = hasilPrediksi + (
                          matriksH_test[i][k] * matriksOWeight[k][j])
9                 row.append(hasilPrediksi)
10            targetPrediksi.append(row)
11        return targetPrediksi
```

Kode Sumber 5.8 Perhitungan Target Prediksi

Penjelasan Kode Sumber 5.8 terhadap proses perhitungan target prediksi sebagai berikut:

1. Baris 1 – 11: Proses inisialisasi fungsi target prediksi dengan parameter matriks H *test* dan matriks *output weight*. Hasil dari target prediksi didapatkan dari perhitungan dengan mengalikan matriks H *test* dan matriks *output weight*. Kemudian hasil dari perkalian ini digunakan pada proses selanjutnya.

5.1.9 Implementasi Proses Denormalisasi Data Prediksi

Proses denormalisasi dilakukan terhadap data hasil prediksi, yaitu nilai yang didapatkan berdasarkan perhitungan target prediksi pada proses sebelumnya. Proses denormalisasi ini dibutuhkan untuk mengembalikan nilai asli dari hasil prediksi yang sebelumnya telah melalui tahap normalisasi. Implementasi perhitungan denormalisasi terhadap data prediksi dapat dilihat pada Kode Sumber 5.9.

Function: Denormalisasi Data Prediksi

```
1   def denormPrediksi(self,targetPrediksi):
2       denormPrediksi = np.zeros((len(targetPrediksi),
3   (targetPrediksi[0])))
4       for i in range(len(targetPrediksi)):
5           for j in range(len(targetPrediksi[i])):
6               if pilih == "multi":
                    denormPrediksi[i][j] = (targetPrediksi[i][j]
                * (self.max[j] - self.min[j])) + self.min[j]
                elif pilih == "single":
7                   denormPrediksi[i][j] = (targetPrediksi[i][j]
                * (self.max - self.min)) + self.min
8       return denormPrediksi
```

Kode Sumber 5.9 Denormalisasi Data Prediksi

Penjelasan Kode Sumber 5.9 terhadap proses perhitungan target prediksi sebagai berikut:

1. Baris 1 – 7: Proses inisialisasi fungsi denormalisasi prediksi dengan parameter nilai dari target prediksi yang didapatkan dari proses sebelumnya. Proses denormalisasi ini dilakukan dengan menggunakan nilai maksimal dan minimal yang sebelumnya telah digunakan pada proses normalisasi data. Hasil dari denormalisasi kemudian disimpan ke dalam *array* `denormPrediksi`.
2. Baris 4 – 5: Proses perulangan pada baris dan kolom data `targetPrediksi`.
3. Baris 6: Kondisi *if* dijalankan ketika data yang diproses adalah data *multivariate* dengan melakukan proses denormalisasi menggunakan nilai

maksimal dan minimal yang sebelumnya telah didapatkan pada proses normalisasi data.

4. Baris 7: Kondisi *else* dijalankan ketika data yang diproses adalah data *single variable* dengan melakukan proses denormalisasi menggunakan nilai maksimal dan minimal yang sebelumnya telah didapatkan pada proses normalisasi data.
5. Baris 8: Mengembalikan nilai pada *variable* `denormPrediksi`.

5.1.10 Implementasi Perhitungan Nilai *Error*

Perhitungan nilai *error* merupakan selisih dari nilai target aktual dengan nilai target hasil prediksi. Hasil dari perhitungan nilai *error* ini digunakan pada tahap selanjutnya, yaitu pada proses perhitungan nilai *MAPE*. Implementasi perhitungan nilai *error* dapat dilihat pada Kode Sumber 5.10.

Function: Perhitungan Nilai Error

```
1  def nilaiError(self,denormPrediksi):
2      targetTesting = []
3      for target in self.dataTesting:
4          row = []
5          row.append(target[len(self.dataTesting[0])-1])
6          targetTesting.append(row)
7      nilaiError = np.zeros((len(denormPrediksi),
       len(denormPrediksi[0])))
8      for i in range(len(targetTesting)):
9          for j in range(len(denormPrediksi[0])):
10             nilaiError[i][j] = np.abs(targetTesting[i][j] -
               denormPrediksi[i][j])/targetTesting[i][j]
11     return nilaiError
```

Kode Sumber 5.10 Perhitungan Nilai *Error*

Penjelasan Kode Sumber 5.10 terhadap proses perhitungan nilai *error* sebagai berikut:

1. Baris 1 – 6: Proses pengambilan nilai target pada data *testing*, yaitu mengambil kolom terakhir pada data *training* yang selanjutnya disimpan ke dalam *array* dengan nama `targetTesting`.
2. Baris 7 – 11: Proses perhitungan nilai *error*, yaitu selisih dari target *testing* dan hasil prediksi yang telah didenormalisasi. Setiap melakukan perhitungan nilai selisih, hasilnya kemudian dibagi dengan nilai dari target *testing*. Hasil keseluruhan disimpan ke dalam *array* dengan nama `nilaiError`.

5.1.11 Implementasi Perhitungan Nilai *MAPE*

Proses perhitungan nilai *MAPE* merupakan tahap akhir dalam proses *testing*, sekaligus tahap terkahir dalam proses implementasi metode ELM. Perhitungan nilai *MAPE* ini membutuhkan nilai *error* yang sebelumnya telah didapatkan, kemudian dijumlahkan untuk mencari nilai rata-rata. Implementasi perhitungan nilai *MAPE* dapat dilihat pada Kode Sumber 5.11.

```
Function: Perhitungan Nilai MAPE
1  def nilaiMape(self,nilaiError, dataTesting):
2      mape = (np.sum(nilaiError)*100)/len(dataTesting)
3      return mape
```

Kode Sumber 5.11 Perhitungan Nilai *MAPE*

Penjelasan Kode Sumber 5.11 terhadap proses perhitungan nilai *error* sebagai berikut:

1. Baris 1 – 3: Proses inisialisasi fungsi nilai *MAPE* dengan parameter nilai *error* dan data *testing* yang akan diambil panjang datanya untuk menentukan nilai rata-rata. Nilai rata-rata kemudian dikalikan dengan 100 untuk menghasilkan nilai *MAPE*.

5.1.12 Implementasi *Main Class*

Pada bagian *main class* digunakan untuk menjalankan seluruh kode program. Pada bagian ini juga dilakukan *import* data yang digunakan pada tahap implementasi, penginputan nilai j (nilai *input layer*), dan pemanggilan fungsi nilai *MAPE* terhadap *class* ELM. Implementasi *main class* dapat dilihat pada Kode Sumber 5.12.

```
Function: Main Class
1  pilih = input("Pilih data untuk diprediksi [single/multi]:
   ")
2  j = int(input("Masukkan nilai Hidden Neuron (j) = "))
3  if pilih == "single":
4      k = int(input("Masukkan nilai Input Neuron (k) = "))
5      if k == 1:
           dataSingle = pd.read_excel('pdam_single.xlsx',
   'Input 1', delimiter=";")
6      elif k == 2:
           dataSingle = pd.read_excel('pdam_single.xlsx',
   'Input 2', delimiter=";")
7      elif k == 3:
           dataSingle = pd.read_excel('pdam_single.xlsx',
   'Input 3', delimiter=";")
8      elif k == 4:
           dataSingle = pd.read_excel('pdam_single.xlsx',
   'Input 4', delimiter=";")
9      elif k == 5:
           dataSingle = pd.read_excel('pdam_single.xlsx',
10 'Input 5', delimiter=";")
       elif k == 6:
11         dataSingle = pd.read_excel('pdam_single.xlsx',
   'Input 6', delimiter=";")
       elif k == 7:
12         dataSingle = pd.read_excel('pdam_single.xlsx',
   'Input 7', delimiter=";")
       elif k == 8:
13         dataSingle = pd.read_excel('pdam_single.xlsx',
   'Input 8', delimiter=";")
```

```
        elif k == 9:
14          dataSingle = pd.read_excel('pdam_single.xlsx',
    'Input 9', delimiter=";")
        elif k == 10:
15          dataSingle = pd.read_excel('pdam_single.xlsx',
    'Input 10', delimiter=";")

        dataSingle = dataSingle.values
16
        elm1 = ELM(j, dataSingle)
17
    elif pilih == "multi":
18      dataMulti = pd.read_csv('pdam.csv', delimiter=";")
19      dataMulti = dataMulti.values
20      dataMulti = dataMulti.astype(float)
21
        elm1 = ELM(j, dataMulti)
22
    print("mape =", elm1.nilaiMape)
23
```

Kode Sumber 5.12 *Main Class*

Penjelasan Kode Sumber 5.12 terhadap implementasi *main class* sebagai berikut:

1. Baris 1: Menentukan jenis data yang akan diprediksi dengan memasukkan inputan *String* data *single variable* atau *multivariate*.
2. Baris 2: Memasukkan nilai *hidden neuron* yang kemudian disimpan pada variable `j`.
3. Baris 3: Kondisi dijalankan ketika memilih data *single variable* dengan memasukkan *String* "single".
4. Baris 4: Memasukkan nilai *input neuron* yang kemudian disimpan pada variable `k`.
5. Baris 5 – 15: Proses seleksi kondisi berdasarkan nilai inputan pada *variable* `k`, kemudian membaca file menggunakan *library pandas* dan mengambil data sesuai dengan jumlah *input neuron* yang diinginkan pada Baris 3.
6. Baris 16: Mengubah tipe data `dataSingle` menjadi *array*.
7. Baris 17: Menjalankan *class* ELM dengan parameter nilai `j` dan dengan menggunakan jenis data `dataSingle`.
8. Baris 18: Kondisi dijalankan ketika memilih data *multivariate* dengan memasukkan *String* "multi".
9. Baris 19 – 21: Membaca data *multivariate* dengan *library pandas*, kemudian mengubah tipe data menjadi *array* dan *convert* ke *float*.
10. Baris 22: Menjalankan *class* ELM dengan parameter nilai `j` dan dengan menggunakan jenis data `dataMulti`.
11. Baris 23: Mencetak nilai *MAPE*.

BAB 6 PENGUJIAN DAN HASIL ANALISIS

Pada bab ini akan dibahas mengenai pengujian dan hasil analisis dari implementasi metode ELM dalam memprediksi debit air yang layak didistribusi. Pada pengujian yang pertama dilakukan dengan menggunakan data *multivariate*, dengan skenario pengujian jumlah *neuron* pada *hidden layer* dan perbandingan jumlah data terhadap data *training* dan data *testing*. Pengujian yang kedua dilakukan dengan menggunakan data *single variable*, dengan skenario pengujian jumlah *neuron* pada *input layer*, jumlah *neuron* pada *hidden layer* dan perbandingan jumlah data terhadap data *training* dan data *testing*. Tujuan dilakukannya proses ini adalah untuk mengetahui nilai *MAPE* yang terkecil pada tiap pengujian yang dilakukan. Dalam masing-masing tahap pengujian bersifat sekuensial, yang berarti nilai *MAPE* yang terkecil berdasarkan pengujian yang pertama kali dilakukan akan digunakan pada pengujian tahap selanjutnya, yaitu dengan menggunakan parameter yang ditentukan pada pengujian sebelumnya.

6.1 Pengujian dengan Menggunakan Data *Multivariate*

6.1.1 Pengujian Terhadap Jumlah *Neuron* pada *Hidden Layer*

Pengujian dengan menggunakan parameter jumlah *hidden neuron* dilakukan karena berpengaruh pada saat pembentukan inisialisasi bobot input di awal yang digunakan selama proses implementasi ELM hingga menghasilkan nilai *MAPE*. Pada pengujian ini dilakukan percobaan selama 10 kali dengan memasukkan jumlah *neuron* yang berbeda-beda, yaitu angka 1 hingga 10. Pengujian tahap ini menggunakan 36 jumlah data, dengan pembagian 24 data *training* dan 12 data *testing*. Setelah dilakukan percobaan selama 10 kali, akan didapatkan nilai *MAPE* dan nilai rata-rata *MAPE* terhadap jumlah *neuron* yang digunakan yang dapat dilihat pada Tabel 6.1.

Tabel 6.1 Pengujian Terhadap Jumlah *Neuron* pada *Hidden Layer*

Percobaan ke-	Nilai *MAPE* pada Jumlah *Hidden Neuron*									
	1	2	3	4	5	6	7	8	9	10
1	12,119	24,925	17,95	5,325	21,398	20,392	31,278	40,72	47,865	60,131
2	20,475	15,67	26,923	28,169	37,469	52,142	46,975	29,724	25,938	53,929
3	24,682	27,088	11,564	19,583	11,074	49,197	26,284	39,164	32,662	32,979
4	35,335	17,694	35,355	25,198	34,468	29,866	39,265	58,427	51,693	36,569
5	21,398	21,13	6,167	5,044	22,183	19,642	26,377	47,176	58,723	44,944
6	22,472	12,838	8,5	9,201	15,303	15,498	54,485	56,813	34,324	55,805
7	18,029	13,303	17,231	11,732	17,404	17,248	49,387	41,173	30,925	82,541
8	17,617	14,066	28,998	14,873	20,06	27,245	51,134	31,865	47,076	49,425

9	17,444	14,011	12,096	22,162	9,909	42,48	27,66	44,566	46,089	87,342
10	27,236	9,471	21,447	21,864	21,698	33,305	26,127	49,01	59,814	71,315
Rata-rata nilai *MAPE*	21,680	17,019	18,623	**16,315**	21,096	30,701	37,897	43,809	43,510	57,498

Hasil dari pengujian terhadap jumlah *neuron* pada *hidden layer* direpresentasikan ke dalam bentuk grafik garis yang dapat dilihat pada Gambar 6.1.

Gambar 6.1 Grafik Hasil Pengujian Terhadap Jumlah *Neuron* pada *Hidden Layer*

Berdasarkan hasil grafik dari Gambar 6.1 didapatkan bahwa nilai rata-rata *MAPE* terbaik berada pada saat kondisi jumlah *neuron* 4, yaitu dengan menghasilkan rata-rata nilai *MAPE* sebesar 16,315%. Rata-rata nilai MAPE yang dihasilkan pada saat menggunakan jumlah *hidden neuron* 1 sampai 4 cenderung menghasilkan grafik yang menurun, yaitu menghasilkan nilai MAPE yang semakin rendah. Hal tersebut terjadi karena penggunaan jumlah *neuron* yang semakin tinggi berdampak pada jaringan yang akan semakin banyak menerima *inputan*, sehingga dapat lebih mengenal banyak pola data untuk menghasilkan nilai *MAPE* yang semakin baik. Namun, pada saat menggunakan jumlah *hidden neuron* 5 sampai 10, cenderung menghasilkan grafik yang kian naik atau dengan kata lain menghasilkan nilai rata-rata *MAPE* yang semakin besar dibandingkan dengan kondisi sebelumnya.

Hasil yang diperoleh pada pengujian ini disebabkan karena jika menggunakan banyak *neuron* pada *hidden layer* dapat menyebabkan *overfitting*. *Overfitting* dapat terjadi pada kondisi ketika menggunakan banyak *neuron* yang tidak diperlukan dalam suatu jaringan (Panchal and Panchal, 2014). Penggunaan

hidden neuron yang lebih banyak daripada kebutuhan akan menambahkan lebih banyak kompleksitas, sehingga penggunaan *hidden neuron* yang banyak lebih cocok digunakan ketika menghadapi keadaan kompleksitas yang cukup tinggi.

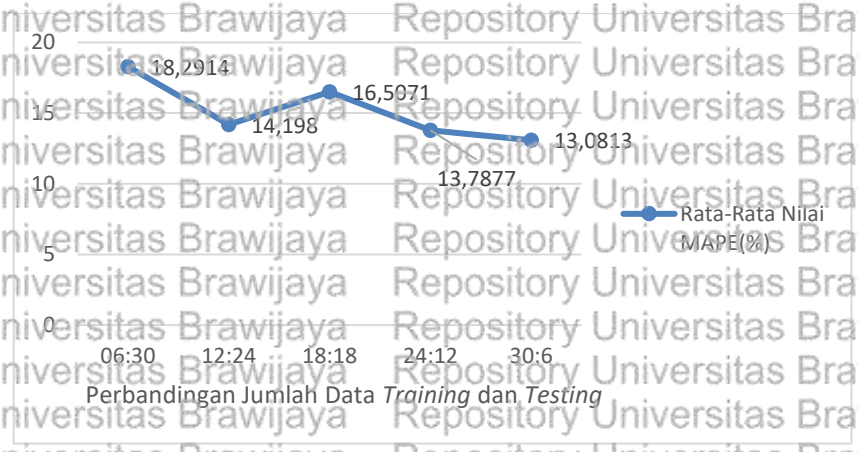
6.1.2 Pengujian Terhadap Perbandingan Jumlah Data *Training* dan Data *Testing*

Pengujian dengan menggunakan parameter perbandingan jumlah data *training* dan data *testing* yang bertujuan untuk mengetahui pengaruh jumlah data *training* dan data *testing* terhadap hasil akhir yang diperoleh. Pada pengujian ini dilakukan percobaan selama 10 kali dengan menggunakan jumlah *neuron* yang terbaik berdasarkan pengujian sebelumnya, yaitu 4 *neuron*. Pengujian tahap ini dilakukan sebanyak 5 pola perbandingan data *training* dan data *testing* yang berbeda. Setelah dilakukan percobaan selama 10 kali, akan didapatkan nilai *MAPE* dan nilai rata-rata *MAPE* terhadap masing-masing perbandingan jumlah data *training* dan data *testing* yang dapat dilihat pada Tabel 6.2.

Tabel 6.2 Pengujian Terhadap Perbandingan Jumlah Data *Training* dan Data *Testing*

Percobaan ke-	Nilai *MAPE* pada Perbandingan Jumlah Data *Training* dan Data *Testing*				
	6 : 30	12 : 24	18 : 18	24 : 12	**30 : 6**
1	14,695	11,048	12,531	13,015	13,274
2	21,102	23,562	8,72	8,291	8,536
3	16,749	8,433	6,829	16,784	10,968
4	11,063	20,193	19,03	8,741	17,073
5	27,75	19,004	15,477	13,854	5,696
6	19,309	6,008	21,951	19,289	13,901
7	9,251	23,306	21,118	17,125	13,231
8	19,007	6,721	17,882	18,979	18,275
9	25,724	15,724	15,968	12,163	16,456
10	18,264	7,981	25,565	9,636	13,403
Rata-rata nilai *MAPE*	18,291	14,19	16,507	13,787	**13,081**

Hasil dari pengujian terhadap perbandingan jumlah data *training* dan data *testing* direpresentasikan ke dalam bentuk grafik garis yang dapat dilihat pada Gambar 6.2.



Gambar 6.2 Grafik Hasil Pengujian Terhadap Perbandingan Jumlah Data *Training* dan Data *Testing*

Berdasarkan hasil grafik dari Gambar 6.2 didapatkan bahwa nilai rata-rata *MAPE* terbaik berada pada saat menggunakan data *training* sebanyak 30 data dan data *testing* sebanyak 6 data, yaitu dengan menghasilkan rata-rata nilai *MAPE* sebesar 13,081%. Berdasarkan hasil tersebut juga dapat dilihat bahwa penggunaan jumlah data *training* yang lebih banyak dibandingkan data *testing* akan menghasilkan nilai *MAPE* yang relatif lebih kecil, karena peran proses data latih sangat penting dalam hal pengenalan pola data dan juga dalam hal penerapan algoritme yang nantinya diuji pada tahap *testing* data. Berdasarkan hasil pengujian pada tahap ini, disimpulkan bahwa penggunaan data *training* yang lebih besar akan membantu proses implementasi metode ELM untuk menghasilkan kondisi yang terbaik, yaitu menghasilkan nilai *MAPE* yang terkecil.

6.2 Pengujian dengan Menggunakan Data *Single Variable*

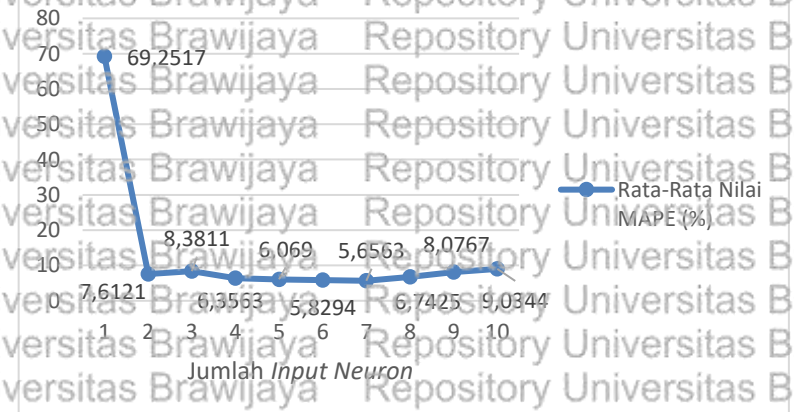
6.2.1 Pengujian Terhadap Jumlah *Neuron* pada *Input Layer*

Pengujian terhadap jumlah *input neuron* yang digunakan pada saat melakukan prediksi dengan data *single variable* berpengaruh terhadap setiap proses yang dijalankan, karena jika menggunakan data *single variable/time series* akan berpengaruh terhadap pola data yang digunakan ketika menggunakan *input neuron* yang berbeda-beda. Pada pengujian ini dilakukan percobaan selama 10 kali dengan memasukkan jumlah *neuron* yang berbeda-beda, yaitu angka 1, 2, 3, 4, 5, 6, 7, 8, 9, dan 10. Penggunaan data pada proses ini bergantung pada jumlah *neuron* yang digunakan. Setelah dilakukan percobaan selama 10 kali, akan didapatkan nilai *MAPE* dan nilai rata-rata *MAPE* terhadap jumlah *neuron* yang digunakan yang dapat dilihat pada Tabel 6.3.

Tabel 6.3 Pengujian Terhadap Jumlah *Input Neuron*

Percobaan ke-	Nilai *MAPE* pada Jumlah *Input Neuron*									
	1	2	3	4	5	6	7	8	9	10
1	82,161	5,475	8,439	6,157	5,127	7,09	2,851	9,243	10,922	20,187
2	84,593	6,131	14,34	8,827	5,553	6,302	6,365	8,199	6,41	10,667
3	46,843	7,48	10,29	5,726	7,188	4,166	7,537	7,163	14,936	12,516
4	56,021	8,849	4,981	10,312	8,19	5,357	2,774	9,916	5,255	7,197
5	50,815	12,655	3,631	3,963	4,528	6,087	3,263	4,915	9,207	4,658
6	42,982	6,679	11,99	8,562	5,341	5,003	7,107	4,517	6,085	6,58
7	54,303	7,421	9,905	4,05	8,704	9,479	6,044	6,242	7,244	7,554
8	87,167	8,787	8,733	7,579	5,451	5,558	7,415	4,939	5,579	9,62
9	96,798	3,368	5,721	2,427	6,266	4,39	5,734	6,538	6,518	6,347
10	90,834	9,276	5,781	5,96	4,342	4,862	7,473	5,753	8,611	5,013
Rata-rata nilai *MAPE*	69,251	7,612	8,381	6,356	6,069	5,829	**5,656**	6,742	8,076	9,034

Hasil dari pengujian terhadap perbandingan jumlah *neuron* pada *input layer* direpresentasikan ke dalam bentuk grafik garis yang dapat dilihat pada Gambar 6.3.



Gambar 6.3 Grafik Hasil Pengujian Terhadap Jumlah *Neuron* pada *Input Layer*

Berdasarkan hasil grafik dari Gambar 6.3 didapatkan bahwa nilai rata-rata *MAPE* terbaik berada pada saat menggunakan 7 *input neuron*, yaitu dengan menghasilkan rata-rata nilai *MAPE* sebesar 5,656%. Berdasarkan pengujian pada tahap ini terjadi dua kondisi, yaitu ketika jumlah *neuron* yang dimasukkan 1 sampai

7 cenderung menghasilkan kondisi nilai *MAPE* yang semakin kecil, sehingga semakin baik karena nilai *error* yang dihasilkan semakin kecil pula. Hal ini disebabkan karena semakin banyak *neuron* yang digunakan maka proses prediksi pada waktu selanjutnya akan lebih mudah karena semakin banyak data yang telah diobservasi. Namun, ketika menggunakan jumlah *neuron* 8 hingga 10 kembali menghasilkan nilai *MAPE* yang semakin besar dibandingkan dengan kondisi sebelumnya. Hal ini terjadi karena pada kondisi seperti ini penggunaan jumlah *neuron* pada *input layer* yang semakin banyak akan menyebabkan pengenalan pola data yang terlalu luas sedangkan kompleksitas data pada penelitian ini cukup rendah, sehingga dapat menghasilkan nilai *MAPE* yang semakin besar.

6.2.2 Pengujian Terhadap Jumlah *Neuron* pada *Hidden Layer*

Pengujian dengan menggunakan parameter jumlah *hidden neuron* dilakukan karena berpengaruh pada saat pembentukan inisialisasi bobot input di awal yang akan digunakan selama proses implementasi ELM hingga menghasilkan nilai *MAPE*. Pada pengujian ini dilakukan percobaan selama 10 kali dengan memasukkan jumlah *neuron* pada *input layer* berdasarkan kondisi terbaik pada hasil pengujian sebelumnya, yaitu 10 *input neuron.* Pengujian tahap ini menggunakan 25 jumlah data, dengan data *training* sebanyak 20 data dan data *testing* sebanyak 5 data. Setelah dilakukan percobaan selama 10 kali, akan didapatkan nilai *MAPE* dan nilai rata-rata *MAPE* terhadap jumlah *neuron* yang digunakan yang dapat dilihat pada Tabel 6.4.

Tabel 6.4 Pengujian Terhadap Jumlah *Neuron* pada *Hidden Layer*

Percobaan ke-	Nilai *MAPE* pada Jumlah *Hidden Neuron*									
	1	2	3	**4**	5	6	7	8	9	10
1	5,417	3,881	4,072	4,138	5,016	9,362	17,743	15,754	11,611	25,989
2	11,397	8,535	4,203	2,421	2,897	4,294	5,322	8,16	24,049	14,13
3	11,192	2,646	7,711	4,308	5,633	4,552	13,375	5,99	7,942	28,445
4	3,627	6,658	3,453	4,185	7,03	6,419	7,249	5,787	17,381	31,021
5	5,422	3,006	4,749	4,747	5,629	4,851	12,929	25,43	12,693	32,011
6	6,034	4,187	4,014	7,602	4,804	4,62	11,372	14,829	14,95	13,575
7	5,824	5,181	4,134	3,478	2,549	13,525	27,355	23,784	29,614	30,367
8	2,184	3,454	7,932	2,787	5,648	18,511	19,686	12,029	19,414	39,116
9	3,315	6,642	3,02	5,295	5,978	15,403	11,816	14,736	17,103	12,149
10	11,118	5,796	3,391	5,438	4,965	7,147	18,173	28,71	10,228	27,119
Rata-rata nilai *MAPE*	6,553	4,998	4,667	**4,439**	5,014	8,868	14,502	15,520	16,498	25,392

Hasil dari pengujian terhadap jumlah *neuron* pada *hidden layer* direpresentasikan ke dalam bentuk grafik garis yang dapat dilihat pada Gambar 6.4

Gambar 6.4 Grafik Hasil Pengujian Terhadap Jumlah *Neuron* pada *Hidden Layer*

Berdasarkan hasil grafik dari Gambar 6.1 didapatkan bahwa nilai rata-rata *MAPE* terbaik berada pada saat kondisi jumlah *neuron* 4, yaitu dengan menghasilkan rata-rata nilai *MAPE* sebesar 4,439%. Dalam pengujian ini rata-rata nilai *MAPE* yang dihasilkan pada saat menggunakan jumlah *hidden neuron* 1 sampai 4 cenderung menghasilkan grafik yang menurun, yaitu menghasilkan nilai *MAPE* yang semakin rendah. Hal tersebut terjadi karena penggunaan jumlah *neuron* yang semakin tinggi berdampak pada jaringan yang akan semakin banyak menerima *inputan*, sehingga dapat lebih mengenal banyak pola data untuk menghasilkan nilai *MAPE* yang semakin baik. Namun, pada saat menggunakan jumlah *hidden neuron* 5 sampai 10, cenderung menghasilkan grafik yang kian naik atau dengan kata lain menghasilkan nilai rata-rata *MAPE* yang semakin besar dibandingkan dengan kondisi sebelumnya. Hal tersebut terjadi karena pada proses perhitungan akan melibatkan semakin banyak masukan data, sehingga terjadi *overfitting* data yang juga terjadi karena masukan berupa *hidden neuron* yang semakin banyak namun tidak sesuai dengan kompleksitas data yang digunakan, karena jumlah data yang digunakan pada penelitian ini terbatas. Selain itu, menurut (Panchal and Panchal, 2014) *overfitting* juga terjadi ketika terlalu banyak menerima inputan *neuron* pada *hidden layer* yang sebenarnya tidak dibutuhkan dalam penyelesaian masalah terkait *neuron network*.

6.2.3 Pengujian Terhadap Perbandingan Jumlah Data *Training* dan Data *Testing*

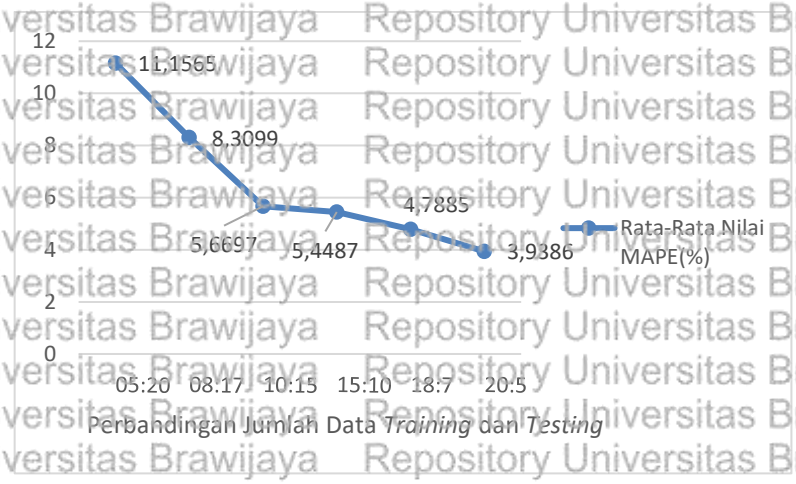
Pengujian dengan menggunakan parameter perbandingan jumlah data *training* dan data *testing* yang bertujuan untuk mengetahui pengaruh jumlah data *training* dan data *testing* yang digunakan terhadap hasil akhir yang diperoleh. Pada

pengujian ini dilakukan percobaan selama 10 kali dengan menggunakan jumlah *input neuron* dan *hidden neuron* yang terbaik berdasarkan pengujian sebelumnya, yaitu 7 *input neuron* dan 4 *hidden neuron*. Pengujian tahap ini dilakukan sebanyak 6 pola perbandingan data *training* dan data *testing* yang berbeda. Setelah dilakukan percobaan selama 10 kali, akan didapatkan nilai *MAPE* dan nilai rata-rata *MAPE* terhadap masing-masing perbandingan jumlah data *training* dan data *testing* yang dapat dilihat pada Tabel 6.5.

Tabel 6.5 Pengujian Ternadap Perbandingan Jumlah Data *Training* dan Data *Testing*

Percobaan ke-	Nilai *MAPE* pada Perbandingan Jumlah Data *Training* dan Data *Testing*					
	5 : 20	8 : 17	10 : 15	15 : 10	18 : 7	20 : 5
1	8,062	6,384	4,626	4,862	7,934	2,095
2	7,93	10,284	4,142	3,868	3,76	4,597
3	13,752	5,603	5,164	8,233	4,273	7,236
4	24,925	11,978	3,051	7,144	6,254	2,898
5	7,741	9,321	3,969	5,654	4,325	3,756
6	9,386	10,962	6,33	3,087	4,54	2,776
7	7,045	5,163	9,791	3,226	4,396	2,28
8	8,756	6,575	3,71	7,203	4,23	5,932
9	12,45	5,609	11,154	3,142	4,239	3,942
10	11,518	11,22	4,76	8,068	3,934	3,874
Rata-rata nilai *MAPE*	11,156	8,309	5,669	5,448	4,788	**3,938**

Hasil dari pengujian terhadap perbandingan jumlah data *training* dan data *testing* direpresentasikan ke dalam bentuk grafik garis yang dapat dilihat pada Gambar 6.5.



Gambar 6.5 Grafik Hasil Pengujian Terhadap Perbandingan Jumlah Data *Training* dan Data *Testing*

Berdasarkan hasil grafik dari Gambar 6.5 didapatkan bahwa nilai rata-rata *MAPE* terbaik berada pada saat menggunakan data *training* sebanyak 20 data dan data *testing* sebanyak 5 data, yaitu dengan menghasilkan rata-rata nilai *MAPE* sebesar 3,938%. Berdasarkan hasil tersebut juga dapat dilihat bahwa pada beberapa kondisi, penggunaan jumlah data *training* yang lebih banyak dibandingkan data *testing* akan menghasilkan nilai *MAPE* yang relatif lebih kecil. Berdasarkan hasil pengujian pada tahap ini, disimpulkan bahwa penggunaan data *training* yang lebih besar akan membantu proses implementasi metode ELM untuk menghasilkan kondisi yang terbaik, yaitu yang menghasilkan nilai *MAPE* yang terkecil.

6.3 Perbandingan Hasil Pengujian Data *Single Variable* dan Data *Multivariate*

Pada bagian ini, peneliti akan menjelaskan perbandingan hasil pengujian dari kedua jenis data yang telah dilakukan. Perbandingan hasil pengujian dilakukan untuk menilai performa yang dihasilkan dari kedua jenis data yang telah digunakan, yang kemudianberpengaruh pada *MAPE* terbaik yang dihasilkan. Perbandingan hasil pengujian yang dilakukan adalah pengujian terkait salah satu arsitektur JST, yaitu *hidden layer* dan pengujian terkait variasi jumlah data latih dan data uji.

6.3.1 Perbandingan Hasil Pengujian *Hidden Neuron*

Perbandingan hasil pengujian ini dilakukan berdasarkan pengujian jumlah *hidden neuron* yang sama pada kedua jenis data yang berbeda, yaitu *multivariate* dan *single variable*. Hasil dari perbandingan pengujian *hidden layer* direpresentasikan ke dalam bentuk grafik garis yang dapat dilihat pada Gambar 6.6.

Gambar 6.6 Grafik Perbandingan Hasil Pengujian *Hidden Neuron*

Berdasarkan hasil grafik dari Gambar 6.6 didapatkan bahwa dalam pengujian *hidden neuron*, penggunaan data *single variable* dan *multivariate* memperoleh kondisi dengan rata-rata nilai *MAPE* yang terbaik ketika menggunakan 4 *hidden neuron*. Berdasarkan aturan *rule-of-thumb* yang digunakan oleh (Karsoliya, 2012) untuk penentuan jumlah *neuron* pada *hidden layer* menjelaskan bahwa jumlah *neuron* yang tepat digunakan pada suatu *hidden layer* adalah 2/3 (atau 70% hingga 90%) dari ukuran *input layer*, yang dalam penelitian ini pada data *single variable* menggunakan 7 *input neuron* dan data *multivariate* menggunakan 8 *input neuron*. Pernyataan selanjutnya pada metode *rule-of-thumb* adalah jumlah *neuron* pada *hidden layer* harus lebih sedikit dari dua kali jumlah *neuron* pada *input layer*. Namun, terkadang pernyataan dalam metode *rule-of-thumb* tidak sepenuhnya benar karena penentuan jumlah *neuron* yang optimal pada *hidden layer* tidak hanya ditentukan berdasarkan parameter *input layer* saja, melainkan juga ditentukan oleh parameter lain yang digunakan, seperti fungsi aktivasi, arsitektur jaringan syaraf tiruan, dan algoritme pelatihan.

Selain itu, pada model data yang tidak stabil atau bersifat fluktuatif terdapat dua kondisi *neuron*, yaitu jumlah *neuron* yang terlalu besar atau terlalu kecil. Sebuah *trade-off* terbentuk yang salah satunya menyatakan bahwa ketika jumlah *neuron* pada *hidden layer* terlalu besar, maka akan menghasilkan *output neuron* yang tidak stabil (Karsoliya, 2012). Berdasarkan hasil perbandingan, diperoleh bahwa rata-rata nilai *MAPE* ketika menggunakan data *single variable* cenderung lebih baik daripada menggunakan data *multivariate*. Hal ini terjadi karena data *single variable* menggunakan nilai *input neuron* terbaik berdasarkan hasil pengujian yang dilakukan sebelumnya, sedangkan data *multivariate* tidak mengalami pengujian *input layer*, sehingga tidak bisa mendapatkan jumlah *input neuron* yang terbaik dan pada pengujian *hidden layer* tetap menggunakan 8 *input neuron* yang sesuai dengan jumlah atribut pada data *multivariate*.

6.3.2 Perbandingan Hasil Pengujian Terhadap Perbandingan Jumlah Data *Training* dan *Testing*

Perbandingan hasil pengujian ini dilakukan berdasarkan perbandingan jumlah data latih dan data uji yang digunakan pada kedua jenis data. Perbandingan jumlah data latih dan data uji yang digunakan pada bagian ini tidak sama dikarenakan jumlah data yang dimiliki pada kedua jenis data berbeda, yaitu pada data *single variable* memiliki 25 data dan data *multivariate* memiliki 36 data. Hasil dari pengujian perbandingan jumlah data latih dan data uji terhadap kedua jenis data dapat dilihat pada Gambar 6.7 dan Gambar 6.8.

Gambar 6.7 Grafik Hasil Pengujian Menggunakan Data *Single Variable*

Gambar 6.8 Grafik Hasil Pengujian Menggunakan Data *Multivariate*

Berdasarkan kedua grafik di atas didapatkan bahwa dalam perbandingan ini diperoleh hasil, yaitu kedua jenis data menghasilkan rata-rata nilai *MAPE* yang terbaik ketika menggunakan data latih lebih banyak daripada data uji. Pada hasil pengujian menggunakan data *single variable* diperoleh rata-rata nilai *MAPE* terbaik ketika menggunakan 20 data latih dan 5 data uji, yaitu 3,938%, sedangkan

ketika menggunakan data *multivariate* diperoleh rata-rata nilai *MAPE* terbaik ketika menggunakan 30 data latih dan 6 data uji, yaitu 13.081%. Penggunaan data latih yang lebih banyak sangat membantu dalam meminimalkan kesalahan (*error*) pada implementasi metode ELM, karena metode ELM merupakan salah satu metode JST yang menerapkan metode pembelajaran (proses *training*). Pada penelitian oleh (Sheela and Deepa, 2014) dijelaskan bahwa penggunaan data yang lebih banyak pada proses pelatihan akan membuat jaringan beradaptasi untuk mengurangi kesalahan pada saat pengenalan pola pelatihan.

Berdasarkan kedua grafik, dapat dilihat bahwa rata-rata nilai *MAPE* yang dihasilkan ketika menggunakan data *single variable* lebih baik daripada ketika menggunakan data *multivariate*. Hal ini dapat terjadi karena pada proses pengujian ini menggunakan parameter terbaik yang dihasilkan dari pengujian-pengujian sebelumnya, namun pada data *multivariate* hanya dilakukan 1 pengujian saja sebelum ini, yaitu pengujian *hidden layer*. Sehingga hasilnya tidak sebaik ketika menggunakan data *single variable* yang sebelumnya telah mengalami proses pengujian sebanyak 2 kali yang sangat berpengaruh terhadap hasil pengujian pada tahap ini.

BAB 7 PENUTUP

7.1 Kesimpulan

Berdasarkan hasil penelitian pada proses pengujian dan analisis terhadap prediksi jumlah debit air yang layak didistribusi menggunakan metode ELM, maka terdapat beberapa kesimpulan, diantaranya:

1. Metode *Extreme Learning Machine* (ELM) dapat diterapkan dalam memprediksi debit air yang layak didistribusi pada PDAM Kabupaten Gowa Makassar dengan menggunakan dua jenis data, yaitu data *single variable/time series* dan *multivariate*. Proses penerapan metode ELM pada kedua jenis data dilakukan dengan langkah penginputan *dataset*, inisialisasi bobot input dan bias, proses normalisasi data, proses *training* data, proses *testing* data, penentuan hasil prediksi, proses denormalisasi hasil prediksi, serta perhitungan nilai *MAPE*. Terdapat perbedaan proses ketika menggunakan dua jenis data yang berbeda, yaitu ketika proses normalisasi data pada data *multivariate* dilakukan terhadap masing-masing fitur yang ada, sedangkan pada data *single variable* proses normalisasi dilakukan terhadap seluruh data. Perbedaan yang selanjutnya terletak pada nilai *input neuron* yang digunakan, pada data *multivariate* menggunakan nilai *input neuron* yang konsisten sesuai dengan banyaknya fitur, yaitu 8. Sedangkan pada data *single variable* dapat menggunakan nilai *input neuron* sesuai masukan oleh *user*.
2. Perbandingan tingkat kesalahan prediksi metode ELM menggunakan data *single variable* dan *multivariate* berdasarkan hasil pengujian dan analisis adalah:
   a. Pada pengujian *input neuron* menggunakan data *single variable*, diperoleh nilai rata-rata MAPE terbaik yaitu 5,656% ketika menggunakan 7 *input neuron*. Penggunaan *neuron* yang semakin banyak akan menyebabkan proses prediksi pada waktu selanjutnya akan lebih mudah karena semakin banyak data yang telah diobservasi, namun penggunaan *neuron* yang berlebihan dapat menyebabkan pengenalan pola data yang terlalu luas sehingga menghasilkan nilai MAPE yang semakin besar.
   b. Pada pengujian *hidden neuron* menggunakan data *single variable*, diperoleh nilai rata-rata MAPE terbaik yaitu 4,439% ketika menggunakan 4 *hidden neuron*, sedangkan ketika menggunakan data *multivariate* diperoleh nilai rata-rata MAPE terbaik yaitu 16,315% dengan menggunakan *4 hidden neuron*. Penggunaan *neuron* yang semakin banyak akan menyebabkan *overfitting* dan kompleksitas yang semakin besar.
   c. Pada pengujian dalam penggunaan variasi data latih dan data uji yang menggunakan data *single variable*, diperoleh nilai rata-rata MAPE terbaik yaitu 3,938% dengan menggunakan 20 data latih dan data uji, sedangkan ketika menggunakan data *multivariate* diperoleh nilai rata-rata MAPE

terbaik yaitu 13,081% dengan menggunakan 30 data latih dan 6 data uji. Penggunaan data latih yang lebih banyak sangat membantu dalam proses pelatihan metode ELM untuk meminimalkan kesalahan.

d. Berdasarkan hasil perbandingan pada proses pengujian, diperoleh kesimpulan bahwa penerapan metode ELM dalam memprediksi debit air yang layak didistribusi menghasilkan nilai rata-rata MAPE terbaik ketika menggunakan data *single variable* dibandingkan menggunakan data *multivariate*.

7.2 Saran

Berdasarkan hasil penelitian prediksi debit air yang layak didistribusi dengan metode ELM, terdapat beberapa saran yang dapat dikembangkan untuk penelitian selanjutnya yang sejenis, yaitu melakukan pengujian *input neuron* pada jenis data *multivariate* menggunakan metode *correlation feature* atau metode seleksi fitur yang sejenis dan menerapkan algoritme optimasi agar hasil prediksi yang diperoleh semakin baik.

DAFTAR REFERENSI

Alfiyatin, A. N. *et al.* (2019) 'Penerapan Extreme Learning Machine (ELM) untuk Peramalan Laju Inflasi di Indonesia', *Jurnal Teknologi Informasi dan Ilmu Komputer*, 6(2), p. 179. doi: 10.25126/jtiik.201962900.

Ashar, N. M., Cholissodin, I. and Dewi, C. (2018) 'Penerapan Metode Extreme Learning Machine (ELM) Untuk Memprediksi Jumlah Produksi Pipa Yang Layak (Studi Kasus Pada PT . KHI Pipe Industries)', *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer (J-PTIIK) Universitas Brawijaya*, 2(11), pp. 4621–4628.

Cholissodin, I. and Riyandan, E. (2018) 'Analisis Big Data (Teori & Aplikasi) : Big Data vs Big Information vs Big Knowledge', *Fakultas Ilmu Komputer Universitas Brawijaya (Filkom UB)*, 1.01, pp. 1–476. Available at: https://www.academia.edu/36718594/Buku_Analisis_Big_Data.

Dewi, E. A. (2018) 'PERBANDINGAN METODE HOLT WINTER'S EXPONENTIAL SMOOTHING DAN EXTREME LEARNING MACHINE (ELM) PADA PERAMALAN PENJUALAN SEMEN'.

Ding, S. *et al.* (2015) 'Extreme learning machine: algorithm, theory and applications', *Artificial Intelligence Review*. Springer Netherlands, 44(1), pp. 103–115. doi: 10.1007/s10462-013-9405-z.

Huang, G.-B., Zhu, Q. and Siew, C. (2006) 'Extreme learning machine : Theory and applications', 70, pp. 489–501. doi: 10.1016/j.neucom.2005.12.126.

Jain, Y. K. and Bhandare, S. K. (2014) 'Min Max Normalization Based Data Perturbation Method for Privacy Protection', (4), pp. 45–50.

Julpan, Nababan, E. B. and Zarlis, M. (2015) 'Analisis Fungsi Aktivasi Sigmoid Biner Dan Sigmoid Bipolar Dalam Algoritma Backpropagation Pada Prediksi Kemampuan Siswa', *Jurnal Teknovasi*, 02(1), pp. 103–116.

Karsoliya, S. (2012) 'Approximating Number of Hidden layer neurons in Multiple Hidden Layer BPNN Architecture', *International Journal of Engineering Trends and Technology*, 3(6), pp. 714–717.

Monika, D. *et al.* (2019) 'Model Jaringan Syaraf Tiruan Dalam Memprediksi Ketersediaan Cabai Berdasarkan Provinsi', *Teknika*, 8(1), pp. 17–24. doi: 10.34148/teknika.v8i1.140.

Nugroho, N. A. and Purqon, A. (2015) 'Analisis 9 Saham Sektor Industri di Indonesia Menggunakan Metode SVR', pp. 295–300.

Panchal, F. S. and Panchal, M. (2014) 'International Journal of Computer Science and Mobile Computing Review on Methods of Selecting Number of Hidden Nodes in Artificial Neural Network', *International Journal of Computer Science and Mobile Computing*, 3(11), pp. 455–464. Available at: www.ijcsmc.com.

Prihatin, R. B. (2015) 'Penyediaan Air Bersih di Indonesia: Peran Pemerintah,

Pemerintah Daerah, Swasta, dan Masyarakat', p. 178.
Riandri, I. *et al.* (2018) 'Implementasi Algoritma Extreme Learning Machine pada Prediksi Aktivitas Badai Geomagnetik', 1(1), pp. 9–15.
Sel, I. (1940) 'Iruan 8', pp. 149–204.
Sheela, K. G. and Deepa, S. N. (2014) 'Selection of number of hidden neurons in neural networks in renewable energy systems', *Journal of Scientific and Industrial Research*, 73(10), pp. 686–688.
Siang, J. J. (2005) 'Pemrograman Menggunakan Matlab.pdf'.
Sumani, N. and Tadulako, U. (2018) *BUKU AJAR Penerbit 2018*.
Sun, Z. *et al.* (2008) 'Sales forecasting using extreme learning machine with applications in fashion retailing', *Decision Support Systems*. Elsevier B.V., 46(1), pp. 411–421. doi: 10.1016/j.dss.2008.07.009.
SUNARYO, M. T. (2015) *Analisis perhitungan harga pokok produksi air perusahaan daerah air minum (pdam) tirta jeneberang kabupaten gowa*.
Vong, C. *et al.* (2014) 'Neurocomputing Predicting minority class for suspended particulate matters level by extreme learning machine', *Neurocomputing*. Elsevier, 128, pp. 136–144. doi: 10.1016/j.neucom.2012.11.056.

LAMPIRAN

Waktu	Produksi	Kualitas Air							Distribusi
		Air Baku			Air Bersih				
		Kekeruhan	pH	Alkalinitas	Kekeruhan	pH	Klor Bebas	Alkalinitas	
Jan-16	831,217	85.38	6.86	52.1	3.5	6.67	0.21	47.16	701,357
Feb-16	783,260	97.71	6.93	49.02	18.37	6.47	0.12	44.01	660,451
Mar-16	806,496	94.61	7.18	49.44	3.7	7.01	0.39	45.02	681,519
Apr-16	870,287	80.9	7	47.63	3.94	6.81	0.34	44.19	733,636
May-16	842,618	33.26	6.73	54.83	3.69	6.64	0.28	49.63	710,767
Jun-16	820,895	9.65	6.59	51.38	2.62	6.67	0.1	46.17	690,780
Jul-16	854,348	16.73	6.59	43.39	2.67	6.59	0.08	37.67	722,415
Aug-16	789,168	6.71	6.55	46.1	2.33	6.57	0.15	43.17	663,192
Sep-16	891,202	7.53	6.65	44.92	2.06	6.63	0.28	40.28	749,229
Oct-16	838,792	14.85	6.67	41.52	2.82	6.7	0.24	39.1	704,996
Nov-16	895,474	32.78	6.7	200.69	3.58	6.64	0.29	39.21	752,692
Dec-16	849,715	60.07	6.91	37.91	3.6	6.81	0.22	35.67	714,909
Jan-17	1,052,047	46.53	6.87	40	3.75	6.8	0.21	37.04	889,451

Feb-17	875,446	351.12	7.46	34.04	4.07	6.83	0.17	29.67	739,609
Mar-17	768,736	163.06	7.1	35.58	3.94	6.81	0.19	33.13	649,792
Apr-17	1,040,227	43.52	6.86	38	3.9	6.72	0.26	36	878,588
May-17	880,679	20.67	6.82	47.69	2.81	6.79	0.45	43.65	744,442
Jun-17	895,308	37.19	6.9	44.5	3.1	6.79	0.26	43.29	756,324
Jul-17	879,915	20.42	6.86	44.71	3.8	6.89	0.18	42.43	744,264
Aug-17	886,117	8.57	6.85	45.64	2.93	6.87	0.3	44.98	749,141
Sep-17	939,325	6.67	6.67	69.97	2.21	6.75	0.37	66.18	794,890
Oct-17	895,236	1.99	6.56	77.46	1.28	6.67	0.31	79.37	756,323
Nov-17	932,678	9.11	6.56	46.54	2.42	6.65	0.22	44.9	788,024
Dec-17	921,021	72.08	7	40.3	5.48	6.89	0.34	38.43	778,515
Jan-18	911,724	75	7	35.3	4.1	6.9	0.3	33.8	764,496
Feb-18	973,313	101.82	7.09	68.35	4.28	6.83	0.44	64.09	815,730
Mar-18	878,643	70.19	7.71	53.54	4.27	6.8	0.35	49.16	736,393
Apr-18	963,059	30.37	6.8	35.35	4.33	6.7	0.34	31.8	807,422
May-18	922,421	16.26	6.76	45.18	3.67	6.77	0.37	41.26	773,588
Jun-18	1,010,664	16.77	6.8	45.05	5.19	6.8	0.31	41.07	847,819
Jul-18	907,700	53.42	6.98	48.59	4.7	6.84	0.24	44.9	761,373

Aug-18	926,739	2.98	6.75	48.45	10.96	6.75	0.25	47.92	777,882
Sep-18	1,002,901	5.16	6.68	82.75	2.4	6.68	0.28	81.72	841,220
Oct-18	994,564	3.4	6.8	24.9	1.6	6.8	1.1	19.2	833,826
Nov-18	1,302,233	3.94	6.71	10.64	2.81	6.74	0.47	8.83	866,024
Dec-18	986,003	45.3	6.6	19.1	5.6	6.7	0.2	18.7	826,278