LAMPIRAN A

SOURCE CODE ALGORITME EXTREME LEARNING MACHINE

Fungsi Main ELM

```
1   private void button4_Click(object sender, EventArgs e)
        {
2           timerr.Start();
3           long minMax = 2500;
4           //PengambilanParameter
5           int persentaseDataTraining =
    Convert.ToInt32(comboBox2.Text);
6           int persentaseDataTesting = 20;
7           int fitur = Convert.ToInt32(comboBox3.Text);
8           int neuron = Convert.ToInt32(comboBox4.Text);
9           Random ran = new Random();
10          //Preprocessing
11          long[,] dataFitur = membuatFitur(dataset,
    fitur);
12          long findMin = searchMin(dataFitur, minMax);
13          long findMax = searchMax(dataFitur, minMax);
14          double[,] normalisasi =
    normalisasiData(dataFitur, findMin, findMax);
15          double[,] bobot = generateBobot(fitur, neuron,
    ran);
16          double[,] bias = generateBias(neuron, ran);
17          //Proses Training
18          double[,] dataTrain = dataTraining(normalisasi,
    persentaseDataTraining, persentaseDataTesting);
19          double[,] aktualDataTrain =
    aktualDataTraining(normalisasi, persentaseDataTraining,
    persentaseDataTesting);
20          double[,] hinit = getHinit(dataTrain, bobot);
21          double[,] h = fungsiAktivasiHiddenLayer(hinit,
    bias);
22          double[,] hTranspose = transposeMatriks(h);
23          double[,] beforeInverse =
    perkalianMatriks(hTranspose, h);
24          double[,] inverseMatriks =
    inverseOBE(beforeInverse);
25          double[,] hDagger =
    perkalianMatriks(inverseMatriks, hTranspose);
26          double[,] beta = outputWeight(hDagger,
    aktualDataTrain);
27          //Proses Testing
28          double[,] dataTest = dataTesting(normalisasi,
    persentaseDataTesting);
29          long[,] aktualDataTest =
    aktualDataTesting(dataFitur, persentaseDataTesting);
30          double[,] hinitTest = getHinit(dataTest,
    bobot);
```

```
31              double[,] hTest =
    fungsiAktivasiHiddenLayer(hinitTest, bias);
32              double[,] yTopi = outputLayer(hTest, beta);
33              //Denormalisasi & Evaluasi MAPE
34              double[,] denormalisasi =
    denormalisasiData(yTopi, findMin, findMax);
35              double mape = evaluasiMAPE(denormalisasi,
    aktualDataTest);
36              //PrediksiBaru
37              long[,] dataPred = dataPrediksi(dataFitur);
38              double[,] normalisasiDataPred =
    normalisasiData(dataPred, findMin, findMax);
39              double[,] hinitPred =
    getHinit(normalisasiDataPred, bobot);
40              double[,] hPred =
    fungsiAktivasiHiddenLayer(hinitPred, bias);
41              double[,] yTopiPred = outputLayer(hPred, beta);
42              //Denormalisasi Hasil Prediksi
43              double[,] denormalisasiPred =
    denormalisasiData(yTopiPred, findMin, findMax);
44              long ambilPrediksi =
    Convert.ToInt64(denormalisasiPred[0, 0]);
45              timerr.Stop();
46  }
```

LAMPIRAN B

SOURCE CODE ALGORITME SUPPORT VECTOR REGRESSION

Fungsi Main SVR

```
1   private void button4_Click(object sender, EventArgs e)
2           {
3               timerr.Start();
4               long minMax = 2500;
5               //PengambilanParameter
6               double lamda =
    Convert.ToDouble(comboBox2.Text);
7               double sigma =
    Convert.ToDouble(comboBox3.Text);
8               double cLR = Convert.ToDouble(comboBox4.Text);
9               double complexity =
    Convert.ToDouble(comboBox5.Text);
10              double epsilon =
    Convert.ToDouble(comboBox6.Text);
11              long iterasi = Convert.ToInt64(comboBox7.Text);
12              int fitur = (dataset.GetLength(0) - 1) / 2;
13              //PreProcessing
14              double gamma = findGamma(cLR, lamda);
15              long[,] dataFitur = membuatFitur(dataset,
    fitur);
16              long findMin = searchMin(dataFitur, minMax);
17              long findMax = searchMax(dataFitur, minMax);
18              double[,] normalisasi =
    normalisasiData(dataFitur, findMin, findMax);
19              //ProsesTraining
20              double[,] jarak = hitungJarak(normalisasi);
21              double[,] rbf = gaussianRBF(jarak, sigma,
    lamda);
22              double[,] hasilAlfa = findAlfa(normalisasi,
    rbf, gamma, epsilon, complexity, iterasi);
23              double[,] alfa1 = ambilAlfa1(hasilAlfa);
24              double[,] alfa2 = ambilAlfa2(hasilAlfa);
25              //ProsesTesting & Evaluasi MAPE
26              long[,] dataAktual =
    ambilDataAktual(dataFitur);
27              double[,] hasilPrediksi = prediksi(rbf,
    hasilAlfa);
28              double[,] denormalisasi =
    denormalisasiData(hasilPrediksi, findMin, findMax);
29              double mape = evaluasiMAPE(denormalisasi,
    dataAktual);
30              //PrediksiBaru
31              long[,] dataBaru = dataPrediksi(dataFitur);
32              double[,] normalisasiDataBaru =
    normalisasiData(dataBaru, findMin, findMax);
33
```

```
            double[,] jarakBaru =
34  hitungJarakDataPrediksi(normalisasi, normalisasiDataBaru);
            double[,] rbfBaru = gaussianRBF(jarakBaru,
35  sigma, lamda);
            double[,] hasilPrediksiBaru = prediksi(rbfBaru,
36  hasilAlfa);
            double[,] denormalisasiBaru =
37  denormalisasiData(hasilPrediksiBaru, findMin, findMax);
            long ambilPrediksi =
38  Convert.ToInt64(denormalisasiBaru[0, 0]);
39          timerr.Stop();
40  }
```

LAMPIRAN C

DATA HARGA CABAI RAWIT

No	Tanggal	Cabai Rawit Merah
1	02/01/2020	50000
2	03/01/2020	55000
3	06/01/2020	75000
4	07/01/2020	75000
5	08/01/2020	75000
6	09/01/2020	75000
7	10/01/2020	75000
8	13/01/2020	75000
9	14/01/2020	75000
10	15/01/2020	90000
11	16/01/2020	90000
12	17/01/2020	90000
13	20/01/2020	85000
14	21/01/2020	90000
15	22/01/2020	95000
16	23/01/2020	100000
17	24/01/2020	100000
18	27/01/2020	100000
19	28/01/2020	100000
20	29/01/2020	100000
21	30/01/2020	100000
22	31/01/2020	100000
23	03/02/2020	100000
24	04/02/2020	100000
25	05/02/2020	100000
26	06/02/2020	100000

No	Tanggal	Cabai Rawit Merah
27	07/02/2020	100000
28	10/02/2020	80000
29	11/02/2020	80000
30	12/02/2020	80000
31	13/02/2020	75000
32	14/02/2020	70000
33	17/02/2020	70000
34	18/02/2020	70000
35	19/02/2020	70000
36	20/02/2020	60000
37	21/02/2020	50000
38	24/02/2020	55000
39	25/02/2020	55000
40	26/02/2020	55000
41	27/02/2020	55000
42	28/02/2020	50000
…	…	…
239	23/12/2020	60000
240	28/12/2020	70000
241	29/12/2020	70000
242	30/12/2020	80000
243	04/01/2021	90000
244	05/01/2021	100000
245	06/01/2021	100000
246	07/01/2021	100000
247	08/01/2021	95000

LAMPIRAN D

HASIL DATA PREDIKSI DAN DATA AKTUAL

MENGGUNAKAN ALGORITME EXTREME

LEARNING MACHINE

No	Data Prediksi	Data Akual
1	30019,78	30000
2	30019,78	30000
3	30019,78	30000
4	30019,78	30000
5	30019,78	32500
6	32866,17	35000
7	35392,06	35000
8	35040,36	35000
9	35040,36	35000
10	35040,36	37500
11	37915,73	37500
12	37548,89	37500
13	37548,89	37500
14	37548,89	37500
15	37548,89	37500
16	37548,89	37500
17	37548,89	37500
18	37548,89	37500
19	37548,89	35000
20	34684,82	40000
21	40815,16	40000
22	40055,3	40000
23	40055,3	40000

No	Data Prediksi	Data Aktual
24	40055,3	50000
25	51773,74	50000
26	50048,63	50000
27	50048,63	50000
28	50048,63	50000
29	50048,63	52500
30	52973,73	60000
31	61363,68	60000
32	59970,53	60000
33	59970,53	67500
34	68783,51	60000
35	58659,58	60000
36	59970,53	60000
37	59970,53	60000
38	59970,53	60000
39	59970,53	62500
…	…	…
44	69803,15	80000
45	81452,75	90000
46	90973,54	100000
47	100290,9	100000
48	98607,42	100000
49	98607,42	95000

LAMPIRAN E

HASIL DATA PREDIKSI DAN DATA AKTUAL MENGGUNAKAN ALGORITME SUPPORT VECTOR REGRESSION

No	Data Prediksi	Data Akual
1	30793,08	30000
2	30056,51	30000
3	30054,37	30000
4	30003,15	30000
5	30046,29	30000
6	30543,34	30000
7	32250,83	32500
8	32608,13	32500
9	32638,68	32500
10	32659,71	32500
11	31937,25	32500
12	28614,03	27500
13	27585,22	27500
14	27477,94	27500
15	27536,36	27500
16	28063,26	27500
17	29670,29	30000
18	30047,48	30000
19	30076,75	30000
20	30726,62	30000
21	32515,12	33500
22	31519,36	31000
23	31045,63	31000

No	Data Prediksi	Data Aktual
24	31019	31000
25	31055,94	31000
26	31075,9	31000
27	31086,42	31000
28	31019,72	31000
29	31061,1	31000
30	30495,14	31000
31	28384,87	27500
32	27554,85	27500
33	27452,1	27500
34	27555,2	27500
35	28157,73	27500
36	29624,23	30000
37	30112,1	30000
38	29989,32	30000
39	29996,52	30000
40	30006,89	30000
...	…	…
120	88919,61	90000
121	98162,47	100000
122	101480,9	100000
123	100067,1	100000
124	91608,07	95000

LAMPIRAN F

GRAFIK DATA PREDIKSI DAN DATA AKTUAL MENGGUNAKAN ALGORITME EXTREME LEARNING MACHINE

Grafik hasil prediksi algoritme ELM dihitung menggunakan parameter terbaik yang telah dilakukan uji coba. Berikut adalah grafik hasil prediksi pada algoritme ELM yang menghasilkan tingkat kesalahan terendah.

Jumlah Fitur : 2

Jumlah *Hidden Neuron* : 6

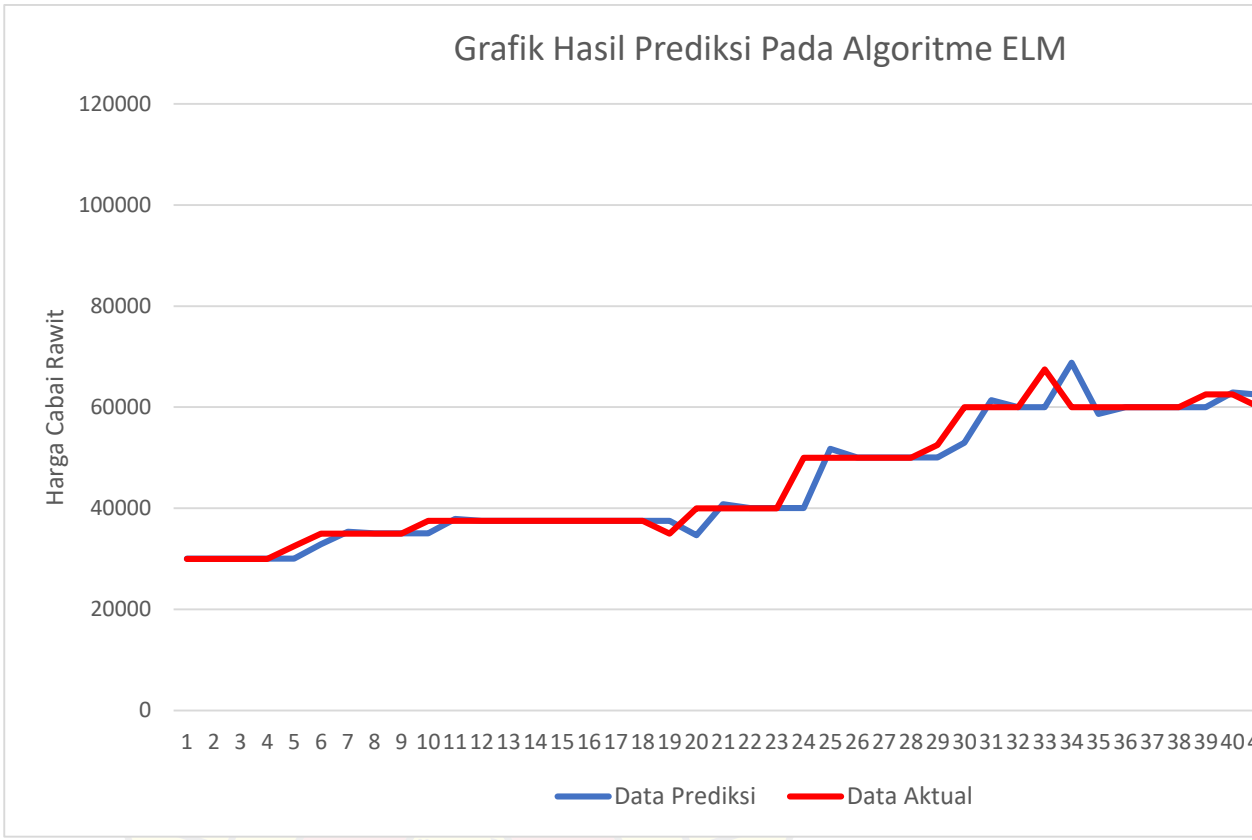
Fungsi Aktivasi : *Sigmoid Biner*

Persentase Jumlah Data *Training* : 80%

Persentase Jumlah Data *Testing* : 20%

Rata-rata nilai MAPE : 3,7249%

Waktu Komputasi : 0,0110832 *seconds*

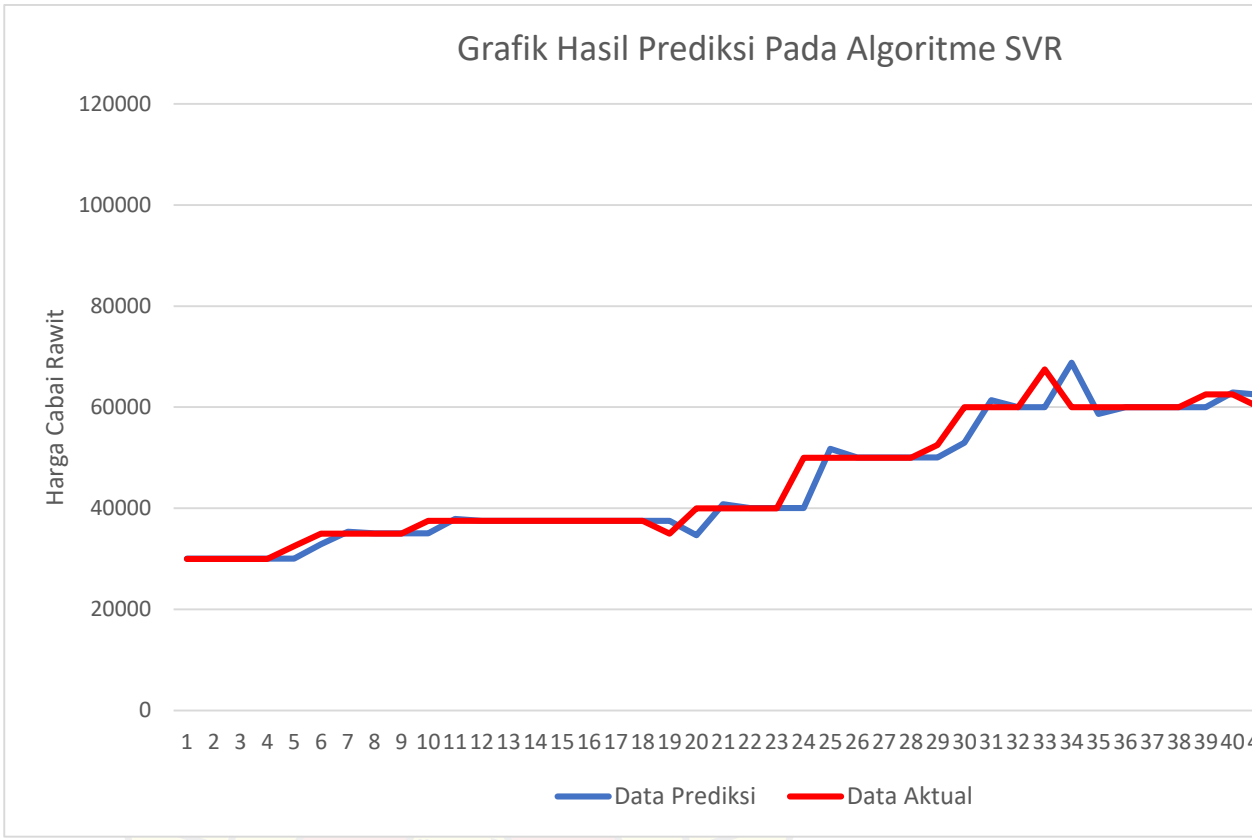

Grafik Hasil Prediksi Pada Algoritme ELM
Harga Cabai Rawit
120000
100000
80000
60000
40000
20000
0
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40
Data Prediksi
Data Aktual

LAMPIRAN G

GRAFIK DATA PREDIKSI DAN DATA AKTUAL MENGGUNAKAN ALGORITME SUPPORT VECTOR REGRESSION

Grafik hasil prediksi algoritme SVR dihitung menggunakan parameter terbaik yang telah dilakukan uji coba. Berikut adalah grafik hasil prediksi pada algoritme SVR yang menghasilkan tingkat kesalahan terendah.

Nilai *Lambda*	: 5,5
Nilai *Sigma*	: 0,5
Nilai *coefisien Learning Rate* (cLR)	: 0,01
Nilai Kompleksitas	: 100
Nilai *Epsilon*	: 0,000005
Jumlah iterasi	: 5000
Rata-rata nilai MAPE	: 1,4919%
Waktu Komputasi	: 3,0919356 *seconds*


Grafik Hasil Prediksi Pada Algoritme SVR
Harga Cabai Rawit
120000
100000
80000
60000
40000
20000
0
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40
Data Prediksi
Data Aktual

UNIVERSITAS
DARMA PERSADA

UNIVERSITAS DARMA PERSADA

Jl. Taman Malaka Selatan, Pondok Kelapa, Jakarta Timur, Indonesia 13450
Telp. (021) 8649051, 8649053, 8649057 Fax. (021) 8649052
E-mail : humas@unsada.ac.id Home page : http//www.unsada.ac.id

LEMBAR PERBAIKAN

SIDANG SKRIPSI

Nama : WAHYU EDI SURYANTO
NIM : 2016230056
Fakultas/Jurusan : Teknik / Teknologi Informasi
Tanggal : Senin, 22 Februari 2021

No.	Keterangan	Dosen
	Hal 118 tidak ada daftar Singkatan dan halaman hal 43 dan hal 24. hal 26 dan hal 24. hal 146 dan hal 24. penulisan diperbaiki sesuai dg perbaikan penguji	Setuju By Adam 30/3/21

Mengetahui, Kajur Teknologi Informasi

Adam Arif Budiman, M.Kom.

JURUSAN TEKNOLOGI INFORMASI
FAKULTAS TEKNIK
UNIVERSITAS DARMA PERSADA

LEMBAR BIMBINGAN

SKRIPSI

TEKNOLOGI INFORMASI – DARMA PERSADA

NIM : 2016230056

NAMA LENGKAP : Wahyu Edi Suryanto

DOSEN PEMBIMBING : Bagus Tri Mahardhika, M. Kom

JUDUL : Perbandingan Algoritme Extreme Learning Machine dengan Support Vector Regression Untuk Prediksi Harga Cabai Rawit di Kota Jakarta (Studi Kasus: Pasar Jatinegara, Jakarta Timur)

No.	Tanggal	Materi	Paraf Dosen Pembimbing
1	10 November 2020	Konsultasi Proposal Skripsi (Bab I)	
2	30 November 2020	Revisi Proposal Skripsi (Bab I)	
3	5 Desember 2020	Penyerahan Bab II	

4	17 Desember 2020	Penyerahan Bab III	
5	18 Desember 2020	Revisi Bab III	
6	25 Desember 2020	Demo Aplikasi	
7	5 Janurari 2021	Penyerahan Bab IV	
8	11 Januari 2021	Revisi Bab IV	
9	14 Januari 2021	Penyerahan Bab V	
10	16 Januari 2021	Konsultasi Persiapan Sidang Isi	

Jakarta, 5 Februari 2021

Dosen Pembimbing

Bagus Tri Mahardhika, MMSI.

LEMBAR PERNYATAAN

Saya yang bertanda tangan di bawah ini:

Nama : Wahyu Edi Suryanto

NIM : 2016230056

Fakultas : Teknik

Jurusan : Teknologi informasi

Judul Laporan : PERBANDINGAN ALGORITME EXTREME LEARNING MACHINE DENGAN SUPPORT VECTOR REGRESSION UNTUK PREDIKSI HARGA CABAI RAWIT DI KOTA JAKARTA (STUDI KASUS: PASAR JATINEGARA, JAKARTA TIMUR).

Menyatakan bahwa laporan tugas akhir ini saya susun sendiri berdasarkan hasil peninjauan, penelitian lapangan, wawancara serta memadukannya dengan buku buku literatur atau bahan-bahan referensi lain yang terkait dan relevan di dalam menyelesaikan laporan tugas akhir ini.

Demikian pernyataan ini saya buat dengan sesungguhnya.

Jakarta, 22 Februari 2021

Wahyu Edi Suryanto

LEMBAR PENGESAHAN

PERBANDINGAN ALGORITME EXTREME LEARNING MACHINE DENGAN SUPPORT VECTOR REGRESSION UNTUK PREDIKSI HARGA CABAI RAWIT DI KOTA JAKARTA

STUDI KASUS: PASAR JATINEGARA, JAKARTA TIMUR

Disusun oleh :

Nama : Wahyu Edi Suryanto

NIM : 2016230056

Bagus Tri Mahardhika, MMSI.	Adam Arif Budiman, S.T., M. Kom.
Pembimbing Laporan	Kajur Teknologi informasi

LEMBAR PENGUJI SKRIPSI

Laporan SKRIPSI yang berjudul :

"PERBANDINGAN ALGORITME EXTREME LEARNING MACHINE DENGAN SUPPORT VECTOR REGRESSION UNTUK PREDIKSI HARGA CABAI RAWIT DI KOTA JAKARTA (STUDI KASUS: PASAR JATINEGARA, JAKARTA TIMUR). " ini telah di ujikan pada tanggal

22 Februari 2021

Penguji 1	Penguji 2
Adam Arif Budiman, S.T., M. Kom.	Timor Setiyaningsih, S.T., M.TI

Penguji 1

22/02/2021
Suzuki Syofian, S.Kom., M.Kom.

Suzuki Syofian, S.Kom., M.Kom.

KATA PENGANTAR

Puji syukur penulis limpahkan kehadirat Allah SWT yang telah memberikan rahmat dan karunia-Nya sehingga penulis dapat menyelesaikan Laporan Tugas Akhir dengan judul *"PERBANDINGAN ALGORITME EXTREME LEARNING MACHINE DENGAN SUPPORT VECTOR REGRESSION UNTUK PREDIKSI HARGA CABAI RAWIT DI KOTA JAKARTA STUDI KASUS: PASAR JATINEGARA, JAKARTA TIMUR".* Penyusunan laporan tugas akhir ini bertujuan melengkapi jenjang Sarjana Strata 1 (S1) pada jurusan Teknologi informasi di Fakultas Teknik Universitas Darma Persada.

Penulis menyadari bahwa masih banyak terdapat kekurangan di dalam penyusunan Laporan Tugas Akhir ini, oleh karena itu penulis menerima semua kritik dan saran yang membangun. Dan diharapkan agar Laporan Tugas Akhir ini dapat memenuhi syarat yang diperlukan.

Dalam kesempatan ini, penulis ingin menyampaikan rasa hormat dan terima kasih kepada semua pihak yang telah memberikan bimbingan dan bantuan yang sangat berharga dalam menyelesaikan Laporan Tugas Akhir ini.

Penulis mengucapkan terima kasih yang sebesar-besarnya kepada:

4. Bapak Ir. Agus Sun Sugiharto, M.T., selaku Dekan Fakultas Teknologi informasi Universitas Darma Persada
5. Bapak Adam Arif Budiman, S.T., M. Kom., selaku Ketua Jurusan Teknologi informasi Univeritas Darma Persada dan Bagus Tri Mahardhika, S.T., M. Kom., selaku dosen pembimbing yang telah meluangkan waktu dan

pikirannya untuk memberikan bimbingan penyusunan Laporan Tugas Akhir ini.

6. Khususnya penulis ingin mengucapkan terima kasih sebesar-besarnya dan mempersembahkan Laporan Tugas Akhir ini kepada kedua orang tua penulis yang senantiasa selalu memberikan dukungan moril yang sangat berarti sehingga dapat terselesaikannya penyusunan Laporan Tugas Akhir ini.

Akhir kata semoga Laporan Tugas Akhir ini bermanfaat bagi kita semua.

Jakarta, 22 Februari 2021

Wahyu Edi Suryanto

ABSTRAK

Cabai rawit merupakan komoditas sayuran yang sangat penting bagi masyarakat Indonesia karena cabai rawit mengandung banyak vitamin dan zat gizi yang dibutuhkan untuk menjaga dan meningkatkan daya tahan tubuh. Selain itu, rasa pedas pada cabai rawit berkhasiat untuk menambah nafsu makan sehingga hal tersebut menjadi alasan banyak masyarakat Indonesia untuk mengkonsumsi cabai rawit. Mengingat kebutuhan masyarakat Indonesia akan cabai rawit terus meningkat membuat harga cabai rawit mengalami fluktuasi. Kenaikan harga yang fluktuatif menimbulkan kerugian pada beberapa kalangan, yaitu pedangan dan pengusaha kuliner yang menggunakan cabai rawit sebagai bahan utamanya. Untuk itu, dibutuhkan suatu metode untuk memprediksi harga cabai rawit di masa yang akan datang. Dengan melakukan prediksi harga cabai rawit diharapkan dapat membantu pedagang untuk mengendalikan jumlah permintaan cabai rawit dan pengusaha kuliner yang menggunakan cabai rawit sebagai bahan utama karena hasil prediksinya dapat digunakan untuk menekan modal pengeluaran sehingga para pengusaha bisa mendapatkan keuntungan yang maksimal. Data yang digunakan pada penelitian ini berupa data *time series* yang disusun berdasarkan waktu. Prediksi pada data *time series* memiliki tingkat kompleksitas yang tinggi sehingga dibutuhkan suatu metode yang dapat menghasilkan prediksi. Algoritme *Extreme Learning Machine* (ELM) dan *Support Vector Regression* (SVR) diusulkan peneliti karena dinilai mampu memberikan hasil prediksi dengan tingkat kesalahan yang rendah. Berdasarkan penelitian yang telah dilakukan, algoritme SVR menghasilkan nilai *Mean Average Percentage Error* (MAPE) sebesar 1,4919% dengan waktu komputasi sebesar 3,0919356 *seconds* sedangkan algoritme ELM menghasilkan nilai MAPE sebesar 3,6746% dengan waktu komutasi sebesar 0,0110832 *seconds*. Dari hasil tersebut dapat disimpulkan bahwa algoritme ELM unggul dalam proses pembelajaran sedangkan algoritme SVR unggul dalam akurasi tingkat kesalahan. Namun, kedua algoritme tersebut mampu menghasilkan nilai prediksi dengan tingkat kesalahan yang kurang dari 10%. Hal tersebut menunjukan bahwa nilai prediksi yang dihasilkan termasuk dalam kategori yang sangat baik.

Kata kunci: *prediksi harga cabai rawit, time series, Extreme Learning Machine, Support Vector Regression, Mean Average Percentage Error*

ABSTRACT

Cayenne pepper is a vegetable commodity that is very important for the people of Indonesia because cayenne pepper contains many vitamins and nutrients needed to maintain and increase endurance. In addition, the spicy taste of cayenne pepper is nutritious for increasing appetite so that it is the reason many Indonesians consume cayenne pepper. The needs of the Indonesian people for cayenne pepper continue to increase, causing the price of cayenne pepper to fluctuate. The fluctuating price increase caused losses to several groups, namely traders and culinary entrepreneurs who used cayenne pepper as the main ingredient. For that, we need a method to predict the price of cayenne pepper in the future. By predicting the price of cayenne pepper, it is hoped that it can help traders to control the amount of cayenne pepper demand and culinary entrepreneurs who use cayenne pepper as the main ingredient because the prediction results can be used to spend capital so that entrepreneurs get maximum profit. The data used in this study is time series data arranged based on time. Prediction on time series data has a high level of complexity, so we need a method that can produce predictions. Researchers proposed the Extreme Learning Machine (ELM) and Support Vector Regression (SVR) algorithm because it is able to provide predictions with low error rates. Based on the research that has been done, the SVR algorithm produces a Mean Average Percentage Error (MAPE) value of 1.4919% with a computation time of 3.0919356 seconds while the ELM algorithm produces a MAPE value of 3.6746% with a commutation time of 0.0110832 seconds. From these results it can be ignored that the ELM algorithm is superior in the learning process while the SVR algorithm is superior in the error rate. However, the second algorithm is able to produce predicted values with an error rate of less than 10%. This shows that the resulting predictive value is in the very good category.

Keywords: price prediction of cayenne pepper, time series, Extreme Learning Machine, Mean Average Percentage Error

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

TEKNOLOGI INFORMASI

UNIVERSITAS DARMA PERSADA

BAB 6

PENDAHULUAN

6.1 Latar belakang

Cabai rawit merupakan komoditas sayuran yang sangat penting karena sudah menjadi bahan utama yang digunakan masyarakat Indonesia pada setiap pengolahan makananannya. Kandungan zat *capsaicin* yang dimiliki cabai rawit membuat rasa pedas sekaligus menjadi penikmat makanan. Kandungan *antioksidan* yang dimiliki cabai rawit juga berfungsi untuk menghambat kerusakan pada sel tubuh akibat radikal bebas dan mencegah penyakit berbahaya seperti kanker dan tumor. Selain itu cabai rawit juga kaya akan vitamin C yang berfungsi untuk menjaga dan meningkatkan daya tahan tubuh manusia (Rosmaniar et al.., 2018). Rasa pedas dan kaya akan manfaat menjadi alasan masyarakat Indonesia untuk mengkonsumsi cabai rawit.

Kini cabai rawit bukan hanya digunakan untuk kebutuhan pangan saja akan tetapi banyak usaha kuliner yang menggunakan cabai rawit sebagai bahan utama pada menu makanannya. Hal ini membuat kebutuhan masyarakat Indonesia akan cabai rawit terus meningkat. Jumlah konsumi yang banyak jika tidak diimbangi dengan jumlah produksi akan membuat harga cabai rawit menjadi sangat tinggi. Beberapa faktor lain yang terjadi seperti musim panen, jumlah produksi, luas lahan panen dan lain-lain membuat harga cabai rawit mengalami fluktuasi. Kenaikan harga yang fluktuatif menimbulkan kerugian pada beberapa kalangan, yaitu ketika harga cabai rawit sedang tinggi para pedagang pasar akan mengalami kerugian

karena jumlah konsumen akan menurun sedangkan cabai rawit tidak dapat bertahan dalam waktu yang lama sehingga para pedagang pasar sulit untuk mendapatkan keuntungan. Kemudian selain para pedagang, pengusaha kuliner juga mengalami kerugian karena harga cabai rawit yang tinggi akan menguras jumlah pengeluaran modal yang harus dikeluarkan.

Berdasarkan beberapa permasalahan yang timbul akibat fluktuasi, dibutuhkan suatu metode untuk memprediksi harga cabai rawit. Prediksi merupakan proses memperkirakan sesuatu yang mungkin terjadi di masa depan. Prediksi tidak perlu memberikan jawaban secara pasti, melainkan berusaha untuk mencari jawaban sedekat mungkin (Herdianto, 2013). Dengan melakukan prediksi harga cabai rawit diharapkan dapat membantu pedagang untuk mengendalikan jumlah permintaan cabai rawit. Manfaat lainnya juga dapat dirasakan pada pengusaha kuliner yang menggunakan cabai rawit sebagai bahan utama karena hasil prediksinya dapat digunakan untuk menekan modal pengeluaran sehingga para pengusaha kuliner bisa mendapatkan keuntungan yang maksimal.

Data yang digunakan pada penelitian ini berupa data *time series* yang disusun berdasarkan waktu. Prediksi pada data *time series* memiliki tingkat kompleksitas yang tinggi sehingga dibutuhkan suatu metode yang dapat menghasilkan prediksi dengan menggunakan pola pergerakan data (Singh & Balasundaram, 2007). Algoritme *Extreme Learning Machine* (ELM) dan *Support Vector Regression* (SVR) diusulkan peneliti karena dinilai mampu memberikan hasil prediksi dengan tingkat kesalahan yang rendah. Algoritme SVR mampu mengatasi masalah *overfitting* (Furi, Jordi, & Saepudin, 2015) sedangkan algoritme ELM memiliki kemampuan untuk mempercepat proses pembelajaran dan menghasilkan prediksi

dengan akurasi yang cukup baik (Wang, et al., 2008). Hal tersebut telah dibuktikan pada penelitian yang membahas Perbandingan *Extreme Leaning Machine* dengan *Support Vector Regression* untuk Prediksi Permeabilitas Reservoir. Dari hasil penelitian yang telah dilakukan, algoritme SVR menghasilkan nilai *Root Mean Square Error* (RMSE) sebesar 0,4178 dengan kecepatan pembelajaran 3436,625 *seconds*, sedangkan algorime ELM menghasilkan nilai RMSE sebesar 0,4189 dengan kecepatan pembelajaran 2,2231 *seconds*. Berdasarkan hasil tersebut, algorime ELM memiliki keunggulan dalam proses pembelajaran sedangkan algoritme SVR unggul dalam akurasi tingkat kesalahan (Cheng, Cai & Pan, 2009). Selain itu, terdapat penelitian serupa yang menggunakan algoritme SVR dan algoritme ELM untuk melakukan prediksi harga cabai, yaitu Prediksi Harga Cabai Merah Menggunakan *Support Vector Regression* dan Prediksi Harga Cabai Rawit di Kota Malang Menggunakan Algoritme *Extreme Learning Machine*. Berdasarkan kedua penelitian tersebut, algoritme ELM mampu menghasilkan *Mean Absolute Percentage Error* (MAPE) sebesar 2,087% (Ariwanda, Cholissodin & Tibyani, 2019) sedangkan algorime SVR mampu menghasilkan MAPE sebesar 4,07% (Sepri, et al., 2020). Hasil MAPE yang kurang dari 10% menunjukan bahwa kedua algoritme tersebut dapat digunakan untuk prediksi (Ikhsan, Setiawan & Tibyani, 2019).

Berdasarkan beberapa penelitian yang telah dipaparkan, peneliti memutuskan untuk menggunakan algoritme ELM dan algoritme SVR untuk prediksi harga cabai rawit. Penelitian ini akan membandingkan kecepatan proses pembelajaran dan tingkat kesalahan yang dihasilkan pada masing-masing algoritme. Penelitian ini juga diharapkan dapat bermanfaat bagi masyarakat Indonesia khususnya bagi para

pedagang cabai dan pengusaha kuliner yang menggunakan cabai rawit sebagai bahan utamanya agar dapat meminimalisasi kerugian yang didapat akibat harga cabai yang fluktuatif.

6.2 Rumusan masalah

Berdasarkan latar belakang di atas, penulis merumuskan masalah sebagai berikut.

2. Berapa persentase tingkat kesalahan terendah yang dihasilkan Algoritme *Extreme Learning Machine* (ELM) dan *Support Vector Regression* (SVR) pada Prediksi Harga Cabai Rawit Merah Di Pasar Jatinegara Jakarta Timur menggunakan perhitungan *Mean Absolute Percentage Error* (MAPE).

6.3 Tujuan

Tujuan dari perancangan sistem ini adalah sebagai berikut.

3. Mengetahui persentase tingkat kesalahan terendah yang dihasilkan Algoritme *Extreme Learning Machine* (ELM) dan *Support Vector Regression* (SVR) pada Prediksi Harga Cabai Rawit Merah Di Pasar Jatinegara Jakarta Timur menggunakan perhitungan *Mean Absolute Percentage Error* (MAPE).

6.4 Manfaat

Manfaat yang didapat dari penelitian ini antara lain.

3. Menambah wawasan penulis dengan mengaplikasikan ilmu pengetahuan secara langsung.

4. Membantu pedagang dan pengusaha kuliner untuk meminimalisir kerugian dari dampak fluktuatif harga cabai rawit.

6.5 Metodologi Penelitian

Metode yang digunakan dalam penelitian ini terdiri dari langkah-langkah berikut.

4. Obeservasi

   Metode observasi atau pengamatan langsung merupakan teknik pengumpulan data dengan cara langsung melihat kegiatan transaksi jual beli untuk mendapatkan informasi yang detail dari harga cabai rawit merah yang selalu mengalami fluktuasi.

5. Wawancara

   Metode wawancara adalah metode pengumpulan data dengan cara melakukan tanya jawab langsung pada setiap narasumber yang berhubungan dengan topik yang dijadikan sebagai bahan penelitian.

6. Metode Literature

   Metode yang dilakukan dengan cara mencari dan membaca data yang bersumber dari website, jurnal ilmiah, skripsi dan referensi lain yang berkaitan dengan penelitian yang dilakukan.

6.6 Sistematika pembahasan

Untuk mempermudah dalam memahami lebih jelas tentang penulisan penelitian ini, maka penulis mengelompokan materi penulisan menjadi lima BAB yang secara garis besar ini dari setiap BAB tersebut saling berhubungan sebagai berikut.

BAB 1 PENDAHULUAN

Bab ini akan menguraikan tentang latar belakang, rumusan masalah, tujuan penelitian, manfaat penelitian, metodologi penulisan dan sistematika penulisan.

BAB 2 LANDASAN TEORI

Bab ini menjelaskan dasar teori sebagai referensi untuk digunakan pada penelitian, topik utama pada penelitian, arsitektur Jaringan Saraf Tiruan (JST), algoritme *Extreme Learning Machine* (ELM) dan *Support Vector Regression* (SVR), perhitungan Normalisasi dan Denormalisasi, Fungsi Aktivasi dan *Mean Absolute Percentage Error* (MAPE).

BAB 3 ANALISIS DAN RANCANGAN SISTEM

Bab ini akan membahas tahapan yang dilalui pada seluruh proses pembentukan penelitian. Tahapan tersebut dimulai dari studi kepustakaan, teknik pengumpulan data, peralatan pendukung, dan skenario pengujian. Bab ini juga membahas alur dari keseluruhan algoritme, perhitungan table secara manual, perancangan antarmuka, dan perancangan skenario pengujian.

BAB 4 IMPLEMENTASI HASIL

Bab ini akan membahas hasil dari implementasi antarmuka dari aplikasi yang telah dikembangkan. Bab ini juga membahas pengujian sesuai dengan perancangan skenario.

BAB 5 KESIMPULAN DAN SARAN

Bab ini berisi kesimpulan dari hasil penelitian, serta saran-saran yang diharapkan dapat bermanfaat bagi peneliti dan semua pihak-pihak yang terkait.

TEKNOLOGI INFORMASI

UNIVERSITAS DARMA PERSADA

BAB 7

LANDASAN TEORI

7.1 Landasan Teori

Bagian ini akan membahas penelitian yang relevan dari topik permasalahan yang sedang diteliti. Berikut ini, terdapat lima penelitian yang dijadikan sebagai referensi. Penelitian pertama membahas Perbandingan *Extreme Leaning Machine* dengan *Support Vector Regression* untuk Prediksi Permeabilitas Reservoir. Dari hasil penelitian yang telah dilakukan, algoritme SVR menghasilkan nilai *Root Mean Square Error* (RMSE) sebesar 0,4178 dengan kecepatan pembelajaran 3436,625 *seconds* sedangkan algorime ELM mampu menghasilkan nilai RMSE sebesar 0,4189 dengan kecepatan pembelajaran 2,2231 *seconds*. Berdasarkan hasil tersebut algorime ELM memiliki keunggulan dalam proses pembelajaran sedangkan algoritme SVR unggul dalam akurasi tingkat kesalahan (Cheng, Cai, & Pan, 2009).

Penelitian kedua membahas Prediksi Harga Cabai Rawit di Kota Malang Menggunakan Algoritme *Extreme Learning Machine*. Komoditas cabai rawit merupakan salah penyumbang inflasi terbesar di Indonesia. Untuk itu, diperlukan prediksi harga cabai rawit agar pemerintah dapat melakukan tindakan untuk mencengah permasalahan yang mungkin timbul akibat inflasi. Data yang digunakan pada penelitian ini merupakan data harga cabai rawit yang dikumpulkan mulai tanggal 1 Januari 2017 hingga 31 Desember 2018. Dari hasil penelitian yang telah dilakukan, algoritme ELM menghasilkan rata-rata nilai *Mean Absolute Percentage Error* (MAPE) sebesar 2,087% (Ariwanda, Cholissodin, & Tibyani, 2019).

Penelitian ketiga membahas Peramalan Harga Cabai Merah Besar Wilayah Jawa Timur Menggunakan Metode *Extreme Learning Machine*. Cabai merah besar memiliki tingkat konsumsi yang tinggi karena digunakan untuk bumbu dapur dan bahan masakan. Harga cabai merah besar yang tidak menentu mengakibatkan kerugian bagi negara dan masyarakat. Maka dari itu, diperlukan prediksi harga cabai merah besar untuk memperkirakan kenaikan harga di masa yang akan datang. Data yang digunakan pada penelitian ini merupakan data harga cabai merah besar yang dikumpulkan mulai tanggal 18 Juli 2016 hingga 28 Desember 2018. Dari hasil penelitian yang telah dilakukan, algoritme ELM menghasilkan rata-rata nilai MAPE sebesar 3% (Adiatmaja, Setiawan, & Wihandika, 2019).

Penelitian keempat membahas Prediksi Harga Cabai Merah Menggunakan Support Vector Regression. Cabai merupakan komoditas sayuran yang sangat penting bagi masyarakat Indonesia karena merupakan bahan utama dalam berbagai pengolahan makanan. Harga cabai yang tidak menentuk memicu timbulnya inflasi perekonomian. Untuk mengantisiapsi harga cabai dipasaran agara tetap dalam batas wajar diperlukan prediksi harga cabai. Data yang digunakan pada penelitian ini merupakan data harga cabai rawit yang dikumpulkan mulai tanggal 1 Januari 2017 hingga 31 Desember 2019. Dari hasil penelitian yang telah dilakukan, algoritme SVR menghasilkan nilai MAPE sebesar 4,07% (Sepri, et al., 2020).

Penelitian kelima membahas *Support Vector Regression* Untuk Peramalan Permintaan Darah: Studi Kasus Unit Transfusi Darah Cabang – PMI Kota Malang. PMI bertanggung jawab untuk memenuhi permintaan darah dari rumah sakit. Pengelola pusat penyimpanan darah memiliki tugas untuk mengelola jumlah pasokan darah. Kurangan atau lebihnya jumlah pasokan darah di tempat

penyimpaan seharusnya tidak terjadi, karena berdampak pada tingginya angka pasien yang meninggal. Untuk mengurangi kesalahan yang terjadi, diperlukan penelitian untuk melakukan prediksi permintaan darah. Dari hasil penelitian yang telah dilakukan, algoritme SVR menghasilkan nilai MAPE sebesar 3,899%. Hasil nilai MAPE yang kurang dari 10% dapat dikategorikan baik untuk hasil prediksi. (Rifqi, Setiawan, & Bactiar, 2018). Berikut merupakan daftar referensi penelitian yang ditunjukan pada Tabel 2.1.

Tabel 7.1 Daftar Landasan Pustaka

No.	Penelitian	Objek	Metode	Hasil
1.	(Cheng, Cai, & Pan, 2009).	Prediksi Permeabilitas Reservoir (Waduk minyak bumi).	*Extreme Learning Machine* dan *Support Vector Machine* dengan menggunakan perhitungan nilai *error Root Mean Square Error* (RMSE).	Algoritme *Support Vector Machine* menghasilkan nilai RMSE sebesar 0,4178 dengan kecepatan pembelajaran sebesar 3436,625s. Algoritme *Extreme Learning Machine* menghasilkan nilai RMSE sebesar 0,4189 dengan kecepatan pembelajaran sebesar 2,2231s.
2.	(Ariwanda, Cholissodin, & Tibyani).	Data harga cabai rawit di Kota Malang tanggal 1 Januari 2017 hingga 31 Desember 2018.	*Extreme Learning Machine* dengan menggunakan perhitungan nilai *error Mean Absolute Percentage Error* (MAPE).	Algoritme *Extreme Learning Machine* menghasilkan nilai MAPE sebesar 2,087%.
3.	(Adiatmaja, Setiawan, & Wihandika).	Data harga cabai rawit di Kota Malang tanggal 18 Juli 2016 hingga	*Extreme Learning Machine* dengan menggunakan perhitungan nilai *error Mean Absolute*	Algoritme *Extreme Learning Machine* menghasilkan nilai MAPE sebesar 3%.

		28 Desember 2018.	*Percentage Error* (MAPE).	
4.	(Sepri, et al., 2020).	Data harga cabai rawit di Kota Malang tanggal 1 Januari 2017 hingga 31 Desember 2019.	*Support Vector Regression* dengan menggunakan perhitungan nilai *error Mean Absolute Percentage Error* (MAPE).	Algoritme *Support Vector Regression* menghasilkan nilai MAPE sebesar 4,07%.
5.	(Rifqi, Setiawan, & Bactiar).	Data jumlah pasokan darah.	*Support Vector Regression* dengan menggunakan perhitungan nilai *error Mean Absolute Percentage Error* (MAPE).	Algoritme *Support Vector Regression* menghasilkan nilai MAPE sebesar 3,899%.

7.2 Prediksi

Prediksi merupakan proses memperkirakan sesuatu yang mengambarkan kondisi di masa yang akan datang berdasarkan data pada periode tertentu. Prediksi tidak perlu memberikan jawaban pasti, melainkan berusaha untuk mencari jawaban sedekat mungkin dengan apa yang akan terjadi (Herdianto, 2013). Prediksi diperlukan untuk memperkirakan kondisi yang akan datang sehingga hasil prediksi dapat dijadikan sebagai acuan dalam mengambil suatu keputusan serta perencanaan kedepannya.

Berdasarkan sifatnya, prediksi dibagi menjadi dua jenis, yaitu prediksi kualitatif dan prediksi kuantitatif. Prediksi kualitatif merupakan prediksi untuk data non-numerik sedangkan prediksi kuantitatif merupakan prediksi yang berdasarkan pada data numerik. Berdasarkan jangka waktu, prediksi dibagi menjadi tiga bagian, yaitu prediksi jangka pendek, prediksi jangka menengah, dan prediksi jangka panjang. Prediksi jangka pendek merupakan prediksi untuk rentang waktu kurang dari satu tahun. Kemudian prediksi jangka menengah merupakan prediksi untuk

rentang waktu tiga bulan sampai tiga tahun. Prediksi jangka panjang merupakan prediksi untuk rentang waktu lebih dari tiga tahun.

Karena penelitian ini menggunakan data numerik dan prediksinya hanya dilakukan untuk satu hari kedepan maka penelitian ini termasuk dalam jenis prediksi kuantitatif dan termasuk dalam prediksi jangka pendek.

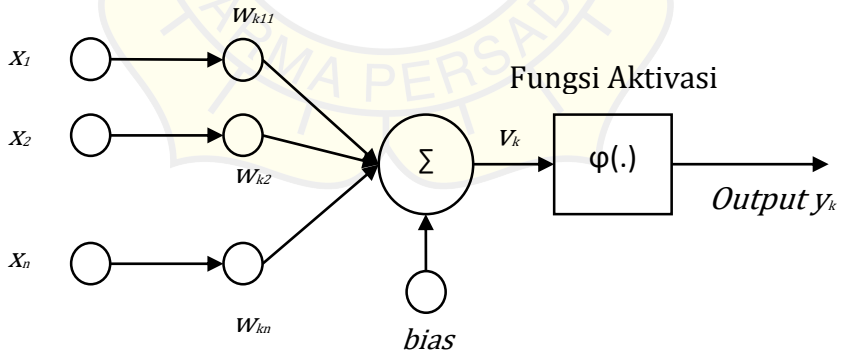
7.3 Cabai Rawit

Tanaman cabai rawit merupakan jenis tanaman dengan berbagai kadungan zat-zat gizi yang cukup lengkap, yaitu lemak, protein, karbohidrat, kalsium, fosfor, besi, vitamin A, B1, B2, C dan senyawa alkaloid seperti *capsaicin, flavanoid, oleoresin* dan minyak atsiri (Sujitno & Dianawati, 2015). Selain memiliki banyak kandungan zat-zat gizi, cabai rawit juga miliki manfaat untuk memberikan sensasi pedas pada makanan. Selain itu, buah tanaman ini juga berkhasiat untuk menambah nafsu makan (Tjandra, 2011). Cabai rawit merupakan komoditas sayuran yang sangat penting karena menjadi bahan utama yang digunakan masyarakat Indonesia. Rasa pedas dan kaya akan manfaat menjadi alasan utama untuk mengkonsumsi cabai rawit. Mengingat kebutuhan masyarakat Indonesia akan cabai rawit terus meningkat membuat harga cabai rawit mengalami fluktuasi. Kenaikan harga yang fluktuatif menimbulkan kerugian pada beberapa kalangan, yaitu pedangan dan pengusaha kuliner yang menggunakan cabai rawit sebagai bahan utamanya. Untuk itu, dibutuhkan suatu metode untuk memprediksi harga cabai rawit di masa yang akan datang. Dengan melakukan prediksi harga cabai rawit diharapkan dapat membantu pedagang untuk mengendalikan jumlah permintaan cabai rawit dan pengusaha kuliner yang menggunakan cabai rawit sebagai bahan utama karena hasil

prediksinya dapat digunakan untuk menekan modal pengeluaran sehingga para pengusaha bisa mendapatkan keuntungan yang maksimal.

7.4 Jaringan Saraf Tiruan

Jaringan Saraf Tiruan (JST) merupakan suatu teknik yang digunakan untuk membangun sebuah program cerdas dengan menggunakan pendekatan pemodelan yang menstimulasikan cara kerja otak manusia untuk menyelesaikan masalah melalui perhitungan komputer (Fikriya, Irawan, & Soetrisno, 2017). JST bekerja dengan memanfaatkan *neuron* untuk memberikan informasi dari satu *neuron* ke *neuron* lainnya. Bobot (*weight*) digunakan sebagai perantara untuk menyalurkan informasi tersebut. Setiap informasi yang disalurkan kepada *neuron* lainnya memiliki nilai tertentu. Nilai tersebut akan diproses dan disalurkan kembali hingga menghasilkan *output*. Berikut adalah struktur *neuron* pada jaringan saraf tiruan ditunjukan pada Gambar 2.2.



Gambar 7.1 Struktur Jaringan Saraf Tiruan

Sumber: (Fikriya, Irawan, & Soetrisno, 2017)

7.5 Fungsi Aktivasi

Fungsi aktivasi merupakan sebuah perhitungan untuk menentukan *output* berdasarkan *input neuron*. Fungsi aktivasi yang digunakan harus disesuaikan dengan bentuk datanya. Karena data yang digunakan pada penelitian ini merupakan data *time series*, maka fungsi aktivasi yang dapat digunakan adalah fungsi aktivasi *nonlinear*. Fungsi aktivasi ini mampu mengenali pola pengerakan data acak sehingga fungsi aktivasi ini dapat digunakan untuk prediksi data *time series*. Fungsi aktivasi *nonlinear* memiliki beberapa macam pemodelan. Pada penelitian ini fungsi aktivasi yang digunakan adalah fungsi aktivasi *sigmoid biner*. Berikut merupakan perhitungan dari fungsi aktivasi *sigmoid biner* ditunjukan pada Persamaan 2.1 (Srimuang & Intarasothonchun, 2015).

Fungsi *Sigmoid Biner*

$$H = \frac{1}{1+\exp\left(-H_{init}\right)} \tag{2.1}$$

Keterangan:

H = *Output* pada fungsi aktivasi

exp = Basis bilangan logaritma atau biasa disebut dengan bilangan *euler*, bilangan ini memiliki nilai pembulatan sebesar 2,71828183

H_{init} = *Input* pada fungsi aktivasi

7.6 Normalisasi Data

Normalisasi data merupakan suatu teknik untuk mentransformasikan data kedalam satuan yang sama. Normalisasi dilakukan untuk menghilangkan *outlier* pada data yang memiliki nilai yang jauh berbeda dengan nilai lainnya. Normalisasi juga digunakan untuk mempersiapkan data sebelum dihitung menggunakan pendekatan pemodelan. Pada dasarnya, normalisasi memiliki berbagai macam perhitungan. *Min-Max Normalization* digunakan peneliti karena memilki kesesuaian pada data yang digunakan. *Min-Max Normalization* bekerja dengan melakukan transformasi linier terhadap data asli. Untuk lebih lanjut, perhitungan *Min-Max Normalization* ditunjukan pada Persamaan 2.2 (Mustaffa & Yusof, 2011).

$$X_n = \frac{(X_0 - X_{min})}{(X_{max} - X_{min})} \quad (2.2)$$

Keterangan:

X_n = Nilai hasil normalisasi

X_0 = Nilai yang akan dinormalisasi

X_{max} = Nilai maksimum dari keseluruhan data

X_{min} = Nilai minimum dari keseluruhan data

7.7 Denormalisasi Data

Denormalisasi data merupakan proses pengembalian data normalisasi menjadi data yang sebenarnya. Denormalisasi data dilakukan ketika suatu pendekatan pemodelan data telah memberikan hasil akhirnya. Perhitungan denormalisasi disesuaikan dengan perhitungan normalisasi yang digunakan. Berikut adalah

perhitungan denormalisasi data ditunjukan pada Persamaan 2.3 (Mustaffa & Yusof, 2011).

$$X_0 = X_n.(X_{max} - X_{min}) + X_{min} \quad (2.3)$$

Keterangan:

X_0 = Nilai hasil denormalisasi

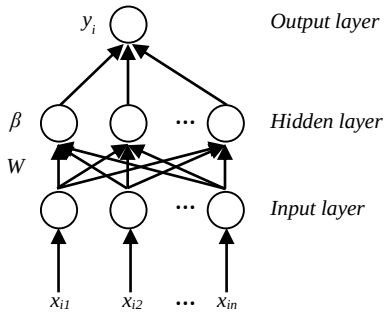
X_n = Nilai yang akan didenormalisasi

X_{max} = Nilai maksimum dari keseluruhan data

X_{min} = Nilai minimum dari keseluruhan data

7.8 *Extreme Learning Machine* (ELM)

Algoritme *Extreme Learning Machine* (ELM) pertama kali diperkenalkan oleh Guang-Bin Huang, dkk pada tahun 2006. Algoritme ELM merupakan bagian dari JST yang dibuat dengan tujuan untuk mengatasi kelemahan dari algoritme JST lainnya terutama pada proses kecepatan pembelajaran (Huang et al, 2004). Algoritme ini sendiri memiliki kemampuan pembelajaran ribuan kali lebih cepat daripada algorime lainnya dan mampu menghasilkan generalisasi yang sangat baik (Wang, et al., 2008). Arsitektur yang digunakan pada algoritme ELM merupakan arsitektur *Single Layer Feedforward Network* (SLFN). Arsitektur SLFN tidak memiliki iterasi didalamnya, karena jaringan *node* yang terhubung pada setiap tidak membentuk *directed cycle/graph*. Arsitektur SLFN sendiri memiliki tiga layer, yaitu input layer, hidden layer dan output layer. Untuk mengetahui lebih lanjut, arsitektur SLFN pada algoritme ELM dapat dilihat pada Gambar 2.3.



Gambar 7.2 Arsitektur Algoritme *Extreme Learning Machine*

Sumber: (Abadi, Musyafa & Soeprijanto, 2014)

Algoritme ELM mempunyai dua proses, yaitu proses pembelajaran atau proses *training* dan proses pengujian atau proses *testing*, kedua proses tersebut saling berkaitan satu sama lain. Untuk lebih lanjut, tahapan yang dilalui pada algoritme ELM dapat dijelaskan sebagai berikut (Fachrony, Cholissodin & Santoso, 2018).

7.8.1 Proses *Training*

Proses *training* pada algoritme ELM merupakan proses memberikan pembelajaran pada sistem untuk mengenali pola data. Berikut adalah proses training pada algoritme ELM dengan menggunakan *bias*.

6. Inisialisasi bobot W_{jk} dan bias *b* secara acak pada rentang tertentu.

   Inisialisasi nilai bobot berkisar antara [0, 1], sedangkan nilai *bias* berkisar antara [-1,1]. Besar ukuran matriks bobot W_{jk} ditentukan pada jumlah fitur (k) dan *hidden neuron* (j). Kemudian, besar ukuran pada matriks bias *b* hanya disesuaikan dengan jumlah *hidden neuron* yang digunakan.

7. Menghitung nilai H_{init}

Proses perhitungan nilai H_{init} digunakan untuk menghitung nilai *hidden layer*. Berikut adalah rumus perhitungan nilai H_{init} ditunjukan pada Persamaan 2.4.

$$H_{init} = X.W^T + bias \quad (2.4)$$

Keterangan:

H_{init} = Matriks H_{init}

x = Matriks data *training*

W^T = Matriks *transpose* bobot

8. Menghitung *hidden layer* (H)

Pada perhitungan nilai *hidden layer*, jenis fungsi aktivasi yang digunakan adalah fungsi *sigmoid biner*. Berikut merupakan proses perhitungan *hidden layer* yang ditunjukan pada Persamaan 2.5.

$$H = \frac{1}{1+\exp(-H_{init})} \quad (2.5)$$

Keterangan:

H = Matriks *Output hidden layer*

exp = Bilangan *euler* dengan nilai pembulatan sebesar 2,71828183

9. Menghitung H *dagger* ($H^\dagger$)

Pada perhitungan H *dagger* atau *Moore-Penrose pseudo inverse* membutuhkan proses perhitungan *inverse* matriks. Berikut merupakan proses perhitungan H *dagger* ditunjukan pada Persamaan 2.6.

$$H^\dagger = (H^T.H)^{-1}.H^T \quad (2.6)$$

Keterangan:

$H^{\dagger}$ = Matriks H *dagger*

H^{T} = Matriks H *transpose*

$(H^{T}.H)^{-1}$ = Matriks *inverse* dari perkalian H *transpose* dengan H

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

Proses perhitungan *output weight* merupakan tahap terakhir pada proses *training*. Nilai yang dihasilkan pada perhitungan ini akan diteruskan pada proses *testing* untuk mendapatkan hasil prediksi. Proses perhitungan *output weight* ditunjukan pada Persamaan 2.7.

$$\hat{\beta} = H^{\dagger}.Y \tag{2.7}$$

Keterangan:

$\hat{\beta}$ = Matriks *ouput weight*

$H^{\dagger}$ = Matriks H *dagger*

Y = Matriks data target

7.8.2 Proses *Testing*

Proses *testing* pada algoritme ELM merupakan tahap yang dilakukan untuk menggeneralisasi data baru. Proses *testing* pada algoritme ELM dapat diuraikan sebagai berikut.

4. Menghitung H_{init}

Proses perhitungan H_{init} disesuaikan pada Persamaan 2.4. Data yang digunakan untuk menghitung H_{init} adalah data *testing*.

5. Menghitung *hidden layer* (H)

Matriks H_{init} yang telah dihitung akan digunakan pada proses perhitungan *hidden layer*. Proses perhitungan *hidden layer* pada proses *testing* disesuaikan dengan Persamaan 2.5.

6. Menghitung *Output Layer* ($\hat{Y}$)

Proses perhitungan *output layer* merupakan proses terakhir pada algortime ELM. Proses perhitungan ini membutuhkan matriks *hidden layer* dan matriks *output weight* yang dihasilkan dari proses *training*. Hasil perhitungan *output layer* perlu didenormalisasi untuk mengembalikan bentuk data yang sebenarnya sehingga hasil tersebut dapat dilakukan pengujian untuk mengukur tingkat kesalahannya. Proses untuk menghitung *output layer* ditunjukan pada Persamaan 2.8.

$$\hat{Y} = H.\hat{\beta} \tag{2.8}$$

Keterangan:

$\hat{Y}$ = Matriks *output layer*

H = Matriks *hidden layer*

$\hat{\beta}$ = Matriks *ouput weight*

7.9 *Support Vector Regression* (SVR)

Support Vector Regression (SVR) merupakan model pengembangan dari *Support Vector Machine* (SVM) yang digunakan untuk menyelesaikan kasus prediksi pada data *time series*. Algoritme SVR bekerja dengan mencari garis pemisah atau *hyperplane* terbaik dengan cara mengukur jarak antara garis pemisah

dengan data terdekat. Tujuan algoritme SVR dibuat adalah untuk mengatasi permasalahan *overfitting* yang terjadi pada model pendekatan regresi lainnya. Algoritme ini sendiri memiliki kelebihan dalam memanfaatkan fungsi kernel untuk memetakan vektor input ke ruang fitur berdimensi tinggi sehingga algoritme SVR dapat mengatasi *overfitting* dan mampu menghasilkan performa generalisasi yang lebih baik (Maharesi, 2013). Untuk lebih lanjut, alur tahapan yang dilalui algoritme SVR dapat diuraikan sebagai berikut (Vijayakumar & Wu, 1999).

6. Inisialisasi parameter *sigma* (σ), *lambda* (λ), *epsilon* (ε), *coefisien Learning Rate* (cLR), kompleksitas (C) dan jumlah iterasi maksimum.

7. Inisialisasi nilai *Lagrange Multiplier* α_i = 0 dan α_i^* = 0 kemudian hitung matriks *Hessian* menggunakan fungsi kernel *Gaussian RBF* pada Persamaan 2.9.

$$R_{ij} = (K(x_i, x_j) + \lambda^2)\ untuk\ i, j = 1, \dots, n \tag{2.9}$$

Keterangan:

R_{ij} = Matriks *Hessian*

x_i = Data ke-i

x_j = Data ke-j

$K(x_i, x_j)$ = Fungsi kernel *Gaussian RBF*

λ = Variable skalar

8. Masuk pada iterasi pertama sampai batas iterasi yang ditentukan, untuk setiap data ke-*i* sampai ke-*n* lakukan perhitungan sebagai berikut.

   a. Nilai *error* ke-*i*.

$$E_i = y_i - \sum_{j=1}^{n} (\alpha_i^* - \alpha_i) R_{ij} \tag{2.10}$$

b. Hitung perubahan nilai *Lagrange Multiplier*

$$\delta\alpha_i^* = min\{\max[\Upsilon(E_i - \varepsilon), -\delta\alpha_i^*], C - \delta\alpha_i^*\} \quad (2.11)$$

$$\delta\alpha_i = min\{\max[\Upsilon(E_i - \varepsilon), -\delta\alpha_i], C - \delta\alpha_i\} \quad (2.12)$$

c. Menambahkan nilai *Lagrange Multiplier* dengan hasil perubahannya

$$\delta\alpha_i^* = \alpha_i^* + \delta\alpha_i^* \quad (2.13)$$

$$\delta\alpha_i = \alpha_i + \delta\alpha_i \quad (2.14)$$

Keterangan:

E_i = Nilai *Error* ke-*i*

y_i = Nilai data target

$\alpha i*$ = *Lagrange Multiplier*

αi = *Lagrange Multiplier*

Rij = Matriks *Hessian*

$\delta\alpha_i^*$ = Perubahan nilai α_i^*

$\delta\alpha_i$ = Perubahan nilai α_i

Υ = Nilai *learning rate* atau *gamma*

ε = Nilai *epsilon*

C = Nilai Kompleksitas

9. Kembali mengulang langkah ketiga sampai batas iterasi maksimum yang telah ditentukan atau sampai dalam kondisi max ($|\delta\alpha_i^*|$) < ε dan max ($|\delta\alpha_i|$) < ε.

10. Menghitung fungsi regresi

Proses perhitungan fungsi regresi merupakan tahap terakhir pada algoritme SVR. Perhitungan ini digunakan untuk menghasilkan nilai prediksi. Proses perhitungan fungsi regresi ditunjukan pada Persamaan 2.15.

$$f(x) = \sum_{j=1}^{n}(\alpha_i^* - \alpha_i)(K(x_i, x_j) + \lambda^2) \quad (2.15)$$

Keterangan:

$\delta\alpha_i^*$ = *Lagrange Multiplier*

$\delta\alpha_i$ = *Lagrange Multiplier*

x_i = Data ke-i

x_j = Data ke-j

$K(x_i, x_j)$ = Fungsi kernel *Gaussian RBF*

λ = Variable skalar

7.10 *Mean Absolute Percentage Error* (MAPE)

MAPE merupakan suatu model perhitungan yang bertujuan untuk menentukan tingkat kesalahan berdasarkan selisih antara nilai hasil prediksi dengan data target. Perhitungan MAPE ditunjukan pada Persamaan 2.16 (Andini & Auristandi, 2016).

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{\hat{y}_i - y_i}{y_i}\right| \; x \; 100\,\% \quad (2.16)$$

Keterangan:

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

y_i = Nilai data target

n = Jumlah data *testing*

Penilaian kinerja sebuah model pembelajaran pada perhitungan nilai MAPE terbagi atas empat kategori. Berikut adalah kriteria penilaian MAPE ditunjukan pada Tabel 2.2 (Rahmadiani & Anggraeni, 2012).

Tabel 7.2 Kriteria Nilai MAPE

Nilai MAPE	Status
< 10%	Sangat baik
10 – 20%	Baik
20 – 50%	Cukup
> 50%	Buruk

TEKNOLOGI INFORMASI

UNIVERSITAS DARMA PERSADA

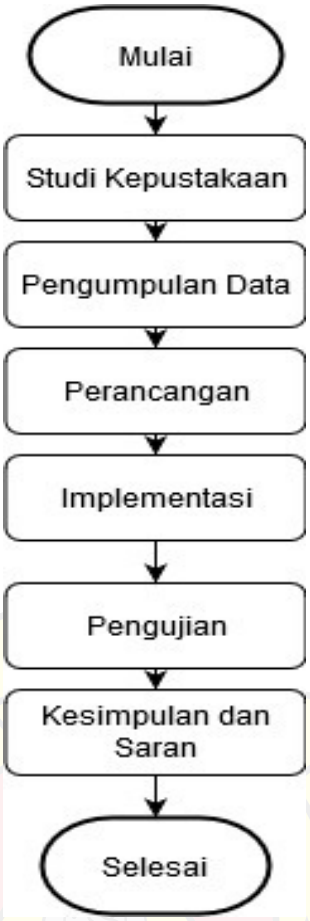
BAB 8

ANALISIS DAN RANCANGAN SISTEM

8.1 Metodologi Penelitian

8.1.1 Strategi Penelitian

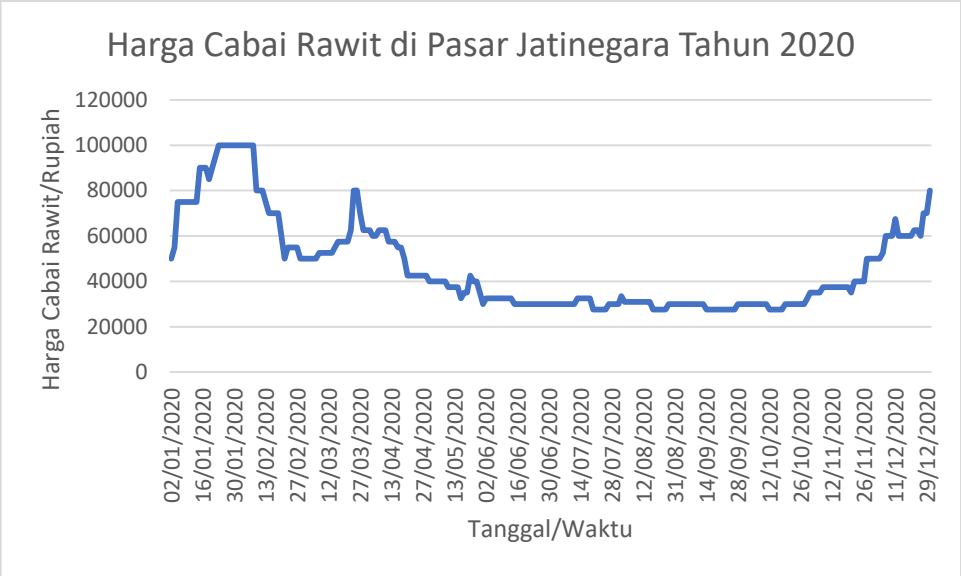
Strategi penelitian dirancang untuk menentukan tahapan yang harus dilalui pada proses penelitian sehingga penelitian dapat berjalan sesuai dengan konteks pemasalahan yang dibahas. Penelitian ini membahas Perbandingan Algoritme *Extreme Learning Machine* (ELM) dengan *Support Vector Regression* (SVR) pada Prediksi Harga Cabai Rawit di Pasar Jatinegara, Jakarta Timur. Berdasarkan penelitian tersebut, berikut adalah proses yang ditempuh pada penelitian, yaitu studi kepustakaan, pengumpulan data, perancangan, implementasi, pengujian, kesimpulan dan saran. Studi kepustakaan digunakan untuk mengumpulkan informasi-informasi atau data-data yang relevan untuk dijadikan sebagai bahan referensi. Pengumpulan data dilakukan untuk mendapatkan data yang akan digunakan sebagai bahan untuk melakukan prediksi. Tahap perancangan merupakan tahap untuk mengatur implementasi yang akan dibentuk dan skenario pengujian berdasarkan algoritme yang digunakan. Implementasi dan pengujian merupakan penerapan dari perancangan yang telah ditentukan. Setelah tahap pengujian, tahap terkahir yang harus dilakukan pada penelitian adalah memberikan kesimpulan berdasarkan hasil pengujian dan saran sebagai masukan bagi peneliti untuk mengembangkan penelitiannya. Bentuk sederhana strategi penelitian digambarkan pada Gambar 3.1.



Gambar 8.1 Diagram Alir Metodologi Penelitian

8.1.2 Data Penelitian

Data yang digunakan pada penelitian ini diambil dari website https://hargapangan.id/tabel-harga/pasar-tradisional/daerah. Dari website tersebut didapatkan data harga cabai rawit merah di jakarta timur lokasi pasar jatinegara yang dikumpulkan mulai tanggal 1 Januari 2020 sampai 31 Desember 2020. Data yang berhasil dikumpulkan berjumlah 242 data. Karena data tersebut disusun berdasarkan waktu, maka dapat tentukan bahwa data tersebut bersifat *time series*. Berikut adalah grafik data harga cabai rawit ditunjukan pada Gambar 3.2.



Gambar 8.2 Grafik Data Harga Cabai Rawit

8.1.3 Implementasi Algoritme

Implementasi merupakan penerapan dari hasil perancangan yang telah dirancang. Agar perancangan dapat diimplementasikan dengan baik dibutuhkan beberapa peralatan pendukung, yaitu berupa perangkat keras (*hardware*) dan perangkat lunak (*software*).

3. Perangkat Keras

   Berikut perangkat keras yang digunakan peneliti untuk implementasi sistem.

   - Laptop Lenovo AMD FX-7500 APU with AMD Radeon R7 Graphics
   - Memori RAM 4096MB

4. Perangkat Lunak

   Berikut perangkat lunak yang digunakan peneliti untuk implemetasi sistem.

   - Sistem Operasi Windows 10 Education 64-bit

- Visual Studio 2017
- Google Chrome *version* 71.0.3578.98 64-bit
- Microsoft Office 2019

8.1.4 Pengujian Algoritme

Untuk mengetahui keberhasilkan sebuah algoritme diperlukan perhitungan untuk menguji tingkat kesalahan yang dihasilkan algoritme tersebut. Perhitungan nilai *Mean Averege Percentage Error* (MAPE) digunakan peneliti untuk menguji seberapa baik algoritme *Extreme Learning Machine* (ELM) dan *Support Vector Regression* (SVR) dalam prediksi harga cabai rawit di pasar jatinegara.

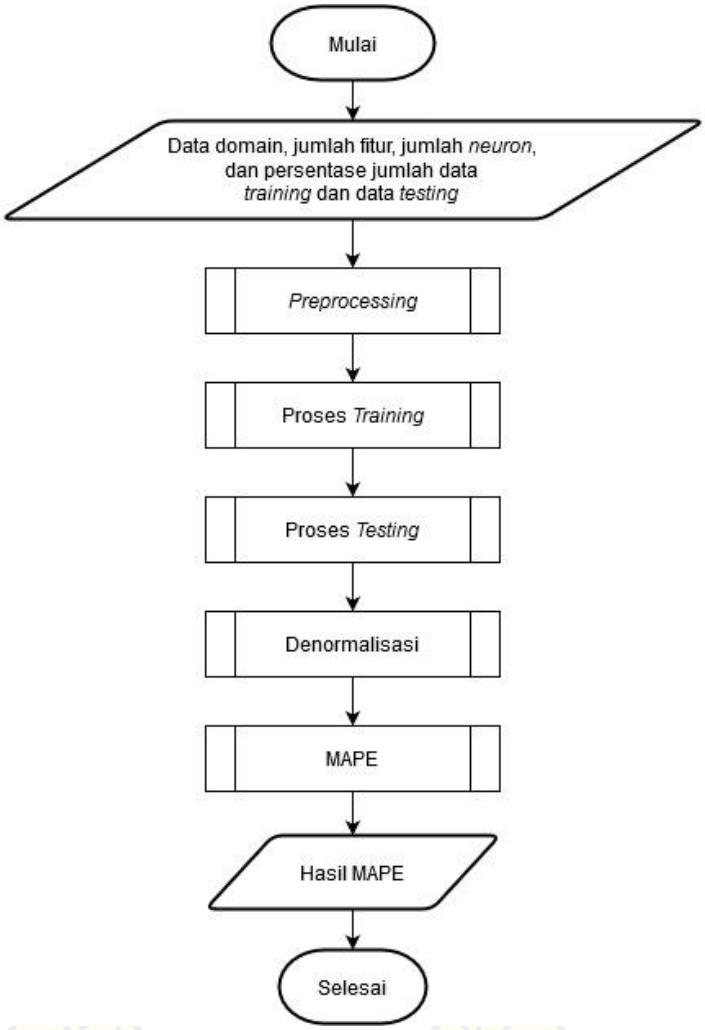
8.1.5 Penarikan Kesimpulan dan Saran

Kesimpulan diambil dari hasil pengujian yang telah dilakukan. Penarikan kesimpulan ditunjukan untuk menjawab rumusan masalah yang sebelumnya telah diuraikan. Saran pada penelitian merupakan masukan bagi peneliti untuk mengembangkan atau menyempurnakan penelitian berdasarkan kekurangan yang terdapat pada penelitian.

8.2 Perancangan Sistem

8.2.1 Alir Perancangan Algoritme ELM

Alir perancangan algoritme ELM merupakan rancangan mengenai seluruh tahapan atau proses yang harus dilalui algroitme. Berikut adalah alur perancangan dari algoritme ELM ditunjukan pada Gambar 3.3.



Gambar 8.3 Diagram Alir Algoritme ELM

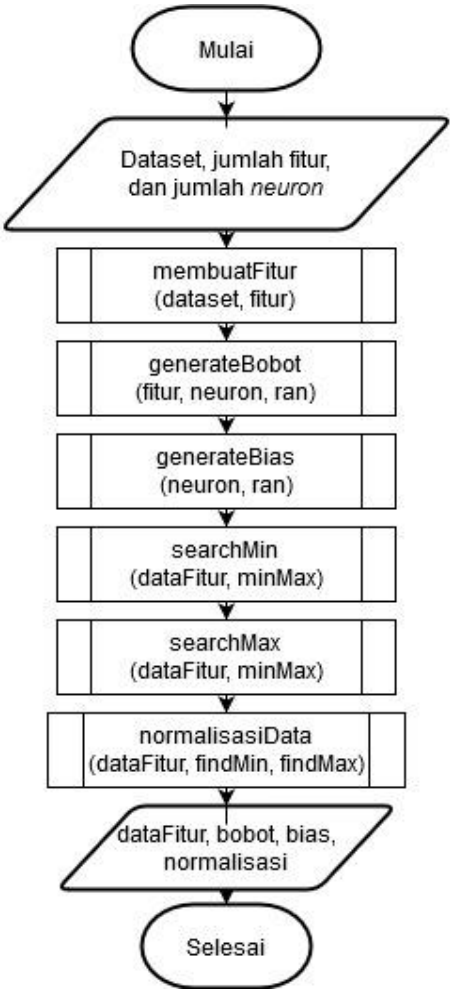
Penjelasan dari diagram alir algoritme ELM pada Gambar 3.3 untuk setiap langkah-langkahnya adalah sebagai berikut.

1. Menentukan nilai parameter pada algoritme ELM.
2. Melakukan tahap *preprocessing* yang meliputi pembuatan fitur, pembuatan bobot dan bias, dan menghitung normalisasi data.
3. Melakukan proses *training* untuk memberikan pembelajaran pada sistem.
4. Melakukan proses *testing* untuk menghasilkan nilai prediksi.

5. Melakukan perhitungan denormalisasi data untuk mengembalikan nilai hasil prediksi kebentuk yang sebenarnya.

6. Melakukan perhitungan MAPE untuk mengukur tingkat kesalahan yang nilai prediksi yang dihasilkan pada algoritme ELM.

8.2.1.2 Alir *Preprocessing*



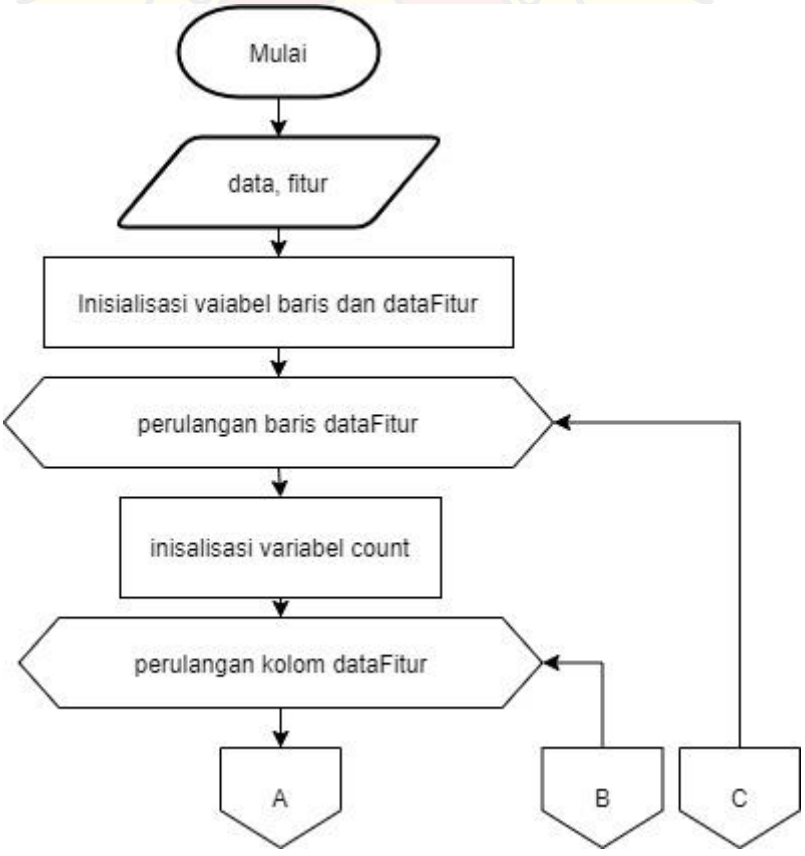
Gambar 8.4 Diagram Alir *Preprocessing*

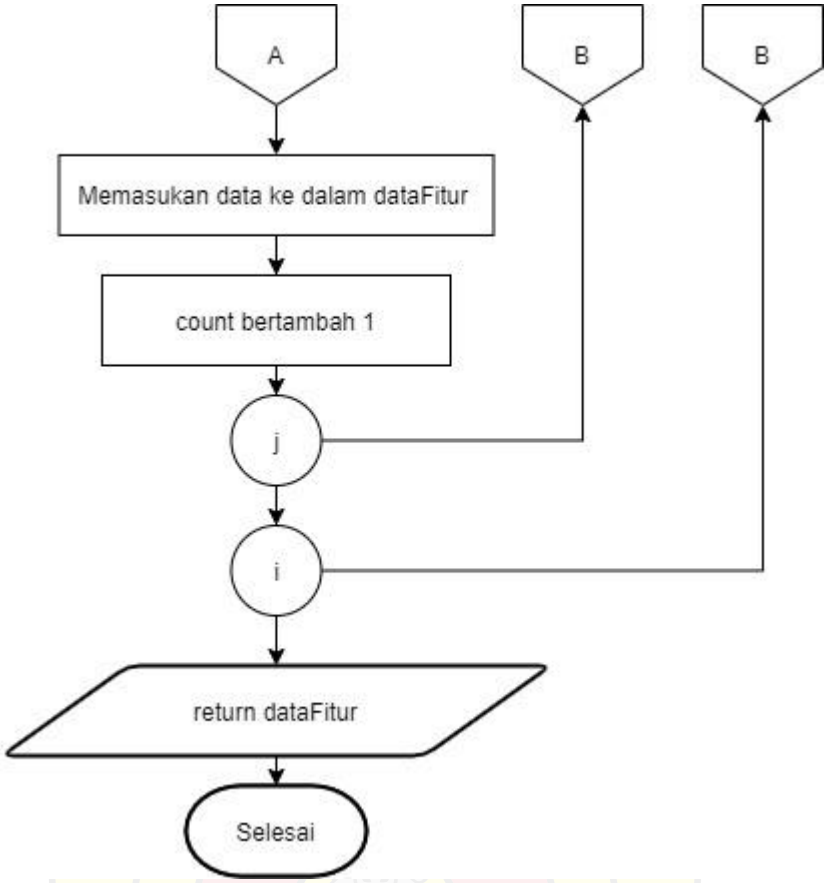
Penjelasan dari diagram alir *preprocessing* pada Gambar 3.4 untuk setiap langkah-langkahnya adalah sebagai berikut.

9. Memasukan parameter dataset, jumlah fitur, jumlah *neuron*.

10. Melakukan proses untuk membuat data fitur.
11. Melakukan proses untuk mengeneralisasi nilai bobot.
12. Melakukan proses untuk mengeneralisasi nilai bias.
13. Melakukan proses untuk mencari nilai minimum pada data fitur.
14. Melakukan proses untuk mencari nilai maksimum pada data fitur.
15. Melakukan proses untuk menghitung normalisasi data.
16. Mengembalikan hasil pembuatan data fitur, bobot, bias, dan nilai hasil perhitungan normalisasi.

(a) Alir Membuat Fitur





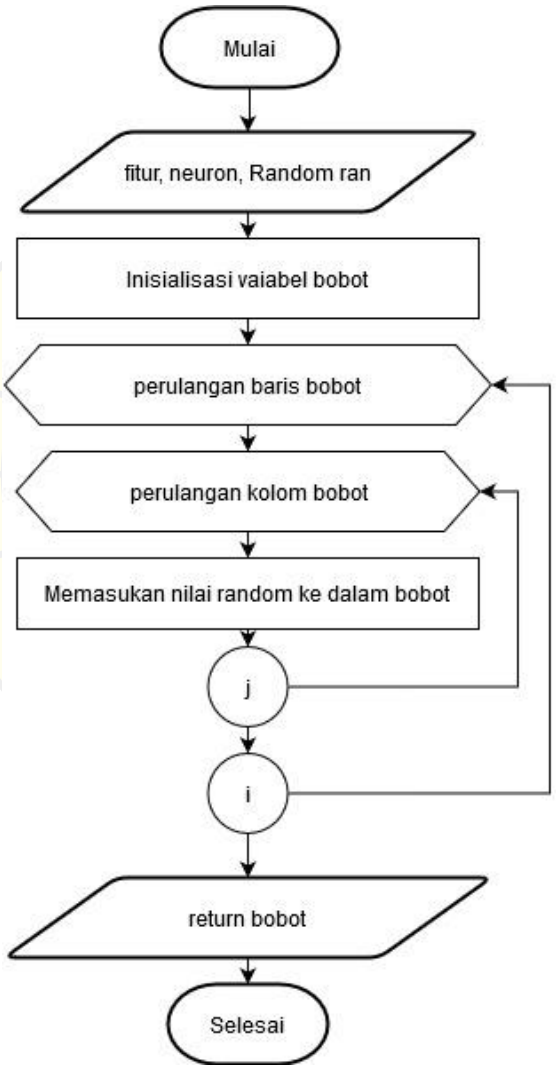
Gambar 8.5 Diagram Alir Membuat Fitur

Penjelasan dari diagram alir membuat fitur pada Gambar 3.5 untuk setiap langkah-langkahnya adalah sebagai berikut.

9. Memasukan parameter dataset dan jumlah fitur.
10. Membuat variable baris untuk menyimpan jumlah baris pada dataset dan array dataFitur untuk menyimpan data fitur
11. Melakukan perulangan i sesuai dengan panjang baris pada dataFitur.
12. Membuat variable countStart sebagai indeks untuk memasukan dataset kedalam dataFitur.
13. Melakukan perulangan j sesuai dengan panjang kolom pada dataFitur.

14. Proses memasukan dataset kedalam dataFitur untuk setiap indeks pada dataFitur.
15. Proses menambahkan nilai satu pada variabel countStart.
16. Mengembalikan hasil pembuatan dataFitur.

(b) Alir Membuat Bobot

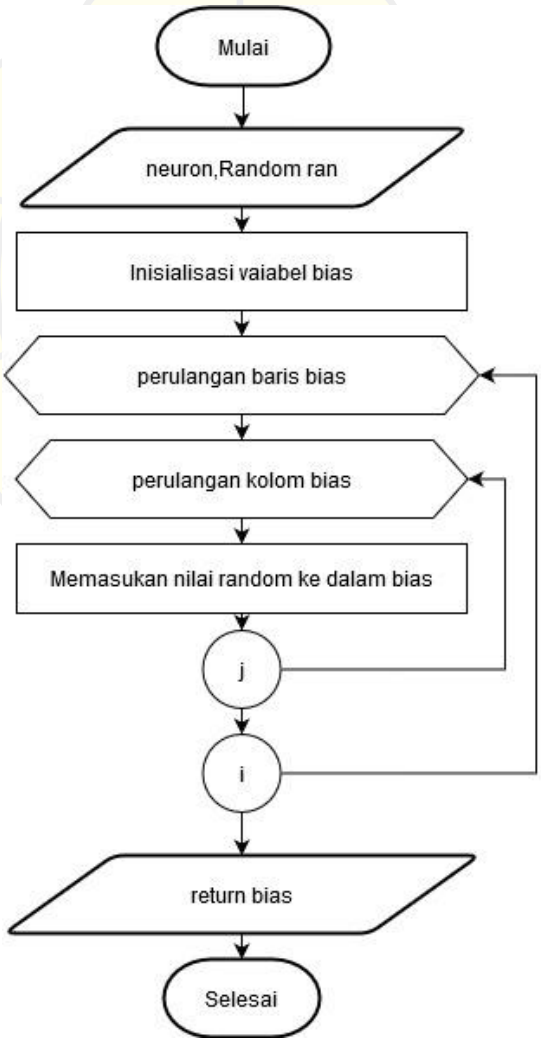


Gambar 8.6 Diagram Alir Membuat Bobot

Penjelasan dari diagram alir membuat bobot pada Gambar 3.6 untuk setiap langkah-langkahnya adalah sebagai berikut.

7. Memasukan parameter jumlah fitur, jumlah *neuron*, dan nilai random.
8. Membuat array bobot untuk menyimpan nilai bobot.
9. Melakukan perulangan i sesuai dengan panjang baris pada bobot.
10. Melakukan perulangan j sesuai dengan panjang kolom pada bobot.
11. Proses memasukan nilai random kedalam bobot.
12. Mengembalikan hasil pembuatan nilai bobot.

(c) Alir Membuat Bias

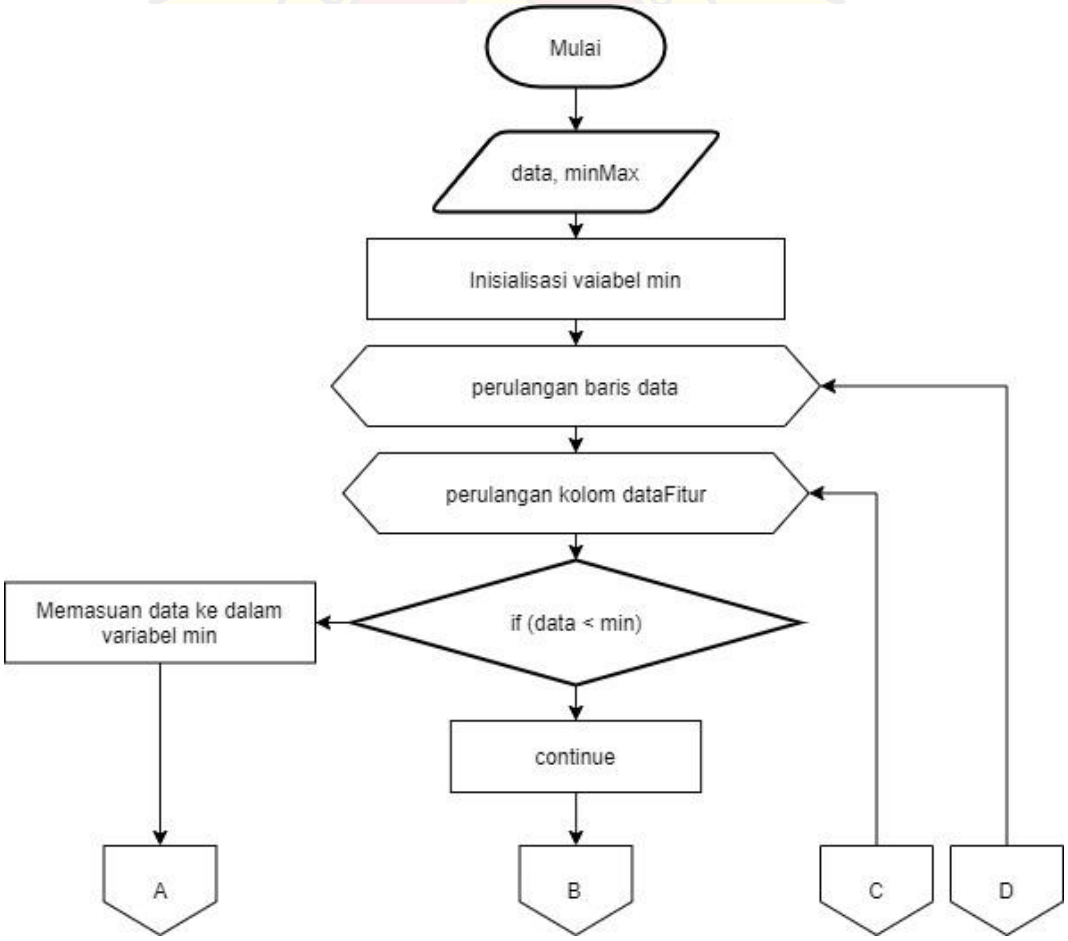


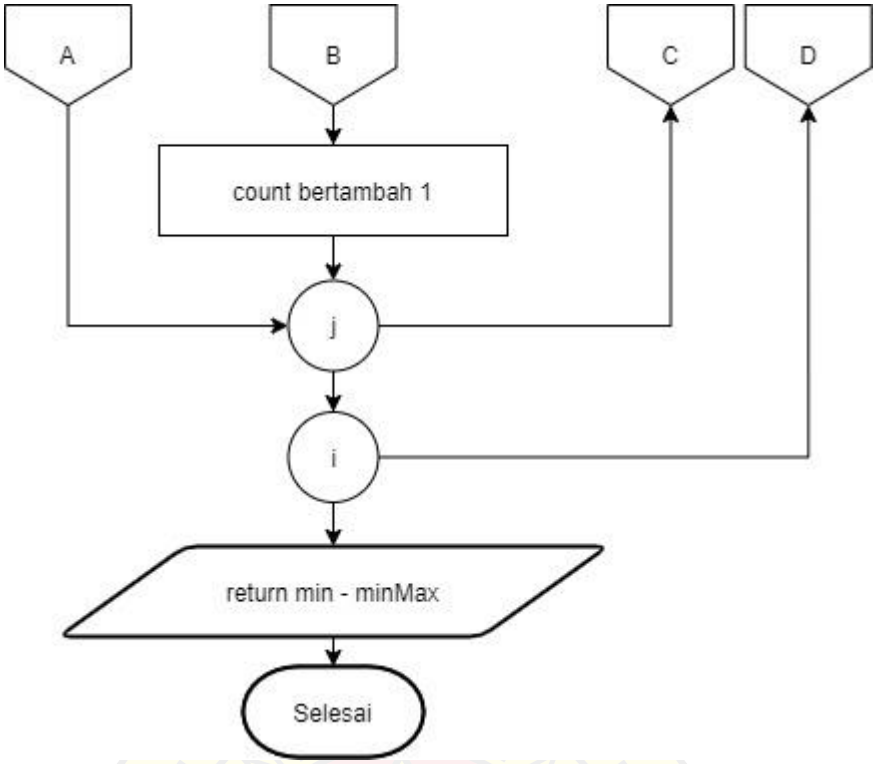
Gambar 8.7 Diagram Alir Membuat Bias

Penjelasan dari diagram alir membuat bias pada Gambar 3.7 untuk setiap langkah-langkahnya adalah sebagai berikut.

7. Memasukan parameter jumlah *neuron*, dan nilai random.
8. Membuat array bias untuk menyimpan nilai bias.
9. Melakukan perulangan i sesuai dengan panjang baris pada bias.
10. Melakukan perulangan j sesuai dengan panjang kolom pada bias.
11. Proses memasukan nilai random kedalam bias.
12. Mengembalikan hasil pembuatan nilai bias.

(d) Alir Mencari Nilai Minimum



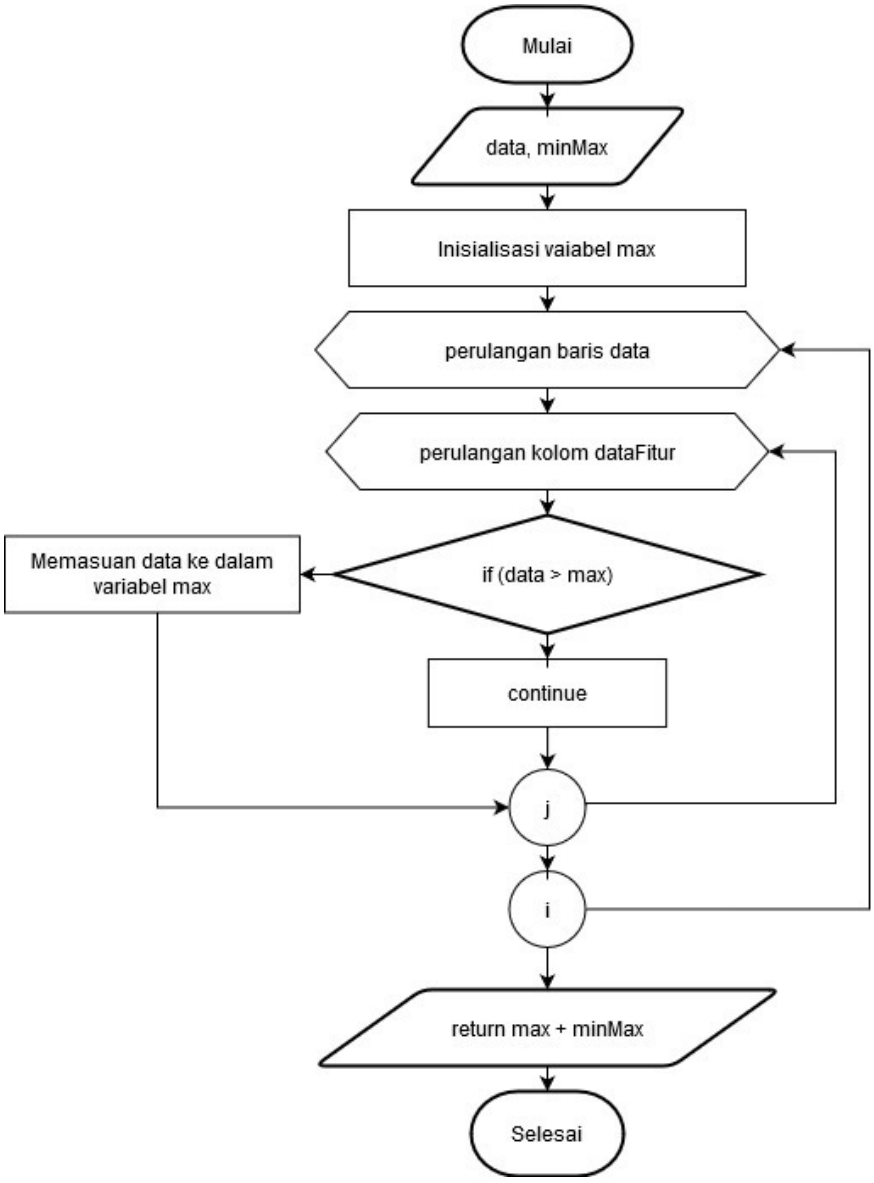


Gambar 8.8 Diagram Alir Mencari Nilai Minimum

Penjelasan dari diagram alir mencari nilai minimum pada Gambar 3.8 untuk setiap langkah-langkahnya adalah sebagai berikut.

9. Memasukan parameter dataFitur dan nilai minMax.
10. Membuat variabel min untuk menyimpan nilai minimum dari dataFitur.
11. Melakukan perulangan i sesuai dengan panjang baris pada dataFitur.
12. Melakukan perulangan j sesuai dengan panjang kolom pada dataFitur.
13. Melakukan seleksi if sebagai syarat untuk melakukan proses selanjutnya.
14. Memasukan nilai minimum pada dataFitur kedalam variabel min jika syarat pada proses seleksi if terpenuhi.
15. Melakukan proses continue untuk melanjukan perulangan jika syarat pada proses seleksi if tidak terpenuhi.
16. Mengembalikan nilai minimum.

(e) Alir Mencari Nilai Maksimum



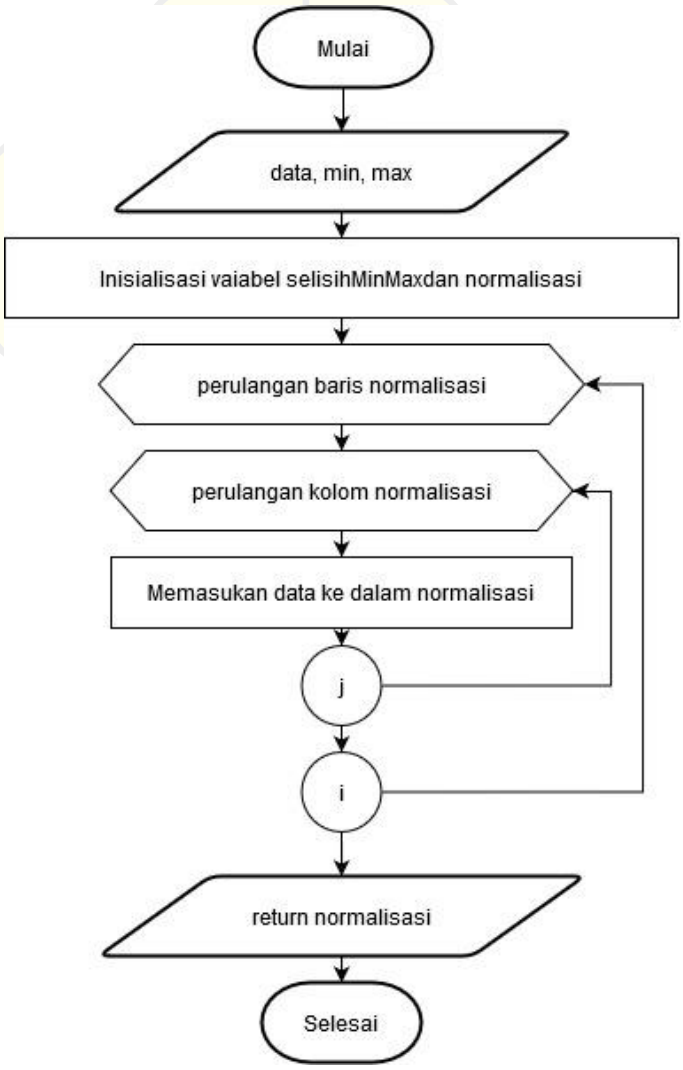
Gambar 8.9 Diagram Alir Mencari Nilai Maksimum

Penjelasan dari diagram alir mencari nilai maksimum pada Gambar 8.9 untuk setiap langkah-langkahnya adalah sebagai berikut.

9. Memasukan parameter dataFitur dan nilai minMax.
10. Membuat variabel max untuk menyimpan nilai maksimum dari dataFitur.
11. Melakukan perulangan i sesuai dengan panjang baris pada dataFitur.

12. Melakukan perulangan j sesuai dengan panjang kolom pada dataFitur.
13. Melakukan seleksi if sebagai syarat untuk melakukan proses selanjutnya.
14. Memasukan nilai maksimum pada dataFitur kedalam variabel max jika syarat pada proses seleksi if terpenuhi.
15. Melakukan proses continue untuk melanjukan perulangan jika syarat pada proses seleksi if tidak terpenuhi.
16. Mengembalikan nilai maksimum.

(f) Alir Normalisasi Data

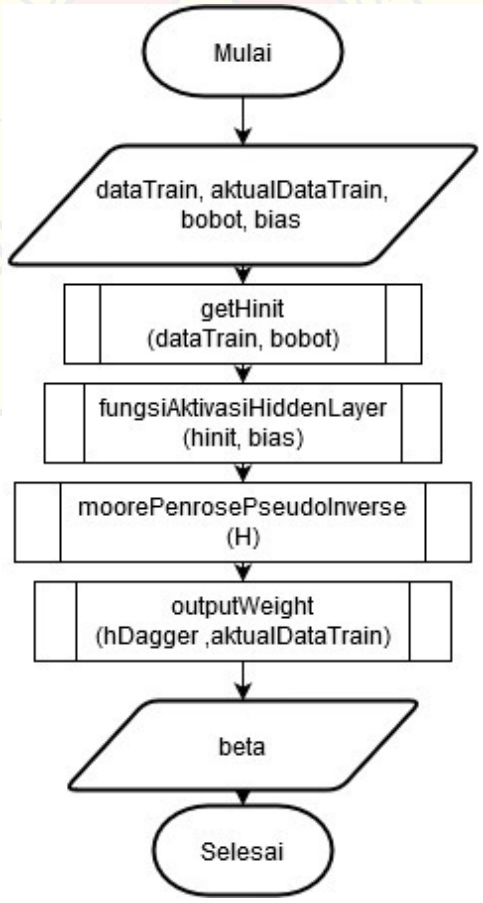


Gambar 8.10 Diagram Alir Normalisasi Data

Penjelasan dari diagram alir normalisasi data pada Gambar 3.10 untuk setiap langkah-langkahnya adalah sebagai berikut.

7. Memasukan parameter dataFitur, min dan max.
8. Membuat variabel selisihMinMax untuk menyimpan selisih nilai min max dan array normalisasi untuk menyimpan hasil perhitungan normalisasi data.
9. Melakukan perulangan i sesuai dengan panjang baris pada normalisasi.
10. Melakukan perulangan j sesuai dengan panjang kolom pada normalisasi.
11. Proses memasukan hasil perhitungan normalisasi data kedalam normalisasi.
12. Mengembalikan nilai normalisasi dari hasil perhitungan normalisasi data.

8.2.1.3 Alir Proses *Training*

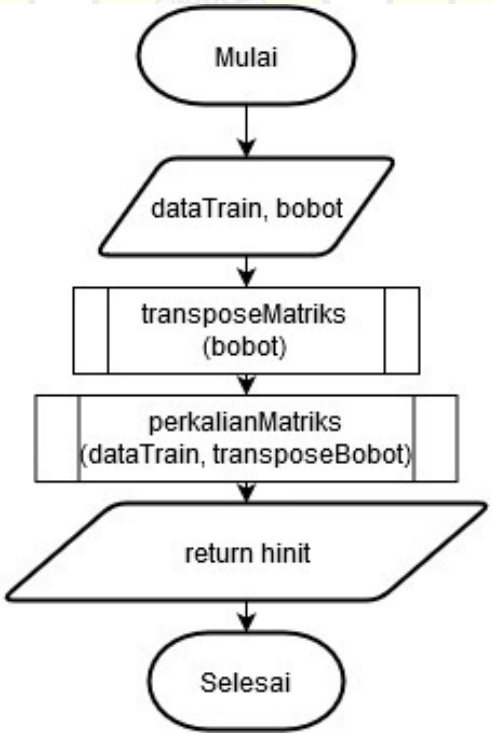


Gambar 8.11 Diagram Alir Proses *Training*

Penjelasan dari diagram alir proses *training* data pada Gambar 3.11 untuk setiap langkah-langkahnya adalah sebagai berikut.

7. Memasukan parameter data *training*, target data *training*, bobot, dan bias.
8. Melakukan proses perhitungan nilai H_{init} menggunakan data fitur dan bobot.
9. Melakukan proses perhitugan nilai *hidden layer* menggunakan fungsi aktivasi *sigmoid biner*.
10. Melakukan proses perhitungan nilai H dagger dengan menggunakan inverse matriks OBE.
11. Melakukan proses perhitungan nilai *output weight* sebagai perhitungan terakhir pada proses *training*.
12. Mengembalikan nilai beta sebagai hasil dari perhitungan *output weight*.

(a) Alir Menghitung Nilai H_{init}

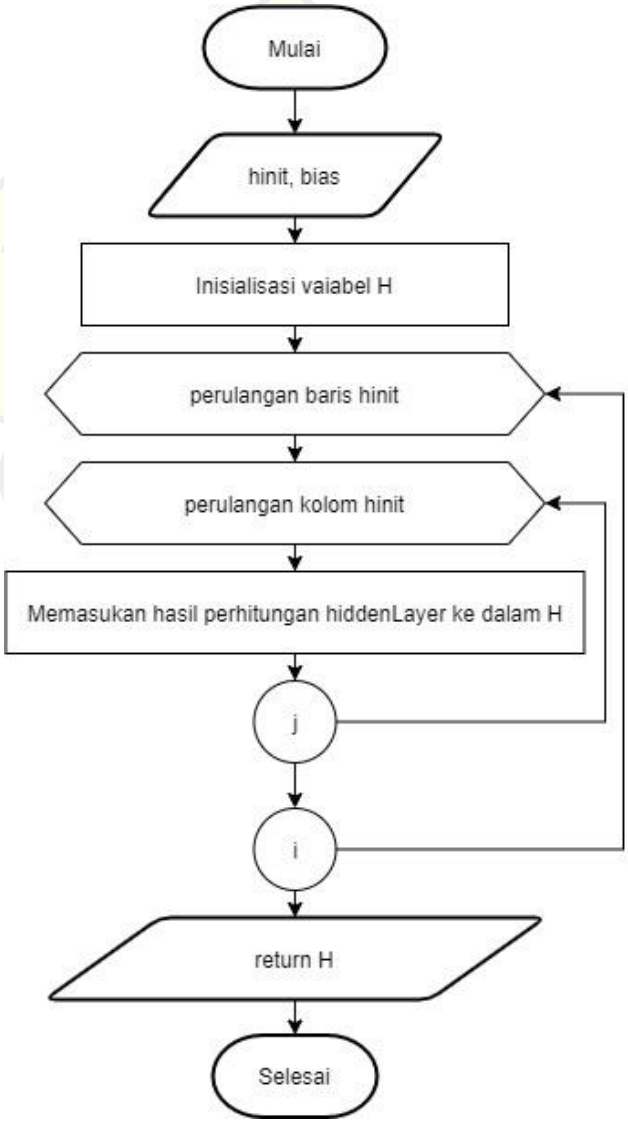


Gambar 8.12 Diagram Alir Menghitung Nilai H_{init}

Penjelasan dari diagram alir menghitung nilai H_{init} pada Gambar 3.12 untuk setiap langkah-langkahnya adalah sebagai berikut.

5. Memasukan parameter data *training* dan bobot.
6. Melakukan *transpose* matriks bobot sebagai bahan untuk perhitungan H_{init}.
7. Melakukan proses perhitungan nilai H_{init} menggunakan bobot dan data fitur.
8. Mengembalikan nilai hinit sebagai hasil dari perhitungan H_{init}.

(b) Alir Menghitung *Hidden Layer*

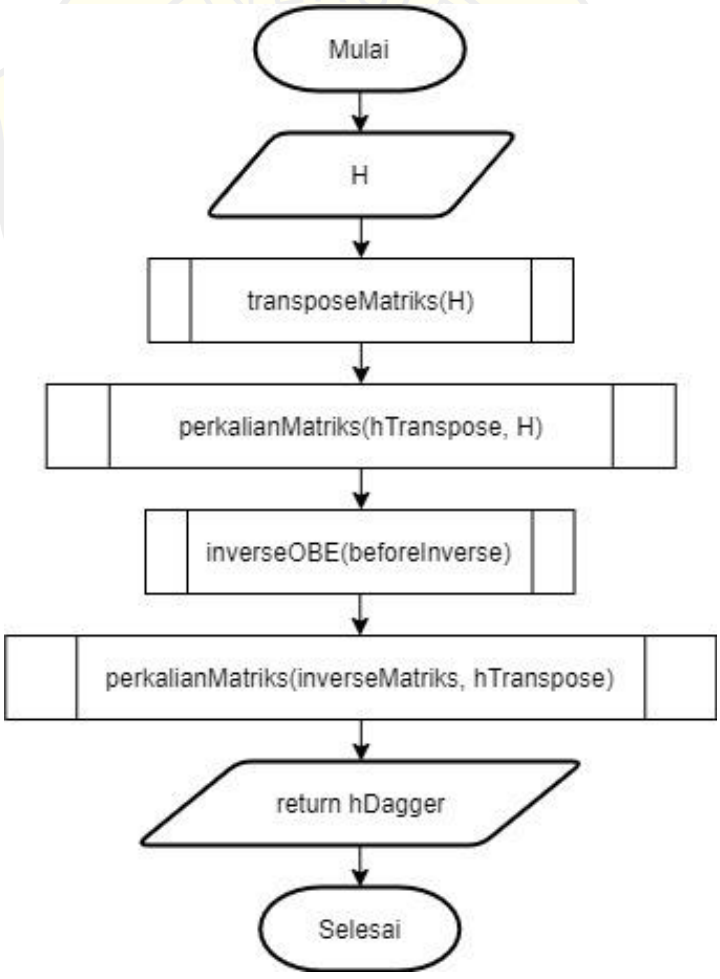


Gambar 8.13 Diagram Alir Menghitung *Hidden Layer*

Penjelasan dari diagram alir menghitung *hidden layer* pada Gambar 3.13 untuk setiap langkah-langkahnya adalah sebagai berikut.

7. Memasukan parameter nilai H_{init} dan nilai bias.
8. Membuat array H untuk menyimpan hasil perhitungan *hidden layer*.
9. Melakukan perulangan i sesuai dengan panjang baris pada array H.
10. Melakukan perulangan j sesuai dengan panjang kolom pada array H.
11. Melakukan perhitungan *hidden layer* menggunakan fungsi *sigmoid biner*.
12. Mengembalikan nilai H sebagai hasil dari perhitungan *hidden layer*.

(c) Alir *Moore-Penrose Pseudo Inverse*

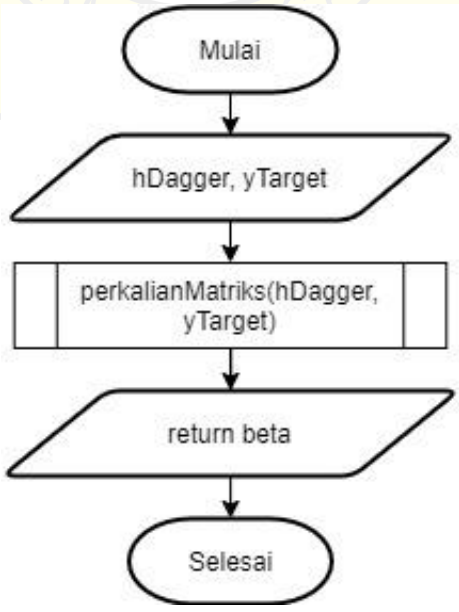


Gambar 8.14 Diagram Alir *Moore-Penrose Pseudo Inverse*

Penjelasan dari diagram alir *Moore-Penrose Pseudo Inverse* pada Gambar 3.14 untuk setiap langkah-langkahnya adalah sebagai berikut.

7. Memasukan parameter H sebagai nilai *hidden layer*.
8. Melakukan *transpose* matriks *hidden layer*.
9. Melakukan proses perkalian matriks *transpose hidden layer* dengan *hidden layer*. Hasil perkalian matriks akan digunakan untuk menghitung *inverse* matriks.
10. Melakukan perhitungan *inverse* matriks menggunakan perhitungan *inverse* matriks OBE.
11. Melakukan perhitungan nilai H *dagger* menggunakan hasil *inverse* matriks dan *transpose hidden layer*.
12. Mengembalikan nilai hDagger sebagai hasil perhitungan *moore-penrose pseudo inverse.*

(d) Alir Menghitung *Output Weight*

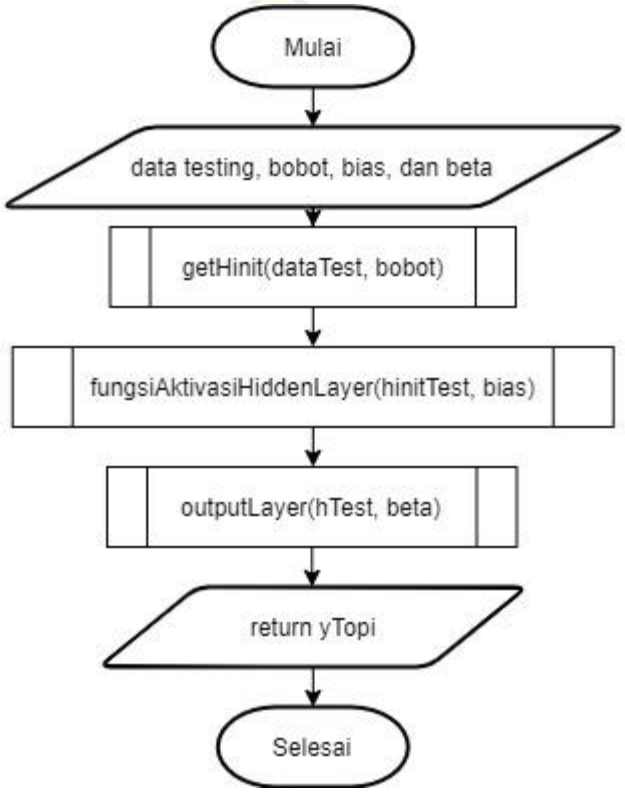


Gambar 8.15 Diagram Alir Menghitung *Output Weight*

Penjelasan dari diagram alir menghitung *output weight* pada Gambar 3.15 untuk setiap langkah-langkahnya adalah sebagai berikut.

4. Memasukan parameter H *dagger* dan data target pada data *training*.
5. Melakukan perhitungan nilai *output weight* menggunakan hasil perhitungan H *dagger* dan data target pada data *training*.
6. Mengembalikan nilai beta sebagai hasil dari perhitungan *output weight*.

8.2.1.4 Alir Proses *Testing*



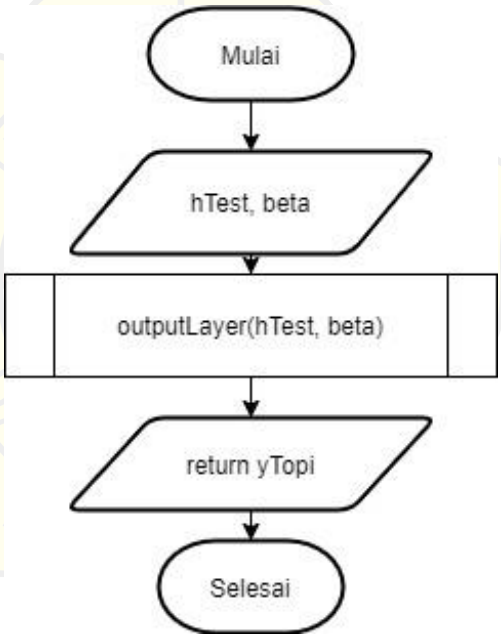
Gambar 8.16 Diagram Alir Proses Testing

Penjelasan dari diagram alir proses *testing* data pada Gambar 3. untuk setiap langkah-langkahnya adalah sebagai berikut.

6. Memasukan parameter data *testing*, bobot, bias, dan beta.

7. Melakukan proses perhitungan nilai H_{init} menggunakan data *testing* dan bobot.
8. Melakukan proses perhitugan nilai *hidden layer* menggunakan fungsi aktivasi *sigmoid biner*.
9. Melakukan proses perhitungan nilai *output layer* sebagai perhitungan terakhir pada proses *testing*.
10. Mengembalikan nilai yTopi sebagai hasil dari perhitungan *output layer*.

(a) Alir Menghitung *Output Layer*



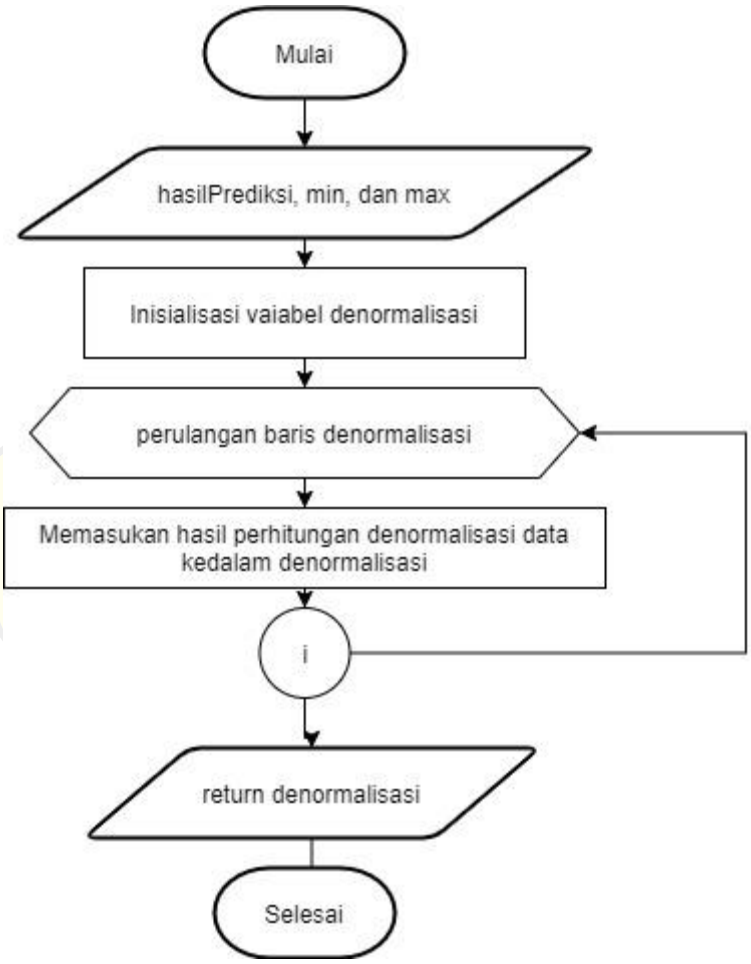
Gambar 8.17 Diagram Alir *Output Layer*

Penjelasan dari diagram alir menghitung *output layer* data pada Gambar 3.17 untuk setiap langkah-langkahnya adalah sebagai berikut.

4. Memasukan parameter *hidden layer* pada proses *testing* dan nilai beta yang dihasilkan pada proses *training*.

5. Melakukan perhitungan nilai *output layer* menggunakan hasil perhitungan *hidden layer* dan nilai beta sebagai *output weight*.
6. Mengembalikan nilai yTopi sebagai hasil dari perhitungan *output layer*.

8.2.1.5 Alir Denormalisasi Data



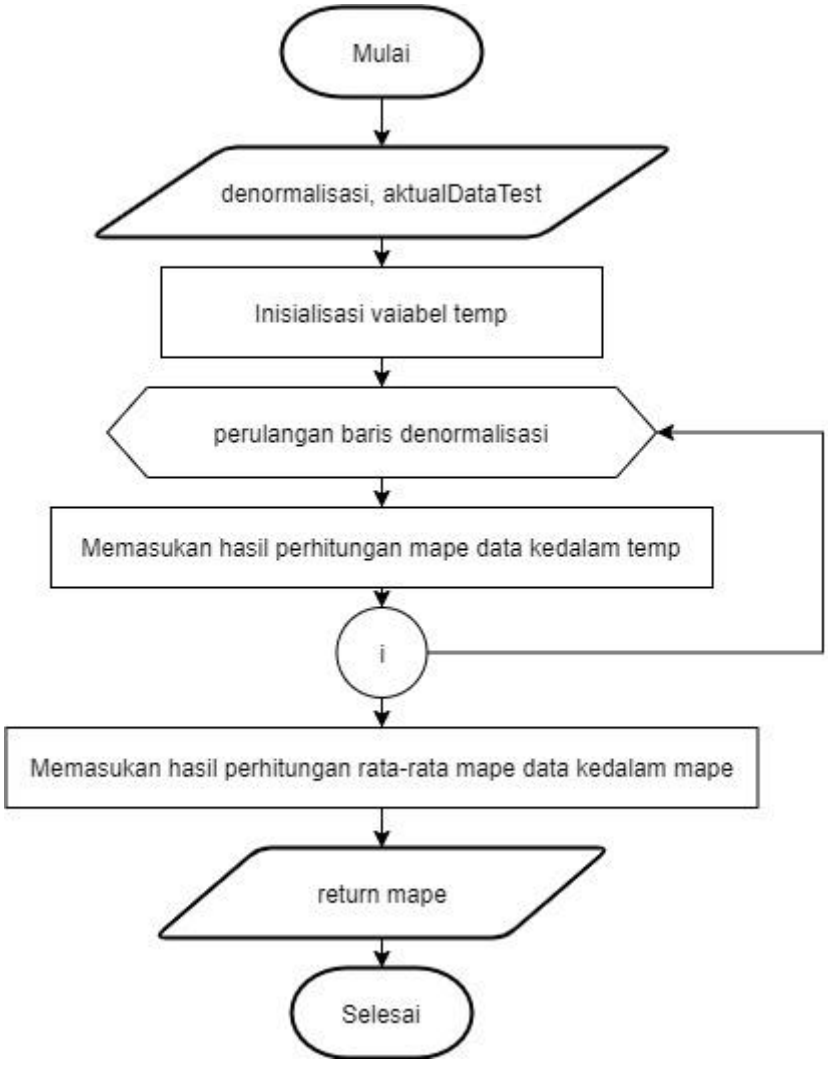
Gambar 8.18 Diagram Alir Denormalisasi Data

Penjelasan dari diagram alir denormalisasi data data pada Gambar 318. untuk setiap langkah-langkahnya adalah sebagai berikut.

6. Memasukan parameter nilai hasil prediksi, min dan max.
7. Membuat array denormalisasi untuk menyimpan hasil perhitungan denoormalisasi data.

8. Melakukan perulangan i sesuai dengan panjang baris pada denormalisasi.
9. Proses memasukan hasil perhitungan denormalisasi data kedalam denormalisasi.
10. Mengembalikan nilai denormalisas sebagai hasil dari perhitungan denormalisasi data.

8.2.1.6 Alir Menghitung MAPE



Gambar 8.19 Diagram Alir Menghitung MAPE

Penjelasan dari diagram alir menghitung MAPE data pada Gambar 3.19 untuk setiap langkah-langkahnya adalah sebagai berikut.

6. Memasukan parameter nilai denormalisasi dan data target pada data *testing*.
7. Membuat variable temp sebagai penyimpanan sementara untuk perhitungan mape.
8. Melakukan perulangan i sesuai dengan panjang baris pada denormalisasi.
9. Proses memasukan hasil perhitungan mape sementara untuk setiap data kedalam temp.
10. Proses memasukan hasil perhitungan rata-rata nilai mape kedalam mape

Mengembalikan nilai mape sebagai hasil perhitungan mape dan merupakan representasi tingkat akurasi dari algoritme ELM.

8.2.2 Perhitungan Manualisasi Algoritme ELM

Perhitungan manualisasi merupakan perhitungan manual yang mana hasil perhitungan tersebut digunakan sebagai acuan untuk menilai kebenaran dari sebuah kode program. Untuk melakukan perhitungan manualisasi dibutuhkan alat bantu berupa Microsoft Excel dan data sample dari jumlah populasi. Berikut merupakan data yang digunakan untuk perhitungan manualisasi ditunjukan pada Tabel 3.1.

Tabel 8.1 Data Harga Cabai Rawit

No	Tanggal	Harga Cabai Rawit
1	24/11/2020	40000
2	25/11/2020	40000
3	26/11/2020	40000
4	27/11/2020	50000
5	30/11/2020	50000

6	01/12/2020	50000
7	02/12/2020	50000
8	03/12/2020	50000
9	04/12/2020	52500
10	07/12/2020	60000
11	08/12/2020	60000
12	10/12/2020	60000
13	24/11/2020	40000

8.2.2.1 Inisialisasi Jumlah Fitur, Hidden Neuron, Bobot dan Bias

Inisialisasi jumlah fitur yang digunakan pada perhitungan manualiasasi sebanyak 2 fitur dan jumlah *hidden neuron* sebanyak 3 *neuron*. Fitur merepresentasikan bagian dari data target sehingga fitur merupakan suatu nilai yang membentuk data target. Fitur yang digunakan pada perhitungan manualisasi adalah data sebelumnya. Jika ditentukan 2 fitur maka data yang digunakan untuk membentuk data target adalah dua data sebelumnya. Fitur 1 akan dituliskan pada tabel sebagai data pertama, Fitur 2 untuk data kedua, Y sebagai data target. Berikut adalah tabel data fitur ditunjukan pada Tabel 3.2.

Tabel 8.2 Data Fitur

Data Ke-	Fitur 1	Fitur 2	Y
1	40000	40000	40000
2	40000	40000	50000
3	40000	50000	50000
4	50000	50000	50000
5	50000	50000	50000
6	50000	50000	50000
7	50000	50000	52500

8	50000	52500	60000
9	52500	60000	60000
10	60000	60000	60000

Data hasil pembetukan fitur akan dibagi menjadi dua bagian, yaitu data *training* dan data *testing*. Persentase perbandingan jumlah data *training* dan data *testing* adalah 80%:20% sehingga jumlah data *training* yang akan digunakan sebanyak 8 data dan jumlah data *testing* yang akan digunakan sebanyak 2 data. Berikut adalah tabel hasil pembagian data ditunjukan pada Tabel 3.3 dan Tabel 3.4.

Tabel 8.3 Data *Training*

Data Ke-	Fitur 1	Fitur 2	Y
1	40000	40000	40000
2	40000	40000	50000
3	40000	50000	50000
4	50000	50000	50000
5	50000	50000	50000
6	50000	50000	50000
7	50000	50000	52500
8	50000	52500	60000

Tabel 8.4 Data *Testing*

Data Ke-	Fitur 1	Fitur 2	Y
1	52500	60000	60000
2	60000	60000	60000

Inisialisasi nilai bobot berkisar antara [-1, 1], sedangkan nilai *bias* berkisar antara [-1,1]. Besar ukuran matriks bobot W_{jk} ditentukan pada jumlah fitur (k) dan *hidden neuron* (j) sedangkan besar ukuran matriks bias hanya ditentukan pada

jumlah *hidden neuron*-nya saja. Berikut adalah tabel hasil generalisasi nilai bobot dan bias ditunjukan pada Tabel 3.5 dan Tabel 3.6.

Tabel 8.5 Hasil Pembangkitan Bobot

	1	2
1	-0,37406	0,248636
2	0,400663	-0,45281
3	0,251504	-0,77196

Tabel 8.6 Hasil Pembangkitan Bias

	1	2	3
1	0,364511	-0,65676	-0,8332

8.2.2.2 Perhitungan Normalisasi Data

Perhitungan yang digunakan pada normalisasi data adalah perhitungan *Min-Max Normalization*, perhitungan tersebut membutuhkan nilai minimum dan nilai maksimum dari data *domain*. Namun, pada hasil perhitungan *Min-Max Normalization* nilai minimum dan nilai maksimum akan mendapatkan nilai 0 dan 1. Hal tersebut sangat mempengaruhi proses perhitungan algoritme ELM. Untuk memaksimalkan perhitungan dibutuhkan nilai tambahan untuk memanipulasi nilai minimum dan nilai maksimum dari data *domain*. Penambahan tersebut akan membuat nilai minimum dan nilai maksimum merupakan data yang tidak pernah dijangkau dari data *domain* sehingga hasil perhitungan *Min-Max Normalization* tidak mendapatkan nilai 0 dan 1. Berikut adalah tabel nilai minimum dan nilai maksimum dari data domain setelah dilakukan penambahan ditunjukan pada Tabel 3.7.

Tabel 8.7 Nilai Minimum dan Maksimum

	Harga Cabai Rawit
Minimum	35000
Maksimum	65000
Penambahan	5000

Langkah selanjutnya adalah menerapkan perhitungan *Min-Max Normalization* sesuai dengan Persamaan 2.2. Berikut adalah normalisasi data menggunakan perhitungan *Min-Max Normalization*.

$$X_{(1,1)} = \frac{(40000 - 35000)}{(65000 - 35000)}$$

$$X_{(1,1)} = \frac{(5000)}{(30000)}$$

$$X_{(1,1)} = 0{,}166667$$

Tabel hasil perhitungan *Min-Max Normalization* untuk data *training* dan data *testing* ditunjukan pada Tabel 3.8 dan Tabel 3.9.

Tabel 8.8 Normalisasi Data *Training*

Data Ke-	Fitur 1	Fitur 2	Y
1	0,166667	0,166667	0,166667
2	0,166667	0,166667	0,5
3	0,166667	0,5	0,5
4	0,5	0,5	0,5
5	0,5	0,5	0,5
6	0,5	0,5	0,5
7	0,5	0,5	0,583333
8	0,5	0,583333	0,833333

Tabel 8.9 Normalisasi Data *Testing*

Data Ke-	Fitur 1	Fitur 2	Y
1	0,583333	0,833333	0,833333
2	0,833333	0,833333	0,833333

8.2.2.3 Perhitungan Proses *Training*

(a) Menghitung Nilai H_{init}

Perhitungan nilai H_{init} membutuhkan *transpose* matriks bobot yang sebelumnya telah dibangkitkan. Berikut merupakan *transpose* matriks bobot ditunjukan pada Tabel 3.10.

Tabel 8.10 Hasil *Transpose* Bobot

	1	2	3
1	-0,37406	0,400663	0,251504
2	0,248636	-0,45281	-0,77196

Langkah selanjutnya setelah melakukan *transpose* matriks bobot adalah menghitung nilai H_{init} sesuai dengan Persamaan 2.6. Berikut adalah perhitungan nilai H_{init} untuk baris pertama dan kolom pertama.

$$H_{init_{(1,1)}} = \big((0{,}166667\ x\ (-0{,}374061)) + (0{,}166667\ x\ 0{,}248636)\big) + 0{,}3645106$$

$$H_{init_{(1,1)}} = \big((-0{,}0623435) + (0{,}0414393)\big) + 0{,}3645106$$

$$H_{init_{(1,1)}} = (-0{,}020904167) + 0{,}3645106$$

$$H_{init_{(1,1)}} = 0{,}3436064$$

Tabel hasil perhitungan H_{init} untuk data *training* ditunjukan pada Tabel 3.11.

Tabel 8.11 Hasil Perhitungan H_{init}

	1	2	3
1	0,3436064	-0,6654525	-0,9199431
2	0,3436064	-0,6654525	-0,9199431
3	0,4264851	-0,8163891	-1,1772627
4	0,3017981	-0,6828348	-1,0934281
5	0,3017981	-0,6828348	-1,0934281
6	0,3017981	-0,6828348	-1,0934281
7	0,3017981	-0,6828348	-1,0934281
8	0,3225178	-0,7205690	-1,1577580

(b) Menghitung Nilai *Hidden Layer*

Perhitungan nilai *hidden layer* disesuaikan dengan Persamaan 2.1. Berikut adalah perhitungan nilai *hidden layer* menggunakan fungsi aktivasi *sigmoid biner* untuk baris pertama dan kolom pertama.

$$H_{(1,1)} = \frac{1}{1 + \exp(-0{,}3436064)}$$

$$H_{(1,1)} = \frac{1}{1 + 0{,}709208}$$

$$H_{(1,1)} = \frac{1}{1{,}709208}$$

$$H_{(1,1)} = 0{,}5850663$$

Tabel hasil perhitungan *hidden layer* untuk data *training* ditunjukan pada Tabel 3.12.

Tabel 8.12 Hasil Perhitungan *Hidden Layer*

	1	2	3
1	0,5850663	0,3395159	0,2849695
2	0,5850663	0,3395159	0,2849695
3	0,6050340	0,3065307	0,2355447
4	0,5748820	0,3356289	0,2509733
5	0,5748820	0,3356289	0,2509733
6	0,5748820	0,3356289	0,2509733
7	0,5748820	0,3356289	0,2509733
8	0,5799377	0,3272677	0,2390749

(c) Menghitung Nilai H *dagger*

Perhitungan nilai H *dagger* membutuhkan *transpose* matriks *hidden layer* yang sebelumnya telah dihitung. Berikut merupakan *transpose* matriks *hidden layer* ditunjukan pada Tabel 3.13.

Tabel 8.13 Hasil *Transpose Hidden Layer*

	1	2	3	4	5	6	7	8
1	0,5850	0,5850	0,6050	0,5748	0,5748	0,5748	0,5748	0,5799
2	0,3395	0,3395	0,3065	0,3356	0,3356	0,3356	0,3356	0,3272
3	0,2849	0,2849	0,2355	0,2509	0,2509	0,2509	0,2509	0,2390

Langkah selanjutnya setelah melakukan *transpose* matriks *hidden layer* adalah menghitung nilai H *dagger* sesuai dengan Persamaan 2.6. Perhitungan H dagger dibagi menjadi tiga tahap. Tahap pertama, yaitu menghitung perkalian matriks

transpose hidden layer dengan matriks *hidden layer* untuk baris pertama dan kolom pertama.

$$(H^T.H)_{(1,1)} = (0{,}5850663\ x\ 0{,}5850663) + (0{,}5850663\ x\ 0{,}5850663) + (0{,}6050340\ x\ 0{,}6050340) + (0{,}5748820\ x\ 0{,}5748820) + (0{,}5748820\ x\ 0{,}5748820) + (0{,}5748820\ x\ 0{,}5748820) + (0{,}5748820\ x\ 0{,}5748820) + (0{,}5799377\ x\ 0{,}5799377)$$

$$(H^T.H)_{(1,1)} = (0{,}3423025) + (0{,}3423025) + (0{,}3660661) + (0{,}3304893) + (0{,}3304893) + (0{,}3304893) + (0{,}3304893) + (0{,}3363277)$$

$$(H^T.H)_{(1,1)} = 2{,}7089564$$

Tabel hasil perkalian *transpose* matriks *hidden layer* dengan matriks *hidden layer* pada perhitungan nilai H *dagger* ditunjukan pada Tabel 3.14.

Tabel 8.14 Hasil Perkalian *Transpose* Matriks *Hidden Layer*

	1	2	3
1	2,7089564	1,5443230	1,1917334
2	1,5443230	0,8821943	0,6808821
3	1,1917334	0,6808821	0,5270037

Tahap kedua, yaitu menghitung *inverse* matriks. Perhitungan *inverse* matriks yang digunakan merupakan *inverse* matriks OBE. Berikut merupakan tabel hasil perhitungan *inverse* matriks OBE ditunjukan pada Tabel 3.15.

Tabel 8.15 Hasil *Inverse* Matriks

	1	2	3
1	181,05711	-334,05616	22,1650364
2	-334,05616	1015,8058	-556,99433
3	22,165036	-556,99433	671,404425

Tahap ketiga atau tahap terakhir untuk menyelesaikan perhitungan nilai H *dagger*, yaitu mengkalikan hasil perhitungan *inverse* matriks dengan *transpose* matriks *hidden layer*. Berikut merupakan hasil perkalian *inverse* matriks dengan *transpose* matriks *hidden layer* untuk baris pertama dan kolom pertama.

$$H^{\dagger}_{(1,1)} = (1181{,}05711\ x\ 0{,}5850663\ ) + \big((-334{,}05616)\ x\ 0{,}3395158\ \big) + (22{,}1650364\ x\ 0{,}2849694)$$

$$H^{\dagger}_{(1,1)} = 105{,}9304118 + (-113{,}417364) + 6{,}3163591$$

$$H^{\dagger}_{(1,1)} = -1{,}1705937$$

Tabel hasil perkalian *inverse* matriks dengan *transpose* matriks *hidden layer* pada perhitungan nilai H *dagger* ditunjukan pada Tabel 3.16.

Tabel 8.16 Hasil Perhitungan H *dagger*

	1	2	3	4	5	6	7	8
1	-1,1706	-1,1706	12,3681	-2,469	-2,469	-2,469	-2,469	0,9752
2	-9,2892	-9,2892	-21,936	9,1002	9,1002	9,1002	9,1002	5,5453
3	15,1894	15,1894	0,8205	-5,696	-5,696	-5,696	-5,696	-8,916

(d) Menghitung Nilai *Output Weight*

Proses perhitungan *output weight* merupakan tahap terakhir pada proses *training*. Proses perhitungan *output weight* disesuaikan dengan Persamaan 2.7. Berikut adalah perhitungan output weight untuk baris pertama dan kolom pertama.

$$\hat{\beta} = ((-1{,}1705938)\ x\ 0{,}1666667) + (-1{,}1705938\ x\ 0{,}5) + (12{,}3681038\ x\ 0.5) + \big((-2{,}4695954)\ x\ 0.5\big) + \big((-2{,}4695954)\ x\ 0.5\big) + \big((-2{,}4695954)\ x\ 0{,}5833333\big) + (0{,}9751568\ x\ 0{,}5833333)$$

$$\hat{\beta} = (-0{,}1950990) + (-0{,}5852969) + 6{,}1840519 + (-1{,}2347977) + (-1{,}2347977) + (-1{,}2347977) + (-1{,}4405973) + 0{,}8126307$$

$$\hat{\beta} = 1{,}0712963$$

Tabel hasil perhitungan *output weight* ditunjukan pada Tabel 3.17.

Tabel 8.17 Hasil Perhitungan Output Weight

	1
1	1,0712963
2	6,4186475
3	-8,7612271

8.2.2.4 Perhitungan Proses *Testing*

(a) Menghitung Nilai H_{init}

Perhitungan nilai H_{init} membutuhkan *transpose* matriks bobot yang sebelumnya telah dibangkitkan pada proses *training*. Perhitungan nilai H_{init} disesuaikan dengan Persamaan 2.6. Berikut adalah perhitungan nilai H_{init} untuk baris pertama dan kolom pertama.

$$H_{init_{(1,1)}} = \big((0{,}5833333\ x\ (-0{,}374061)) + (0{,}8333333\ x\ 0{,}248636)\big) + 0{,}3645106$$

$$H_{init_{(1,1)}} = \big((-0{,}2182023) + (0{,}2071967)\big) + \ 0{,}3645106$$

$$H_{init_{(1,1)}} = (-0{,}0110056) + \ 0{,}3645106$$

$$H_{init_{(1,1)}} = 0{,}3535050$$

Tabel hasil perhitungan H_{init} untuk data *testing* ditunjukan pada Tabel 3.18.

Tabel 8.18 Hasil Perhitungan H_{init}

	1	2	3
1	0,3535050	-0,8003829	-1,3297891
2	0,2599898	-0,7002171	-1,2669131

(b)Menghitung Nilai *Hidden Layer*

Perhitungan nilai *hidden layer* disesuaikan dengan Persamaan 2.1. Berikut adalah perhitungan nilai *hidden layer* menggunakan fungsi aktivasi *sigmoid biner* untuk baris pertama dan kolom pertama.

$$H_{(1,1)} = \frac{1}{1 + \exp\,(-0{,}3535050)}$$

$$H_{(1,1)} = \frac{1}{1 + 0{,}7022225}$$

$$H_{(1,1)} = \frac{1}{1{,}7022225}$$

$$H_{(1,1)} = \ 0{,}5874673$$

Tabel hasil perhitungan *hidden layer* untuk data *training* ditunjukan pada Tabel 3.19.

Tabel 8.19 Hasil Perhitungan *Hidden Layer*

	1	2	3
1	0,5874673	0,3099436	0,2091943
2	0,5646338	0,3317641	0,2197861

(c) Menghitung Nilai *Output Layer*

Proses perhitungan *output layer* merupakan tahap terakhir pada proses *testing*. Proses perhitungan *output layer* disesuaikan dengan Persamaan 2.8. Berikut adalah perhitungan *output layer* untuk baris pertama dan kolom pertama.

$$\hat{Y} = \Big((0{,}5874672\ x\ 1{,}0712963) + (0{,}3099436\ x\ 6{,}4186475) + \big(0{,}20919426\ x\ (-8{,}7612271)\big)\Big)$$

$$\hat{Y} = 0{,}6293515 + 1{,}9894189 + (-1{,}8327984)$$

$$\hat{Y} = 0{,}7859720$$

Tabel hasil perhitungan *output layer* ditunjukan pada Tabel 3.20.

Tabel 8.20 Hasil Perhitungan *Output Layer*

	1
1	0,7859720
2	0,8087705

8.2.2.5 Perhitungan Denormalisasi Data

Perhitungan denormalisasi data disesuaikan dengan Persamaan 2.3. Berikut adalah perhitungan nilai denormalisasi data untuk baris pertama dan kolom pertama.

$$X_{(1,1)} = 0{,}7859720.\ (65000 - 35000) + 35000$$

$$X_{(1,1)} = 0{,}7859720.\,(30000) + 35000$$

$$X_{(1,1)} = 23579{,}16 + 35000$$

$$X_{(1,1)} = 58579{,}16$$

Tabel hasil perhitungan nilai denormalisasi data ditunjukan pada Tabel 3.21.

Tabel 8.21 Hasil Perhitungan Denormalisasi

	1
1	58579,16
2	59263,12

8.2.2.6 Perhitungan MAPE

Perhitungan rata-rata nilai MAPE disesuaikan dengan Persamaan 2.16. Berikut adalah perhitungan rata-rata nilai MAPE dengan menggunakan data target dan data hasil denormalisasi atau data prediksi.

$$MAPE = \frac{1}{2}x\left(\left(\left|\frac{58579{,}16 - 60000}{60000}\right| x\ 100\right) + \left(\left|\frac{59263{,}12 - 60000}{60000}\right| x\ 100\right)\right)$$

$$MAPE = \frac{1}{2}x(2{,}3680666 + 1{,}2281399)$$

$$MAPE = \frac{1}{2}x\ 3{,}5962065$$

$$MAPE = 1{,}7981032$$

Tabel perbandingan data prediksi dan data target ditunjukan pada bagian Tabel 3.22.

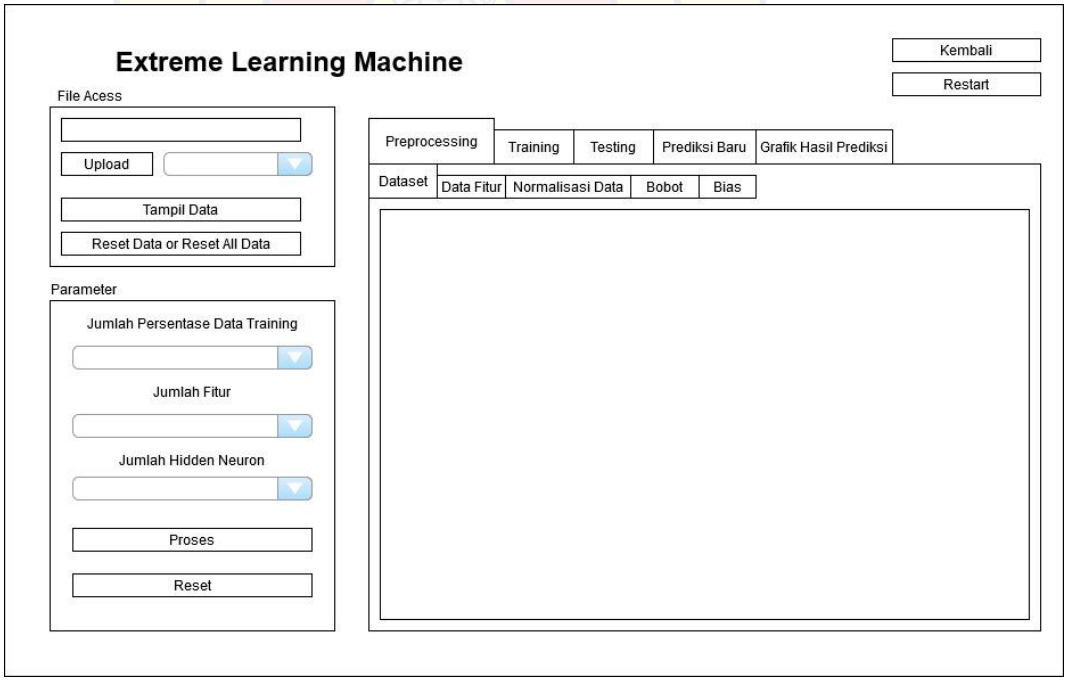
Tabel 8.22 Perbandingan Data Prediksi Dan Data Target

	Data Prediksi	Data Target
1	58579,16	60000
2	59263,12	60000

8.2.3 Perancangan Antarmuka Algoritme ELM

Perancangan antarmuka bertujuan untuk membuat rancangan mengenai bagian-bagian akan ditampilkan pada sistem. Sesuai dengan perancangan yang telah ditentukan terdapat 4 bagian yang akan ditampilkan, yaitu preprocessing, proses *training*, proses *testing*, proses prediksi, dan hasil prediksi.

8.2.3.1 Perancangan Antarmuka *Preprocessing*



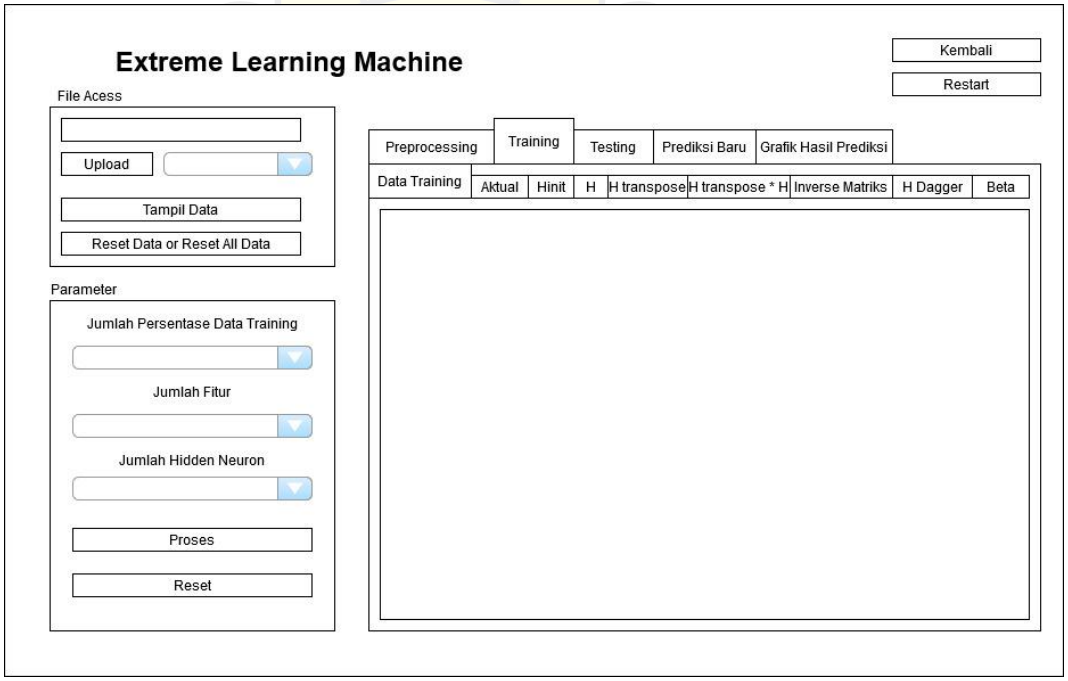
Gambar 8.20 Perancangan Antarmuka *Preprocessing*

Keterangan :

1. Nama metode yang digunakan pada penelitian.
2. Read data textbox digunakan untuk memberikan informasi mengenai lokasi dari file yang dipilih.
3. Button Upload digunakan untuk memilih data excel yang akan digunakan.
4. Input combobox digunakan untuk memilih sheet yang akan dibaca.
5. Button Tampilkan digunakan untuk menampilkan data excel kedalam sistem.
6. Button Reset Data or Reset All Data digunakan untuk mereset seluruh data pada sistem.
7. Input combobox digunakan untuk memilih persentase jumlah data *training*.
8. Input combobox digunakan untuk memilih jumlah fitur.
9. Input combobox digunakan untuk memilih jumlah *neuron*.
10. Button Proses digunakan untuk melakukan proses perhitungan algoritme.
11. Button Reset digunakan untuk mereset nilai parameter.
12. Tab Preprocessing digunakan untuk menampung seluruh tahap yang terdapat pada proses preprocessing.
13. Tab Dataset digunakan untuk menampilkan data excel.
14. Tab Data Fitur digunakan untuk menampilkan hasil pembuatan fitur menggunakan dataset.
15. Tab Normalisasi Data digunakan untuk menampilkan hasil perhitungan normalisasi menggunakan data fitur.

16. Tab Bobot digunakan untuk menampilkan hasil pembuatan nilai bobot menggunakan fungsi Random.
17. Tab Bias digunakan untuk menampilkan hasil pembuatan nilai bias menggunakan fungsi Random.
18. Button Kembali digunakan untuk menampilkan form sebelumnya.
19. Button Restart digunakan untuk mematikan sistem dan menjalankannya kembali.

8.2.3.2 Perancangan Antarmuka Proses *Training*



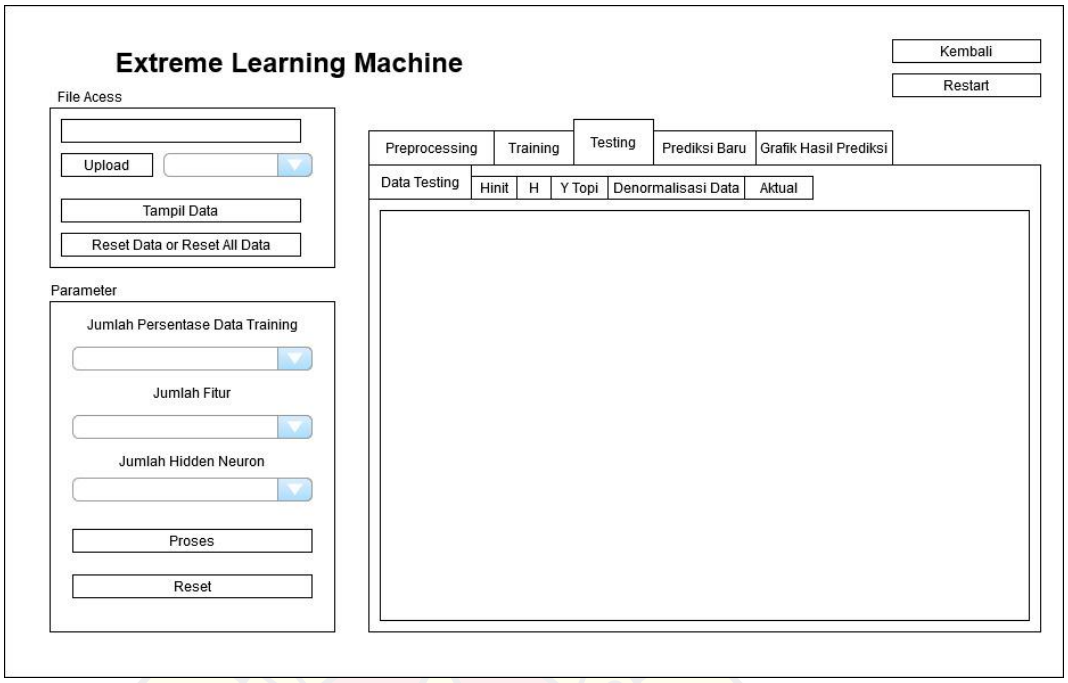
Gambar 8.21 Perancangan Antarmuka Proses *Training*

Keterangan :

1. Tab Training digunakan untuk menampung seluruh tahap yang terdapat pada proses *training*.

2. Tab Data Training digunakan untuk menampilkan hasil pembagian data *training* menggunakan data normalisasi.
3. Tab Aktual digunakan untuk menampilkan hasil pembagian data target menggunakan data normalisasi.
4. Tab Hinit digunakan untuk menampilkan hasil perhitungan nilai H_{init} pada proses *training*.
5. Tab H digunakan untuk menampilkan hasil perhitungan *hidden layer* pada proses *training* menggunakan fungsi aktivasi *sigmoid biner*.
6. Tab H transpose digunakan untuk menampilkan *transpose* matriks *hidden layer*.
7. Tab H transpose * H digunakan untuk menampilkan hasil perhitungan perkalian matriks untuk digunakan pada perhitungan *inverse* matriks.
8. Tab Inverse Matriks digunakan untuk menampilkan hasil perhitungan *inverse* matriks menggunakan perhitungan *inverse* matriks OBE.
9. Tab H Dagger digunakan untuk menampilkan hasil perhitungan *Moore-Penrose Pseudo Inverse*.
10. Tab Beta digunakan untuk menampilkan hasil perhitungan *output weight* pada proses *training*.

8.2.3.3 Perancangan Antarmuka Proses *Testing*



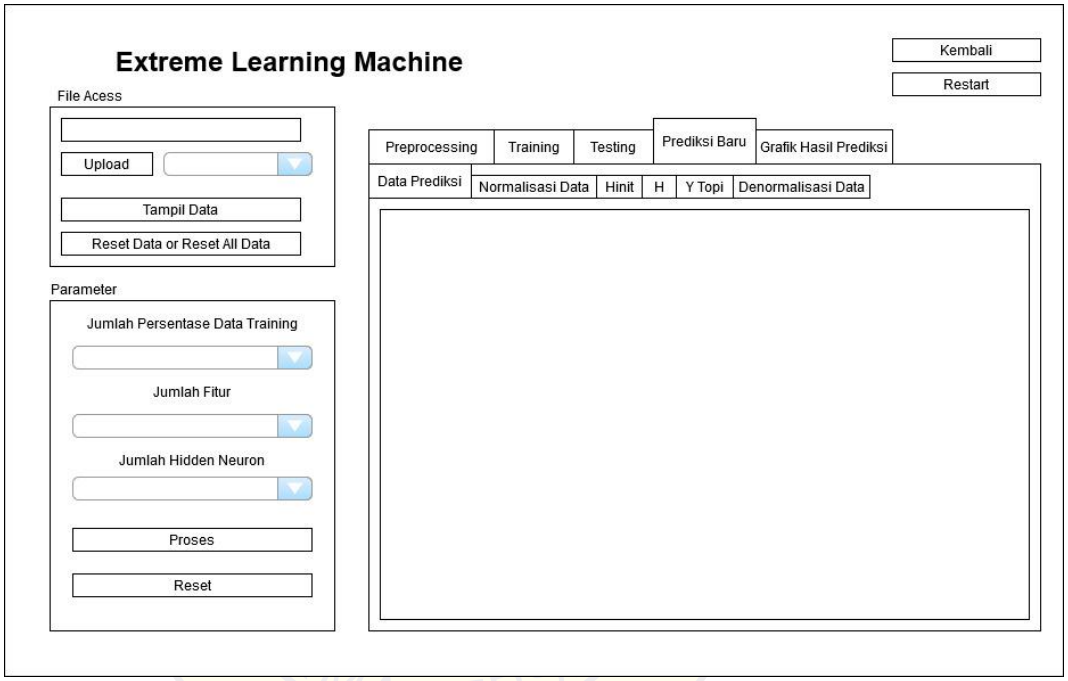
Gambar 8.22 Perancangan Antarmuka Proses *Testing*

Keterangan :

1. Tab Testing digunakan untuk menampung seluruh tahap yang terdapat pada proses *testing*.
2. Tab Data Testing digunakan untuk menampilkan hasil pembagian data *testing* menggunakan data normalisasi.
3. Tab Hinit digunakan untuk menampilkan hasil perhitungan nilai H_{init} pada proses *testing*.
4. Tab H digunakan untuk menampilkan hasil perhitungan nilai *hidden layer* pada proses *testing*.
5. Tab Y Topi digunakan untuk menampilkan hasil perhitungan *output layer* pada proses *testing*.

6. Tab Denormalisasi Data digunakan untuk menampilkan hasil pengembalian data prediksi kedalam bentuk sebenarnya.
7. Tab Aktual digunakan untuk menampilkan data aktual untuk proses testing menggunakan data fitur.

8.2.3.4 Perancangan Antarmuka Proses Prediksi



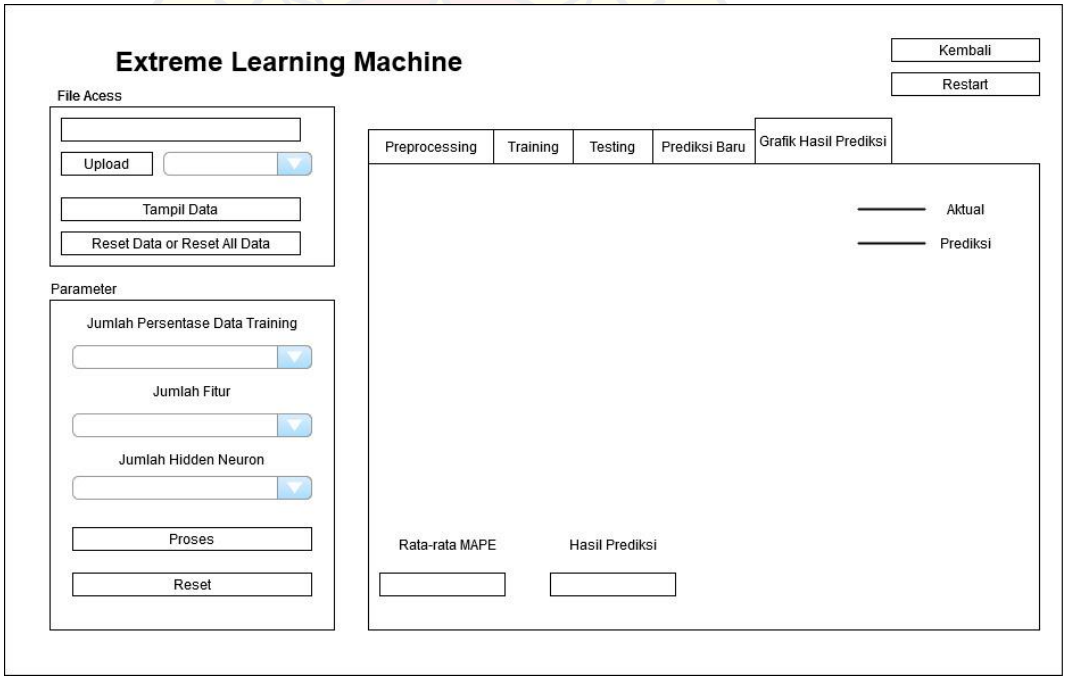
Gambar 8.23 Perancangan Antarmuka Proses Prediksi

Keterangan :

1. Tab Prediksi Baru digunakan untuk menampung seluruh tahap yang terdapat pada proses prediksi.
2. Tab Data Prediksi digunakan untuk menampilkan hasil pengambilan data prediksi menggunakan data fitur
3. Tab Normalisasi Data digunakan untuk menampilkan hasil perhitungan normalisasi data pada data prediksi.

4. Tab Hinit digunakan untuk menampilkan hasil perhitungan nilai H_{init} pada data prediksi.
5. Tab H digunakan untuk menampilkan hasil perhitungan nilai *hidden layer* pada data prediksi.
6. Tab Y Topi digunakan untuk menampilkan hasil perhitungan *output layer* pada data prediksi.
7. Tab Denormalisasi Data digunakan untuk menampilkan hasil pengembalian data prediksi kedalam bentuk sebenarnya.

8.2.3.5 Perancangan Antarmuka Hasil Prediksi



Gambar 8.24 Perancangan Antarmuka Hasil Prediksi

Keterangan :

1. Tab Grafik Hasil Prediksi digunakan untuk menampung seluruh bagian yang terdapat pada hasil prediksi.

2. Tampilan Grafik digunakan untuk menampilkan perbandingan data target dengan data prediksi
3. Read data textbox digunakan untuk menampilkan akurasi tingkat kesalahan menggunakan perhitungan mape.
4. Read data textbox digunakan untuk menampilkan hasil prediksi untuk satu hari kedepan.

8.2.4 Perancangan Pengujian Algoritme ELM

Pengujian parameter pada algoritme ELM ditunjukan untuk mengetahui parameter terbaik yang menghasilkan tingkat kesalahan terendah dengan menggunakan satuan rata-rata nilai MAPE. Karena nilai bobot dan bias dibangkitkan secara acak maka nilai hasil prediksi akan berbeda-beda walaupun menggunakan parameter yang sama. Untuk itu, diperlukan beberapa percobaan untuk setiap parameter untuk mengetahui rata-rata dari beberapa percobaan yang dilakukan. Pada pengujian parameter algortime ELM, peneliti menentukan banyaknya percobaan yang perlu dilakukan untuk setiap parameter berjumlah 5 kali percobaan. Berikut adalah urutan untuk melakukan pengujian parameter pada algoritme ELM.

1. Pengujian Jumlah Fitur
2. Pengujian Jumlah *Neuron*
3. Pengujian Persentase Jumlah Data *Training*

8.2.4.2 Pengujian Jumlah Fitur

Jumlah fitur yang digunakan mengacu pada penelitian yang dilakukan oleh Ikhsan, Setiawan, & Tibyani tahun 2019 yang mana pengujian untuk parameter jumlah fitur dimulai dari 2 sampai 10 fitur. Berikut adalah tabel perancangan pengujian jumlah fitur ditunjukan pada Tabel 3.23.

Tabel 8.23 Perancangan Pengujian Jumlah Fitur

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

8.2.4.3 Pengujian Jumlah *Neuron*

Jumlah *neuron* yang digunakan mengacu pada penelitian yang dilakukan oleh Ikhsan, Setiawan, & Tibyani tahun 2019 yang mana pengujian untuk parameter jumlah *neuron* dimulai dari 2 sampai 10 *neuron*. Berikut adalah tabel perancangan pengujian jumlah *neuron* ditunjukan pada Tabel 3.24.

Tabel 8.24 Perancangan Pengujian Jumlah *Neuron*

Jumlah Neuron	Percobaan Ke-					Rata-rata MAPE
	1	2	3	4	5	
2						
3						
4						
5						
6						
7						
8						
9						
10						

8.2.4.4 Pengujian Persentase Jumlah Data *Training*

Persentase jumlah data *training* yang digunakan mengacu pada penelitian yang dilakukan oleh Ikhsan, Setiawan, & Tibyani tahun 2019 yang mana pengujian untuk parameter persentase jumlah data *training* adalah 50%, 60%, 70%, dan 80% dari keseluruhan data fitur yang telah dibentuk. Kemudian persentase jumlah data *testing* yang digunakan adalah 20%. Jumlah tersebut digunakan untuk setiap pengujian data *training*. Berikut adalah tabel perancangan pengujian persentase jumlah data *training* ditunjukan pada Tabel 3.25.

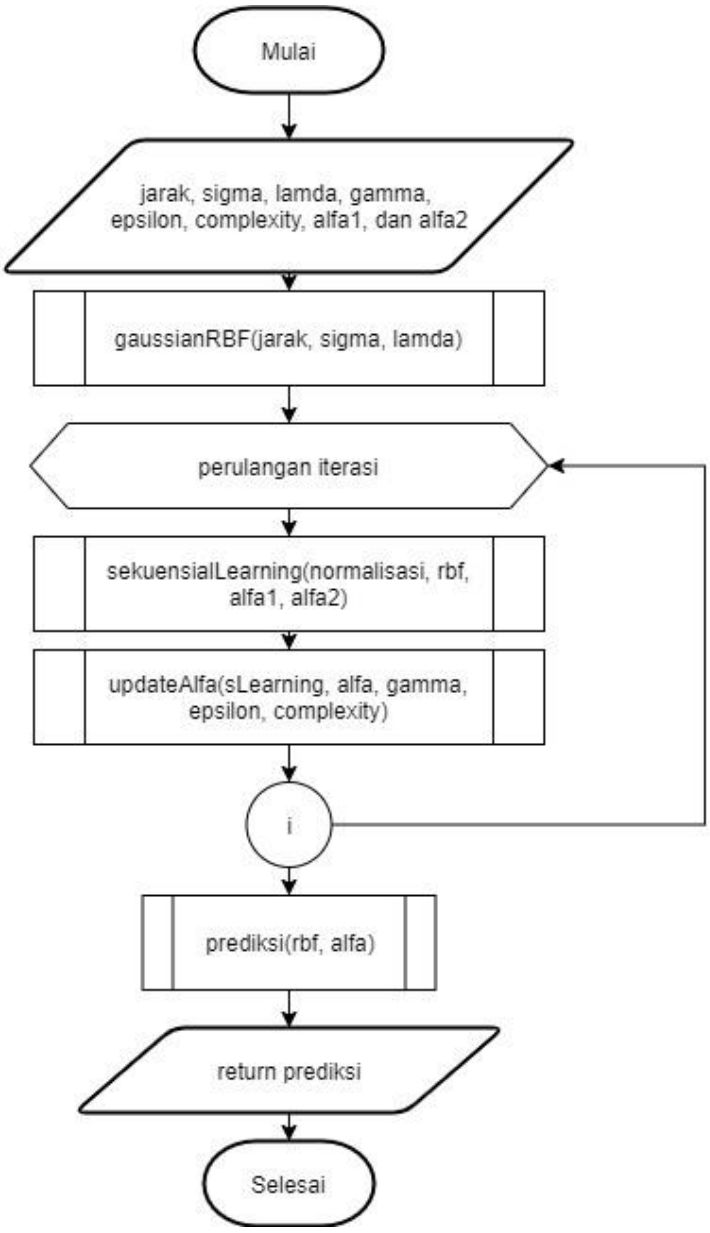
Tabel 8.25 Perancangan Pengujian Persentase Jumlah Data *Training*

Persentase Jumlah Data *Training*	Percobaan Ke-					Rata-rata MAPE
	1	2	3	4	5	
80%:20%						
70%:20%						

60%:20%						
50%:20%						

8.2.5 Perancangan Algoritme SVR

Alir perancangan algoritme SVR merupakan rancangan mengenai seluruh tahapan atau proses yang harus dilalui algroitme. Berikut adalah alur perancangan dari algoritme SVR ditunjukan pada Gambar 3.2.

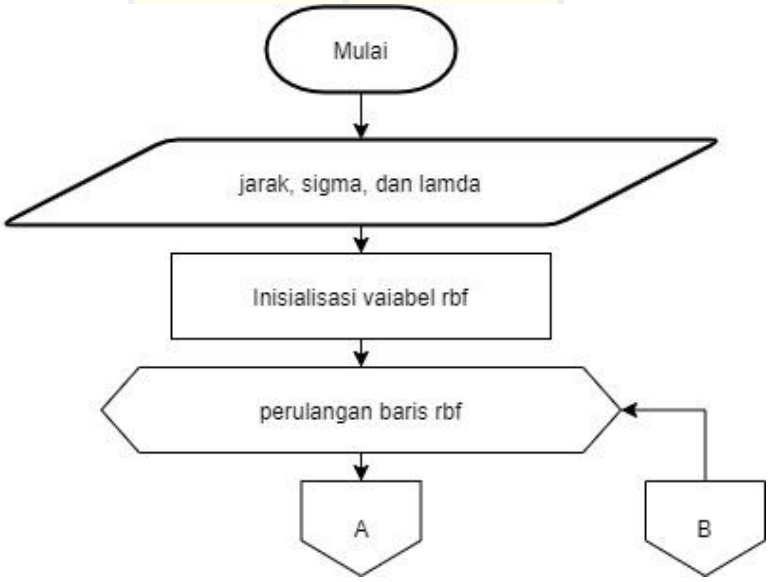


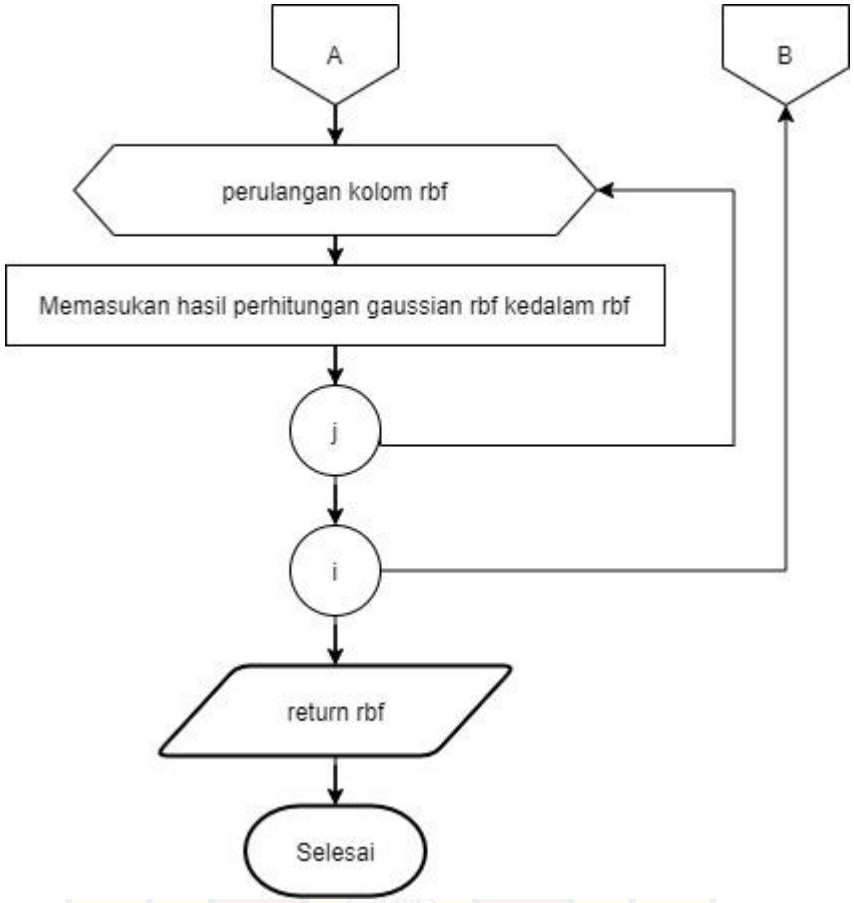
Gambar 8.25 Diagram Alir Alforitme SVR

Penjelasan dari diagram alir algoritme SVR pada Gambar 3.25 untuk setiap langkah-langkahnya adalah sebagai berikut.

1. Menentukan nilai parameter pada algoritme SVR.
2. Melakukan proses perhitungan matriks hessian menggunakan fungsi kernel *gasussian rbf*.
3. Melakukan perulangan i sesuai dengan panjang iterasi.
4. Melakukan proses perhitungan nilai *error* untuk setiap data target.
5. Melakukan proses perhitungan untuk mengupdate nilai *Lagrange Multiplier*.
6. Melakukan proses perhitungan fungsi regresi untuk menghasilkan nilai prediksi menggunakan matriks hessian dan nilai *Lagrange Multiplier* hasil iterasi.
7. Mengembalikan nilai prediksi sebagai hasil dari perhitungan fungsi regresi.

8.2.5.2 Alir Menghitung Matriks Hessian



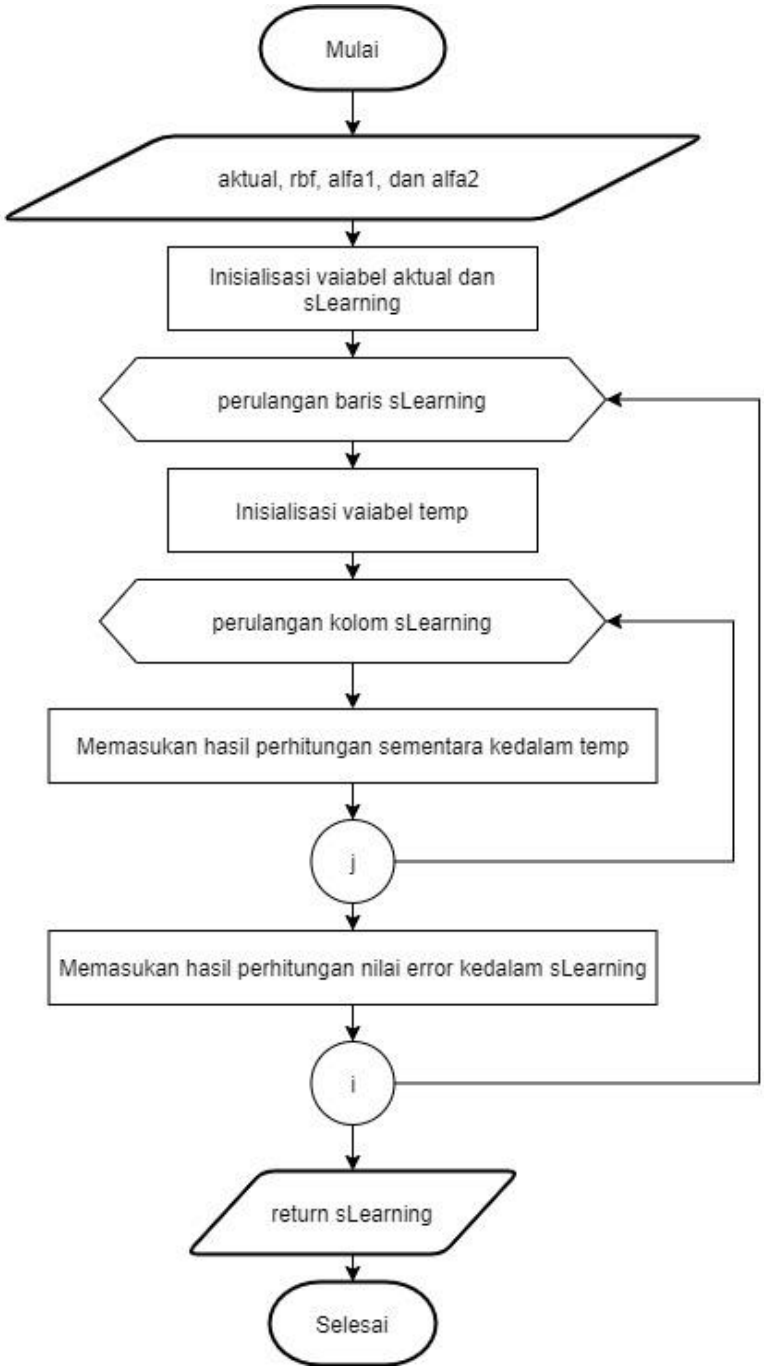


Gambar 8.26 Diagram Alir Menghitung Matriks Hessian

Penjelasan dari diagram alir menghitung matriks hessian pada Gambar 3.26 untuk setiap langkah-langkahnya adalah sebagai berikut.

7. Memasukan parameter hasil perhitungan jarak, *sigma* dan *lambda*.
8. Membuat array rbf untuk menyimpan hasil perhitungan matriks hessian.
9. Melakukan perulangan i sesuai dengan panjang baris pada rbf.
10. Melakukan perulangan j sesuai dengan panjang kolom pada rbf.
11. Melakukan perhitungan matriks hessian menggunakan fungsi kernel *gaussian rbf*.
12. Mengembalikan nilai rbf sebagai hasil dari perhitungan matriks hessian.

8.2.5.3 Alir Menghitung Nilai Error

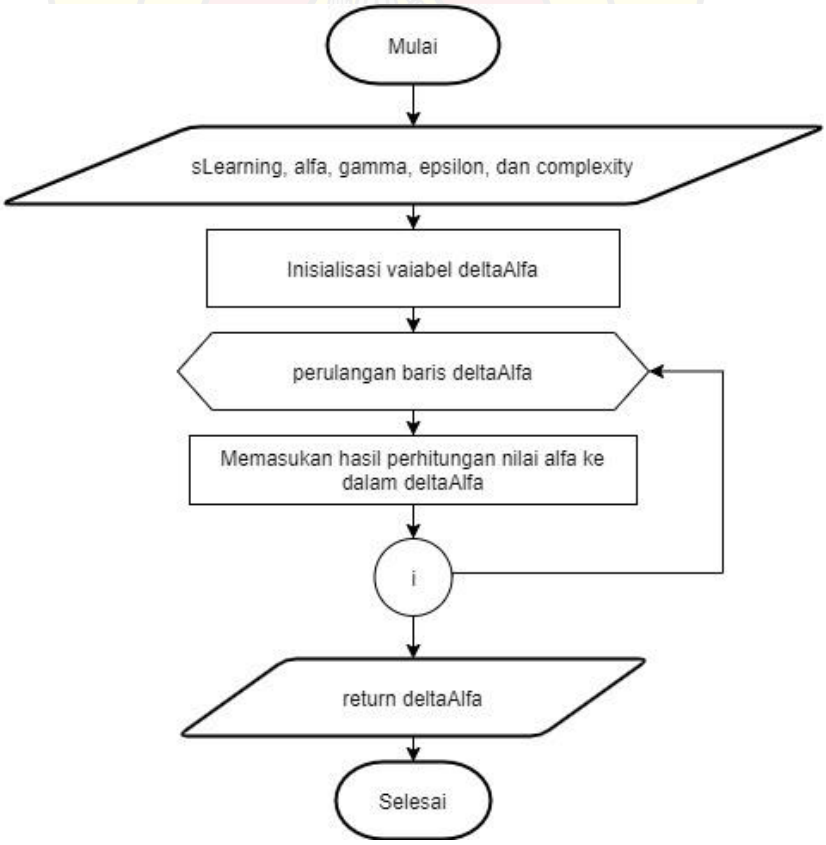


Gambar 8.27 Diagram Alir Menghitung Nilai *Error*

Penjelasan dari diagram alir menghitung nilai *error* pada Gambar 3.27 untuk setiap langkah-langkahnya adalah sebagai berikut.

8. Memasukan parameter data target, hasil perhitungan matriks hessian, alfa1, dan alfa2.
9. Membuat variable aktual untuk mengambil data target dan array sLearning untuk menyimpan hasil perhitungan nilai *error*.
10. Melakukan perulangan i sesuai dengan panjang baris pada sLearning.
11. Membuat variable temp untuk menyimpan perhitungan nilai *error* sementara.
12. Melakukan perulangan j sesuai dengan panjang kolom pada sLearning.
13. Melakukan perhitungan nilai *error* untuk setiap data target.
14. Mengembalikan nilai sLearning sebagai hasil dari perhitungan nilai *error*.

8.2.5.4 Alir Menghitung Nilai *Lagrange Multiplier*

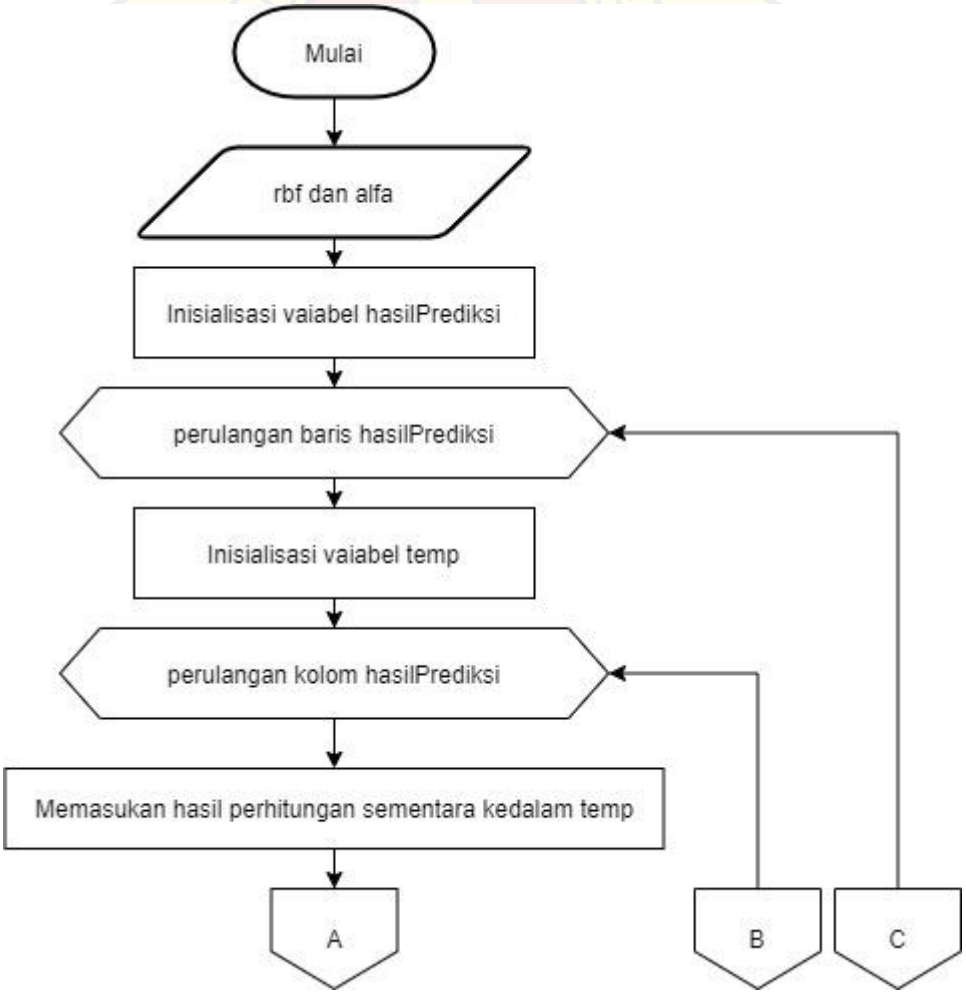


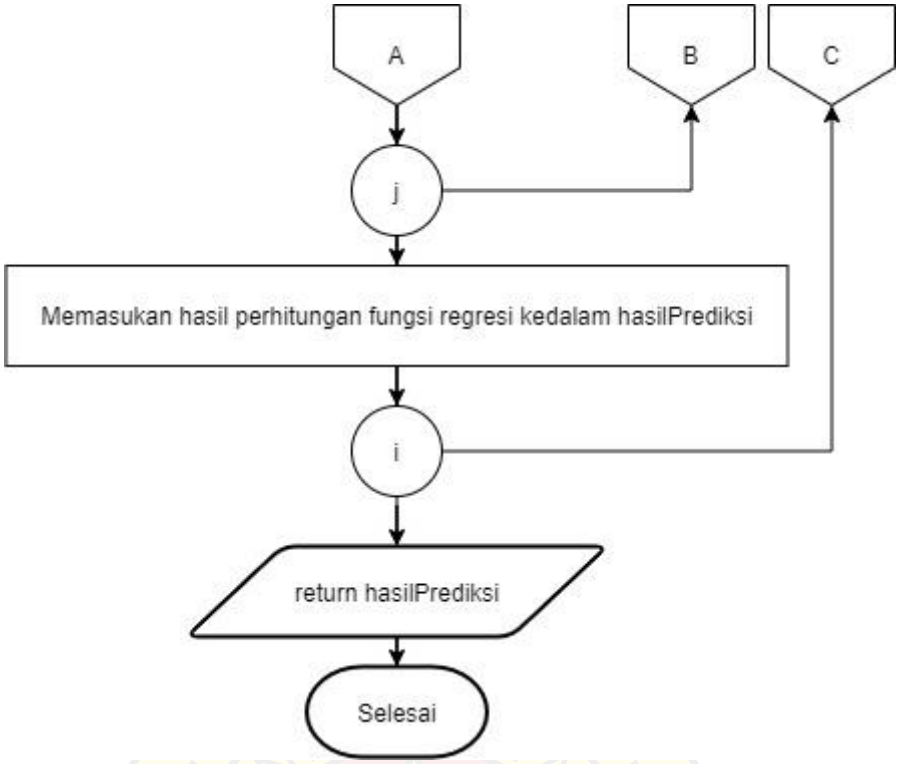
Gambar 8.28 Diagram Alir Menghitung Nilai *Lagrange Multiplier*

Penjelasan dari diagram alir menghitung nilai *Lagrange Multiplier* pada Gambar 3.28 untuk setiap langkah-langkahnya adalah sebagai berikut.

6. Memasukan parameter sLearning, alfa, *gamma*, *epsilon*, dan *complexity*.
7. Membuat array deltaAlfa untuk menyimpan hasil perhitungan nilai *Lagrange Multiplier*.
8. Melakukan perulangan i sesuai dengan panjang baris pada deltaAlfa.
9. Melakukan perhitungan untuk mengupdate nilai *Lagrange Multiplier*.
10. Mengembalikan nilai deltaAlfa sebagai hasil dari perhitungan nilai *Lagrange Multiplier*.

8.2.5.5 Alir Menghitung Fungsi Regresi





Gambar 8.29 Diagram Alir Menghitung Fungsi Regresi

Penjelasan dari diagram alir menghitung fungsi regresi pada Gambar 3.29 untuk setiap langkah-langkahnya adalah sebagai berikut.

8. Memasukan parameter rbf hasil perhitungan matriks hessian, dan alfa.
9. Membuat array hasilPrediksi untuk menyimpan hasil perhitungan fungsi regersi.
10. Melakukan perulangan i sesuai dengan panjang baris pada hasilPrediksi.
11. Membuat variable temp untuk menyimpan perhitungan fungsi regresi sementara.
12. Melakukan perulangan j sesuai dengan panjang kolom pada hasilPrediksi.
13. untuk menghasilkan nilai prediksi menggunakan matriks hessian dan nilai *Lagrange Multiplier* hasil iterasi.

14. Mengembalikan nilai hasilPrediksi sebagai hasil dari perhitungan fungsi regresi.

8.2.6 Perhitungan Manualisasi Algoritme SVR

Perhitungan manualisasi merupakan perhitungan manual yang mana hasil perhitungan tersebut digunakan sebagai acuan untuk menilai kebenaran dari sebuah kode program. Untuk melakukan perhitungan manualisasi dibutuhkan alat bantu berupa Microsoft Excel dan data sample dari jumlah populasi. Berikut merupakan data yang digunakan untuk perhitungan manualisasi ditunjukan pada Tabel 3.26.

Tabel 8.26 Data Harga Cabai Rawit

No	Tanggal	Harga Cabai Rawit
1	28/03/2018	57500
2	29/03/2018	55000
3	02/04/2018	60000
4	03/04/2018	52500
5	04/04/2018	47500
6	05/04/2018	47500

8.2.6.1 Inisialisasi parameter algoritme SVR

Parameter dari algoritme SVR meliputi nilai *sigma*, *lambda*, *epsilon*, kompleksitas, *coefisien Learning Rate* (cLR), dan iterasi. Nilai pada parameter algoritme SVR yang digunakan pada perhitungan manualiasasi adalah 1,5 untuk nilai *sigma*, 5 untuk nilai *lambda*, 0,000001 untuk nilai *epsilon*, 3 untuk nilai kompleksitas, 0,01 untuk nilai cLR, dan 10 untuk banyaknya iterasi. Fitur merepresentasikan bagian dari data target sehingga fitur merupakan suatu nilai yang membentuk data target. Fitur yang digunakan pada perhitungan manualisasi

mengikuti jumlah datanya. Jika jumlah seluruh data adalah 10 maka fitur yang akan terbentuk adalah 5 fitur. Berikut adalah tabel data fitur ditunjukan pada Tabel 3.27.

Tabel 8.27 Data Fitur

Data Ke-	Fitur 1	Fitur 2	Fitur 3	Y
1	57500	55000	60000	52500
2	55000	60000	52500	47500
3	60000	52500	47500	47500

8.2.6.2 Perhitungan Normalisasi Data

Perhitungan yang digunakan pada normalisasi data adalah perhitungan yang digunakan manualisasi sebelumnya, yaitu *Min-Max Normalization*. Perhitungan *Min-Max Normalization* membutuhkan nilai minimum dan nilai maksimum. Berikut adalah tabel nilai minimum dan nilai maksimum dari data domain setelah dilakukan penambahan ditunjukan pada Tabel 3.28.

Tabel 8.28 Nilai Minimum dan Maksimum

	Harga Cabai Rawit
Minimum	45000
Maksimum	62500
Penambahan	2500

Langkah selanjutnya adalah menerapkan perhitungan *Min-Max Normalization* sesuai dengan Persamaan 2.2. Berikut adalah normalisasi data menggunakan perhitungan *Min-Max Normalization*.

$$X_{(1,1)} = \frac{(57500 - 45000)}{(62500 - 45000)}$$

$$X_{(1,1)} = \frac{(12500)}{(17500)}$$

$$X_{(1,1)} = 0{,}7142857$$

Tabel hasil perhitungan *Min-Max Normalization* ditunjukan pada Tabel 3.29.

Tabel 8.29 Normalisasi Data

Data Ke-	Fitur 1	Fitur 2	Fitur 3	Y
1	0,7142857	0,5714286	0,8571429	0,4285714
2	0,5714286	0,8571429	0,4285714	0,1428571
3	0,8571429	0,4285714	0,1428571	0,1428571

8.2.6.3 Perhitungan Matriks Hessian

Perhitungan matriks hessian membutuhkan nilai *lambda* dan *sigma* karena menggunakan fungsi kernel *gaussian rbf*. Pada perhitungan fungsi kernel *gaussian rbf* terdapat perhitungan jarak menggunakan rumus *manhattan*. Berikut merupakan perhitungan jarak *manhattan* untuk baris pertama dan kolom pertama.

$$||x_1 - x_2||^2 = ((0{,}7142857 - 0{,}5714286)^2 + (0{,}5714286 - 0{,}8571429)^2 + (0{,}8571429 - 0{,}4285714)^2)$$

$$||x_1 - x_2||^2 = ((0{,}1428571)^2 + (-0{,}2857143)^2 + (0{,}4285714)^2)$$

$$\left||x_1 - x_2|\right|^2 = (0{,}020408 + 0{,}0816327 + 0{,}183673)$$

$$||x_1 - x_2||^2 = 0{,}2857143$$

Tabel hasil perhitungan jarak ditunjukan pada Tabel 3.30.

Tabel 8.30 Hasil Perhitungan Jarak

	1	2	3
1	0	0,2857143	0,5510204
2	0,2857143	0	0,3469388
3	0,5510204	0,3469388	0

Langkah selanjutnya adalah menghitung matriks hessian menggunakan fungsi kernel *gaussian rbf* dan nilai *lambda*. Berikut adalah perhitungan matriks hessian untuk baris pertama dan kolom pertama.

$$R_{ij} = exp\left(-\frac{0}{2\ x\ 1{,}5^2}\right) + 5^2$$

$$R_{ij} = exp(0) + 25$$

$$R_{ij} = 1 + \ 25$$

$$R_{ij} = 26$$

Tabel hasil perhitungan matriks hessian ditunjukan pada Tabel 3.31.

Tabel 8.31 Hasil Perhitungan Matriks Hessian

	1	2	3
1	0	0,2857143	0,5510204
2	0,2857143	0	0,3469388
3	0,5510204	0,3469388	0

8.2.6.4 Perhitungan Nilai Error

Perhitungan nilai error membutuhkan nilai *lagrange multiplier*, data target hasil normalisasi, dan matriks hessian. Langkah pertama pada perhitungan nilai

error adalah inisialisasi nilai alfa. Berikut adalah table inisialisasi nilai awal *lagrange multiplier* ditunjukan pada table 3.32.

Tabel 8.32 Inisialisasi Nilai Awal *Lagrange Multiplier*

	alfa1	alfa2
1	0	0
2	0	0
3	0	0

Langkah selanjutnya adalah menghitung nilai *error*. Berikut adalah perhitungan nilai error untuk baris pertama dan kolom pertama.

$$E_i = 0{,}4285714 - \left(\left((0-0)\ x\ 26\right) + \left((0-0)\ x\ 25{,}938482\right) + ((0-0)\ x\ 25{,}884751)\right)$$

$$E_i = 0{,}4285714 - \left(\left((0-0)\ x\ 26\right) + \left((0-0)\ x\ 25{,}938482\right) + \left((0-0)\ x\ 25{,}884751\right)\right)$$

$$E_i = 0{,}4285714 - \left((0) + (0) + (0)\right)$$

$$E_i = 0{,}4285714$$

Tabel hasil perhitungan nilai *error* ditunjukan pada Tabel 3.33.

Tabel 8.33 Hasil Perhitungan Nilai *Error*

	1
1	0,428571
2	0,142857
3	0,142857

8.2.6.5 Perhitungan Nilai *Lagrange Multiplier*

Perhitungan nilai *lagrange multiplier* membutuhkan nilai awal *lagrange multiplier*, nilai *gamma*, *epsilon*, dan kompleksitas. Langkah pertama pada perhitungan nilai *lagrange multiplier* adalah menentukan nilai *gamma* menggunakan nilai cLR dan matriks hessian. Berikut adalah perhitungan untuk menentukan nilai *gamma*.

$$\Upsilon = \frac{cLR}{\max(Matriks\ Hessian)}$$

$$\Upsilon = \frac{0{,}01}{26}$$

$$\Upsilon = 0{,}0003846$$

Langkah kedua adalah menghitung perubahan nilai *lagrange multiplier*. Berikut adalah perhitungan *lagrange multiplier* untuk baris pertama dan kolom pertama.

$$\delta\alpha_i^* = min\{\max[0{,}0003846(0{,}428571 - 0{,}000001), -0], 3 - 0\}$$

$$\delta\alpha_i^* = min\{\max[0{,}0001648, -0], 3\}$$

$$\delta\alpha_i^* = min\{0{,}0001648, 3\}$$

$$\delta\alpha_i^* = 0{,}0001648$$

$$\delta\alpha_i = min\{\max[0{,}0003846(-0{,}428571 - 0{,}000001), -0], 3 - 0\}$$

$$\delta\alpha_i = min\{\max[-0{,}0001648, -0], 3\}$$

$$\delta\alpha_i = min\{0, 3\}$$

$$\delta\alpha_i = 0$$

Langkah ketiga adalah mengubah nilai *lagrange multiplier* menggunakan hasil perhitungan sebelumnya. Berikut adalah perubahan nilai *lagrange multiplier* untuk baris pertama dan kolom pertama.

$\delta\alpha_i^* = 0 + 0{,}0001648$

$\delta\alpha_i = 0 + 0$

Tabel hasil perhitungan nilai *lagrange multiplier* pada iterasi ke-1 ditunjukan pada Tabel 3.34.

Tabel 8.34 Hasil Perhitungan Nilai *Lagrange Multiplier* Iterasi ke-1

	alfa1	alfa2
1	0,0001648	0
2	0,0000549	0
3	0,0000549	0

Tabel hasil perhitungan nilai *lagrange multiplier* pada iterasi ke-10 ditunjukan pada Tabel 3.35.

Tabel 8.35 Hasil Perhitungan Nilai *Lagrange Multiplier* Iterasi ke-10

	alfa1	alfa2
1	0,0015342	0
2	0,0004354	0
3	0,0004356	0

8.2.6.6 Perhitungan Fungsi Regresi

Perhitungan fungsi regresi digunakan untuk menghasilkan nilai prediksi. Perhitungan ini membutuhkan nilai *lagrange multiplier* setelah dilakukan iterasi

dan matriks hessian. Berikut adalah perhitungan fungsi regresi untuk baris pertama dan kolom pertama.

$$f(x) = \big((0{,}0015342 - 0)\ x\ 26\big) + \big((0{,}0004354 - 0)\ x\ 25{,}938482\big) + \big((0{,}0004356 - 0)\ x\ 25{,}884751\big)$$

$$f(x) = (0{,}0398902) + (0{,}0112941) + (0{,}0112745)$$

$$f(x) = 0{,}0624589$$

Tabel hasil perhitungan fungsi regresi ditunjukan pada Tabel 3.36.

Tabel 8.36 Hasil Perhitungan Fungsi Regresi

	1
1	0,0624589
2	0,0624092
3	0,0623267

8.2.6.7 Perhitungan Denormalisasi Data

Perhitungan denormalisasi data disesuaikan dengan Persamaan 2.3. Berikut adalah perhitungan nilai denormalisasi data untuk baris pertama dan kolom pertama.

$$X_{(1,1)} = 0{,}0624589.\ (62500 - 45000) + 45000$$

$$X_{(1,1)} = 0{,}0624589.\ (17500) + 45000$$

$$X_{(1,1)} = 1093{,}03 + 45000$$

$$X_{(1,1)} = 46093{,}03$$

Tabel hasil perhitungan nilai denormalisasi data ditunjukan pada Tabel 3.37.

Tabel 8.37 Hasil Perhitungan Denormalisasi

	1
1	46093,03
2	46092,16
3	46090,72

8.2.6.8 Perhitungan MAPE

Perhitungan rata-rata nilai MAPE disesuaikan dengan Persamaan 2.16. Berikut adalah perhitungan rata-rata nilai MAPE dengan menggunakan data target dan data hasil denormalisasi atau data prediksi.

$$MAPE = \frac{1}{3}x\left(\left(\left|\frac{46093{,}03 - 52500}{52500}\right| x\ 100\right) + \left(\left|\frac{46092{,}16 - 47500}{47500}\right| x\ 100\right) + \left(\left|\frac{46090{,}72 - 47500}{47500}\right| x\ 100\right)\right)$$

$$MAPE = \frac{1}{3}x(12{,}2037518 + 2{,}9638731 + 2{,}9669098)$$

$$MAPE = \frac{1}{3}x\ 18{,}1345347$$

$$MAPE = 6{,}0448449$$

Tabel perbandingan data prediksi dan data target ditunjukan pada bagian Tabel 3.38.

Tabel 8.38 Perbandingan Data Prediksi Dan Data Target

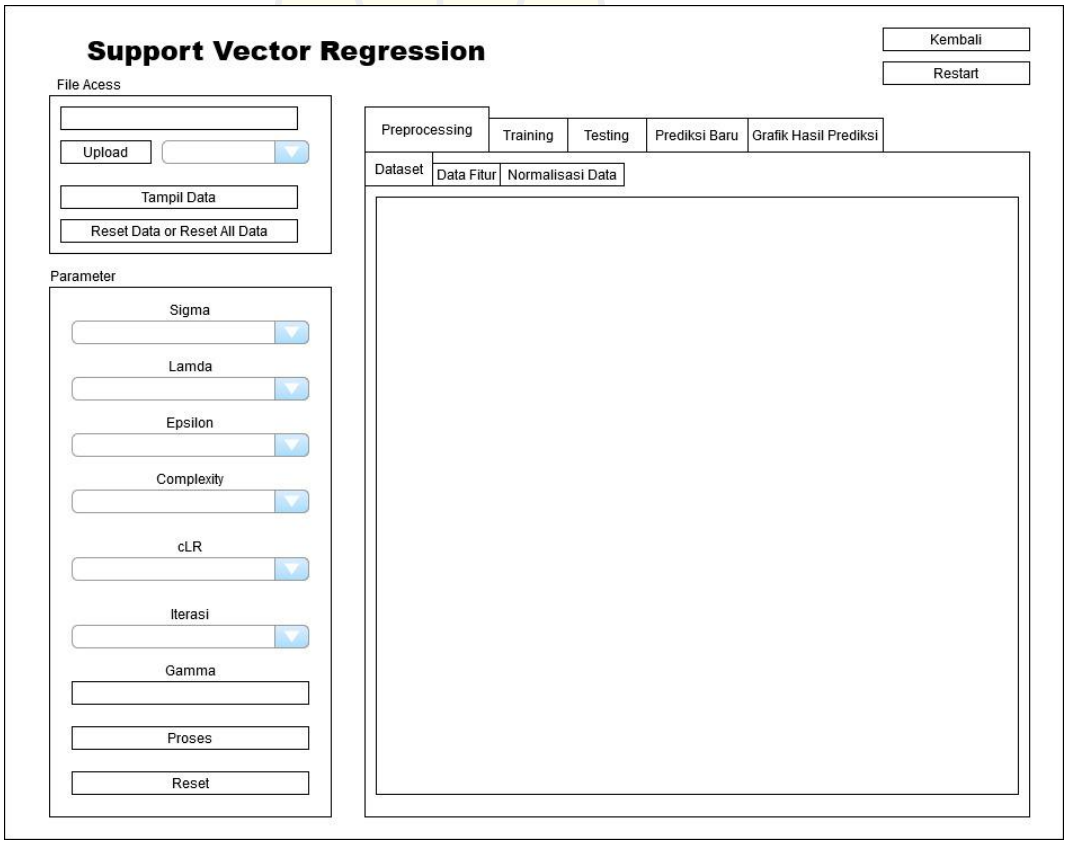
	Data Prediksi	Data Target
1	46093,03	52500
2	46092,16	47500

3	46090,72	47500

8.2.7 Perancangan Antarmuka Algoritme SVR

Perancangan antarmuka bertujuan untuk membuat rancangan mengenai bagian-bagian akan ditampilkan pada sistem. Sesuai dengan perancangan yang telah ditentukan terdapat 4 bagian yang akan ditampilkan, yaitu preprocessing, proses *training*, proses *testing*, proses prediksi, dan hasil prediksi.

8.2.7.1 Perancangan Antarmuka *Preprocessing*



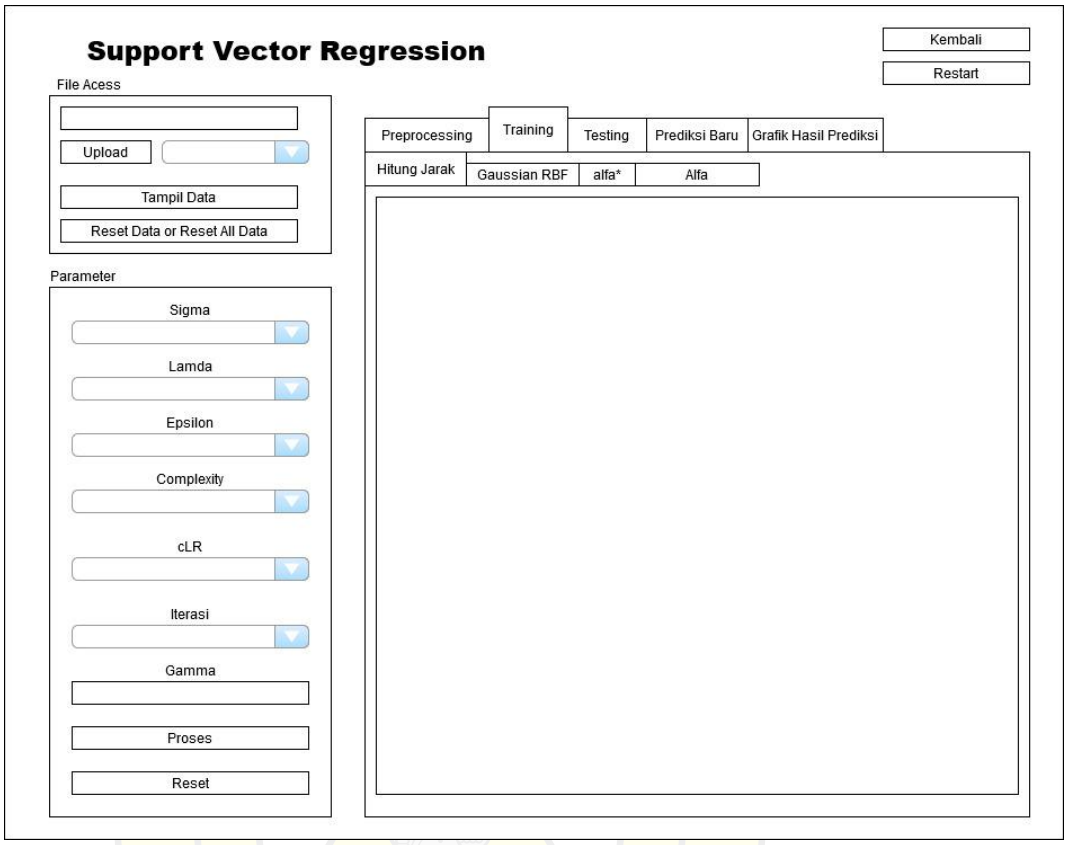
Gambar 8.30 Perancangan Antarmuka *Preprocessing*

Keterangan :

1. Nama metode yang digunakan pada penelitian.
2. Read data textbox digunakan untuk memberikan informasi mengenai lokasi dari file yang dipilih.
3. Button Upload digunakan untuk memilih data excel yang akan digunakan.
4. Input combobox digunakan untuk memilih sheet yang akan dibaca.
5. Button Tampilkan digunakan untuk menampilkan data excel kedalam sistem.
6. Button Reset Data or Reset All Data digunakan untuk mereset seluruh data pada sistem.
7. Input combobox digunakan untuk memilih nilai *sigma*.
8. Input combobox digunakan untuk memilih nilai *lambda*.
9. Input combobox digunakan untuk memilih nilai *epsilon*.
10. Input combobox digunakan untuk memilih nilai kompleksitas.
11. Input combobox digunakan untuk memilih nilai *coefisien Learning Rate (cLR)*.
12. Input combobox digunakan untuk memilih jumlah iterasi.
13. Read data textbox digunakan untuk menampilkan hasil perhitungan nilai gamma.
14. Button Proses digunakan untuk melakukan proses perhitungan algoritme.
15. Button Reset digunakan untuk mereset nilai parameter.
16. Tab Preprocessing digunakan untuk menampung seluruh tahap yang terdapat pada proses preprocessing.
17. Tab Dataset digunakan untuk menampilkan data excel.

18. Tab Data Fitur digunakan untuk menampilkan hasil pembuatan fitur menggunakan dataset.
19. Tab Normalisasi Data digunakan untuk menampilkan hasil perhitungan normalisasi menggunakan data fitur.
20. Tab Bobot digunakan untuk menampilkan hasil pembuatan nilai bobot menggunakan fungsi Random.
21. Tab Bias digunakan untuk menampilkan hasil pembuatan nilai bias menggunakan fungsi Random.
22. Button Kembali digunakan untuk menampilkan form sebelumnya.
23. Button Restart digunakan untuk mematikan sistem dan menjalankannya kembali.

8.2.7.2 Perancangan Antarmuka Proses *Training*



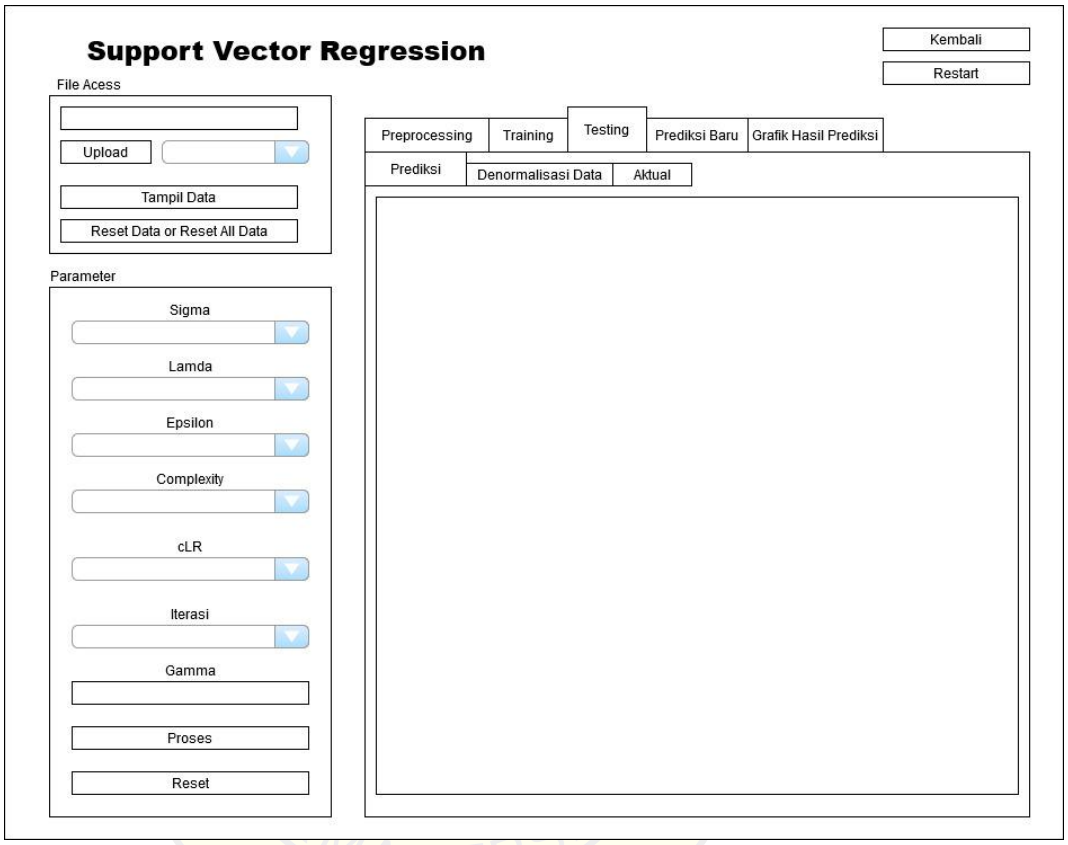
Gambar 8.31 Perancangan Antarmuka Proses *Training*

Keterangan :

1. Tab Training digunakan untuk menampung seluruh tahap yang terdapat pada proses *training*.
2. Tab Hitung Jarak digunakan untuk menampilkan hasil perhitungan jarak pada setiap baris data menggunakan rumus *manhattan*.
3. Tab Gaussian RBF digunakan untuk menampilkan hasil perhitungan matriks hessian menggunakan fungsi kernel *gaussian rbf*.
4. Tab alfa* digunakan untuk menampilkan hasil pehitungan nilai *lagrange multiplier* pertama setelah dilakukan iterasi.

5. Tab alfa digunakan untuk menampilkan hasil pehitungan nilai *lagrange multiplier* kedua setelah dilakukan iterasi.

8.2.7.3 Perancangan Antarmuka Proses *Testing*



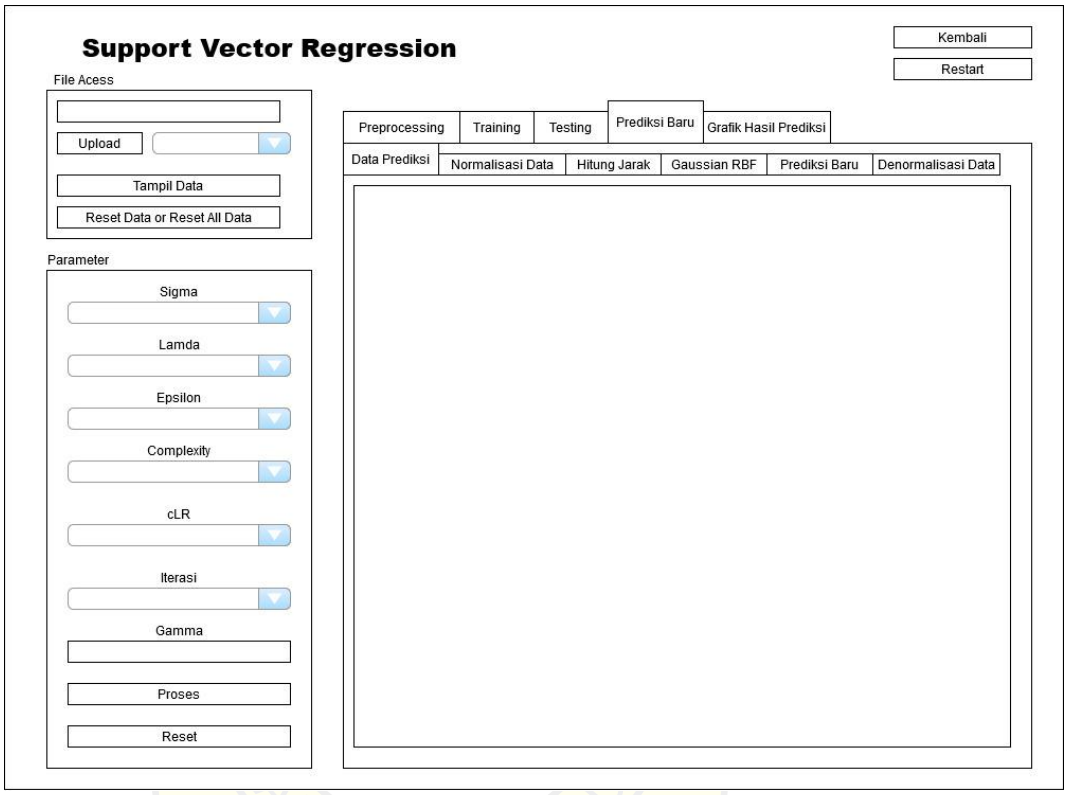
Gambar 8.32 Perancangan Antarmuka Proses *Testing*

Keterangan :

1. Tab Testing digunakan untuk menampung seluruh tahap yang terdapat pada proses *testing*.
2. Tab Prediksi digunakan untuk menampilkan hasil prediksi menggunakan perhitungan fungsi regresi.
3. Tab Denormalisasi Data digunakan untuk menampilkan hasil pengembalian data prediksi kedalam bentuk sebenarnya.

4. Tab Aktual digunakan untuk menampilkan data aktual untuk proses *testing* menggunakan data fitur.

8.2.7.4 Perancangan Antarmuka Proses Prediksi



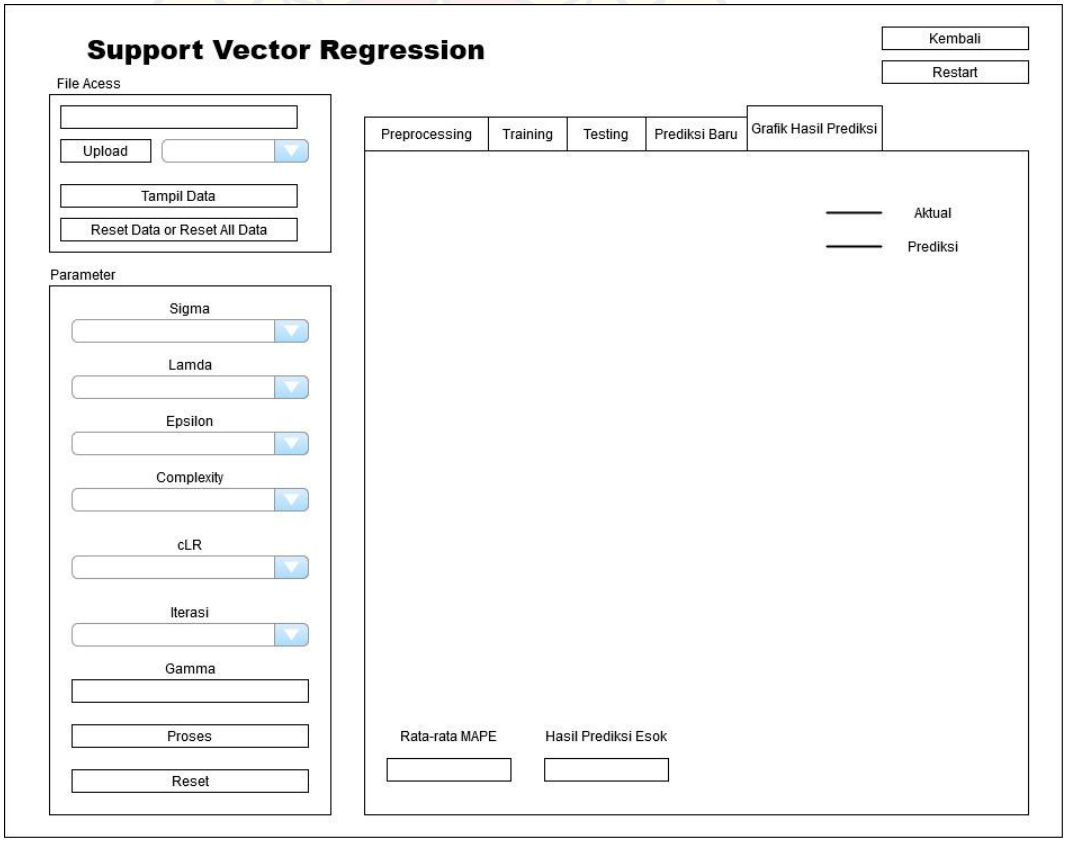
Gambar 8.33 Perancangan Antarmuka Proses Prediksi

Keterangan :

1. Tab Prediksi Baru digunakan untuk menampung seluruh tahap yang terdapat pada proses prediksi.
2. Tab Data Prediksi digunakan untuk menampilkan hasil pengambilan data prediksi menggunakan data fitur
3. Tab Normalisasi Data digunakan untuk menampilkan hasil perhitungan normalisasi data pada data prediksi.

4. Tab Hitung Jarak digunakan untuk menampilkan hasil perhitungan jarak pada setiap baris data menggunakan rumus *manhattan*.
5. Tab Gaussian RBF digunakan untuk menampilkan hasil perhitungan matriks hessian menggunakan fungsi kernel *gaussian rbf*.
6. Tab Prediksi Baru digunakan untuk menampilkan hasil prediksi menggunakan perhitungan fungsi regresi.
7. Tab Denormalisasi Data digunakan untuk menampilkan hasil pengembalian data prediksi kedalam bentuk sebenarnya

8.2.7.5 Perancangan Antarmuka Hasil Prediksi



Gambar 8.34 Perancangan Antarmuka Hasil Prediksi

Keterangan :

1. Tab Grafik Hasil Prediksi digunakan untuk menampung seluruh bagian yang terdapat pada hasil prediksi.
2. Tampilan Grafik digunakan untuk menampilkan perbandingan data target dengan data prediksi
3. Read data textbox digunakan untuk menampilkan akurasi tingkat kesalahan menggunakan perhitungan mape.
4. Read data textbox digunakan untuk menampilkan hasil prediksi untuk satu hari kedepan.

8.2.8 Perancangan Pengujian Algoritme SVR

Pengujian parameter pada algoritme SVR ditunjukan untuk mengetahui parameter terbaik yang menghasilkan tingkat kesalahan terendah dengan menggunakan satuan rata-rata nilai MAPE. Berikut adalah urutan untuk melakukan pengujian parameter pada algoritme SVR.

1. Pengujian Nilai *Lambda*
2. Pengujian Nilai *Sigma*
3. Pengujian Nilai *cLR*
4. Pengujian Nilai Kompleksitas
5. Pengujian Nilai *Epsilon*
6. Pengujian Jumlah Iterasi

8.2.8.2 Pengujian Nilai *Lambda*

Nilai *lambda* yang digunakan mengacu pada penelitian yang dilakukan oleh Averina, Santoso, & Yudistira tahun 2020 yang mana pengujian untuk parameter nilai *lambda* dimulai dari 5,5 sampai 10. Berikut adalah tabel perancangan pengujian nilai *lambda* ditunjukan pada Tabel 3.39.

Tabel 8.39 Perancangan Pengujian Nilai *Lambda*

Nilai Lambda	Nilai MAPE
5,5	
6	
6,5	
7	
7,5	
8	
8,5	
9	
9,5	
10	

8.2.8.3 Pengujian Nilai *Sigma*

Nilai *sigma* yang digunakan mengacu pada penelitian yang dilakukan oleh Averina, Santoso, & Yudistira tahun 2020 yang mana pengujian untuk parameter nilai *sigma* dimulai dari 0,001 sampai 2,5. Berikut adalah tabel perancangan pengujian nilai *sigma* ditunjukan pada Tabel 3.40.

Tabel 8.40 Perancangan Pengujian Nilai *Sigma*

Nilai Sigma	Nilai MAPE
0,001	
0,005	

0,01	
0,05	
0,1	
0,5	
1	
1,5	
2	
2,5	

8.2.8.4 Pengujian Nilai *cLR*

Nilai *cLR* yang digunakan mengacu pada penelitian yang dilakukan oleh Averina, Santoso, & Yudistira tahun 2020 yang mana pengujian untuk parameter nilai *cLR* dimulai dari 0,00001 sampai 0,01. Berikut adalah tabel perancangan pengujian nilai *cLR* ditunjukan pada Tabel 3.41.

Tabel 8.41 Perancangan Pengujian Nilai *cLR*

Nilai cLR	Nilai MAPE
0,00001	
0,00003	
0,00005	
0,0001	
0,0003	
0,0005	
0,001	
0,003	
0,005	
0,01	

8.2.8.5 Pengujian Nilai Kompleksitas

Nilai kompleksitas yang digunakan mengacu pada penelitian yang dilakukan oleh Averina, Santoso, & Yudistira tahun 2020 yang mana pengujian untuk parameter nilai kompleksitas dimulai dari 0,0005 sampai 100. Berikut adalah tabel perancangan pengujian nilai kompleksitas ditunjukan pada Tabel 3.42.

Tabel 8.42 Perancangan Pengujian Nilai Kompleksitas

Nilai Komplesitas	Nilai MAPE
0,0005	
0,001	
0,005	
0,01	
0,05	
0,1	
0,5	
1	
50	
100	

8.2.8.6 Pengujian Nilai *Epsilon*

Nilai *epsilon* yang digunakan mengacu pada penelitian yang dilakukan oleh Averina, Santoso, & Yudistira tahun 2020 yang mana pengujian untuk parameter nilai *epsilon* dimulai dari 0,000005 sampai 0,1. Berikut adalah tabel perancangan pengujian nilai *epsilon* ditunjukan pada Tabel 3.43.

Tabel 8.43 Perancangan Pengujian Nilai *Epsilon*

Nilai Epsilon	Nilai MAPE
0,000005	
0,00001	
0,00005	
0,0001	
0,0005	
0,001	
0,005	
0,01	
0,05	
0,1	

8.2.8.7 Pengujian Jumlah Iterasi

Jumlah iterasi yang digunakan mengacu pada penelitian yang dilakukan oleh Averina, Santoso, & Yudistira tahun 2020 yang mana pengujian untuk parameter jumlah iterasi dimulai dari 1000 iterasi sampai 5000 iterasi. Berikut adalah tabel perancangan pengujian jumlah iterasi ditunjukan pada Tabel 3.44.

Tabel 8.44 Perancangan Pengujian Jumlah Iterasi

Jumlah Iterasi	Nilai MAPE
1000	
2000	
3000	
4000	
5000	

TEKNOLOGI INFORMASI

UNIVERSITAS DARMA PERSADA

BAB 9

IMPLEMENTASI HASIL

9.1 Implementasi Sistem

Bagian implementasi sistem merupakan penerapan dari perancangan yang telah dibentuk. Implementasi yang dilakukan meliputi implementasi antarmuka untuk setiap algoritme. Implementasi antarmuka pada sistem disesuaikan dengan perancangan sebelumnya.

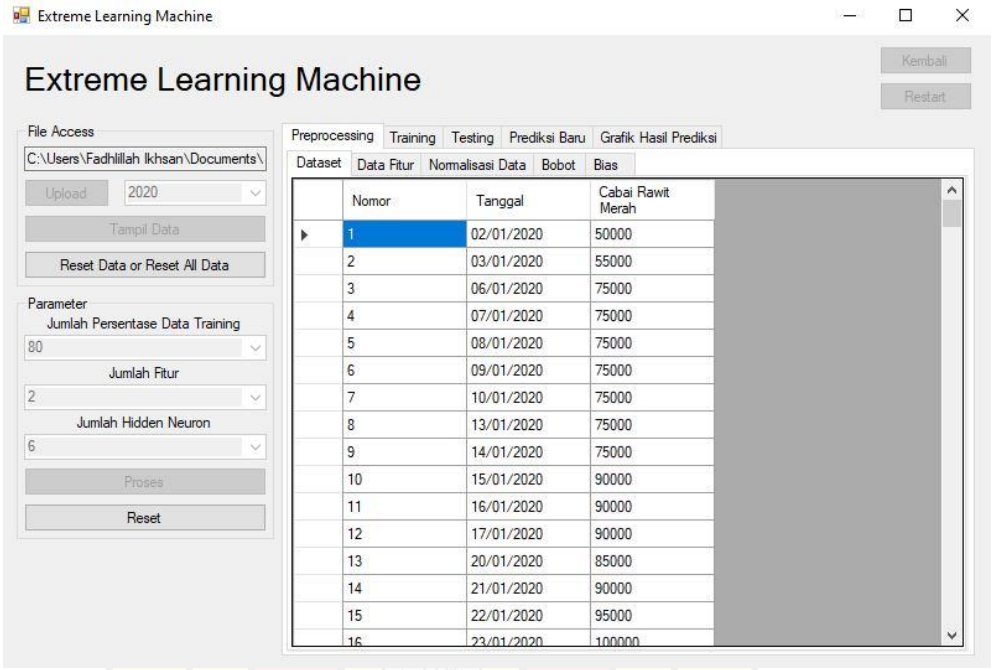
9.1.1 Implementasi Antarmuka *Algoritme Extreme Learning Machine*

Implementasi antarmuka ditampilkan menggunakan aplikasi pemrograman desktop yaitu C#. Pada tampilan sistem terdapat 3 bagian, yaitu file akses, input parameter, dan tampilan data dari hasil perhitungan. Bagian input file digunakan untuk memilih data yang akan dibaca. Bagian input parameter merupakan masukan awal yang akan diberikan kepada algoritme untuk dihitung. Dan bagian tampilan data akan menampilkan data pada setiap proses perhitungan algoritme. Pada implementasi antarmuka, bagian yang akan ditampilkan disesuaikan dengan perancangan yang telah dibentuk. Bagian yang akan ditampilkan pada implementasi antarmuka merupakan tahap yang dilalui pada perhitungan algoritme meliputi tahap preprocessing, training, testing, prediksi, dan grafik hasil prediksi.

9.1.1.1 Tampilan Halaman Preprocessing

Bagian preprocessing akan menampilkan dataset dari pembacaan data excel, kemudian hasil pembentukan data fitur menggunakan dataset, hasil perhitungan

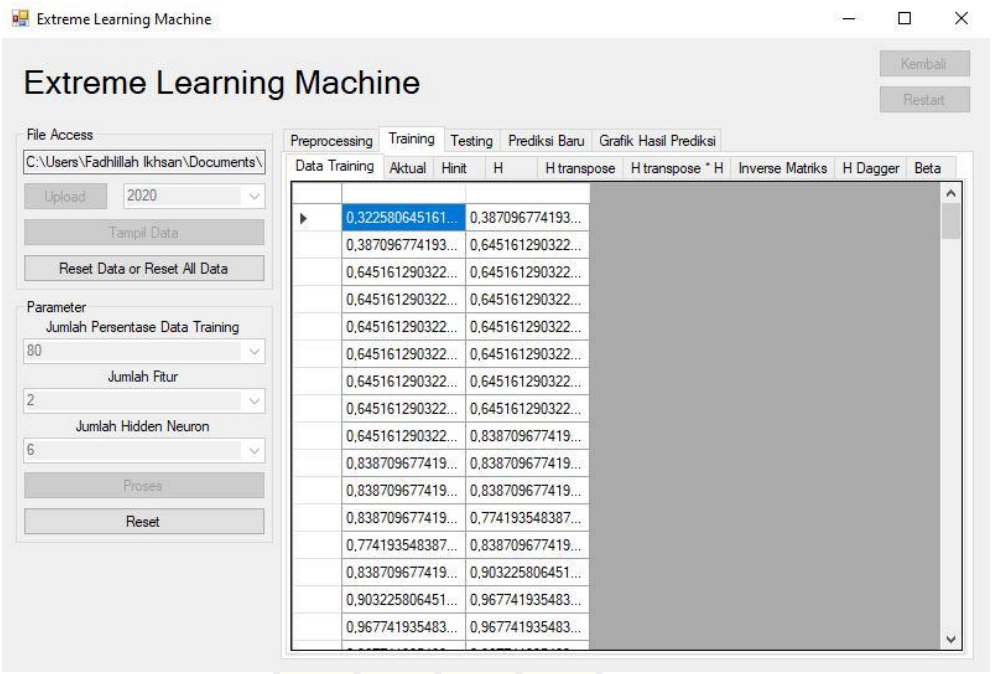
normalisasi data, dan hasil pembuatan bobot dan bias menggunakan nilai random sesuai dengan rentang nilai yang telah ditentukan. Berikut merupakan tampilan pada halaman preprocessing ditunjukan pada Gambar 4.1.



Gambar 9.1 Tampilan Halaman Preprocessing

9.1.1.2 Tampilan Halaman Training

Bagian training akan menampilkan seluruh proses perhitungan yang dilalui pada tahap *training* mulai dari pembagian data fitur dan data target, perhitungan nilai Hinit, perhitungan nilai *hidden layer*, perhitungan *moore-penrose pseudo inverse* untuk menghasilkan nilai H *dagger* melalui perhitungan *inverse* matriks dan *transpose hidden layer*, dan perhitungan nilai *output weight* sebagai proses terakhir pada tahap training. Berikut merupakan tampilan pada halaman training ditunjukan pada Gambar 4.2.



Gambar 9.2 Tampilan Halaman Training

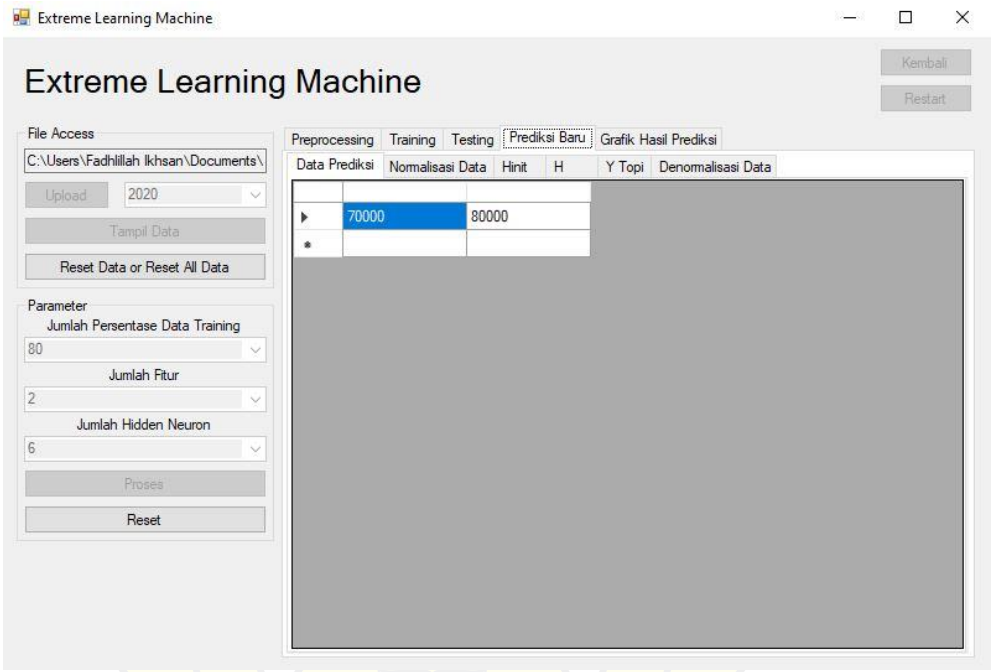
9.1.1.3 Tampilan Halaman Testing

Bagian halaman testing akan menampilkan seluruh proses perhitungan yang dilalui pada tahap *testing* mulai dari pembagian data fitur dan data target untuk proses *testing*, perhitungan nilai Hinit, perhitungan nilai *hidden layer*, dan perhitungan *output layer* untuk menghasilkan nilai prediksi menggunakan nilai *output weight* yang dihasilkan pada proses *training* dan nilai *hidden layer* yang dihasilkan pada proses *testing*. Berikut merupakan tampilan pada halaman testing ditunjukan pada Gambar 4.3.

9.1.1.4 Tampilan Halaman Prediksi Baru

Bagian prediksi baru akan menampilkan seluruh proses perhitungan algoritme mulai dari pengambilan data prediksi, normalisasi data, perhitungan nilai Hinit, perhitungan nilai *hidden layer*, perhitungan *moore-penrose pseudo inverse* untuk

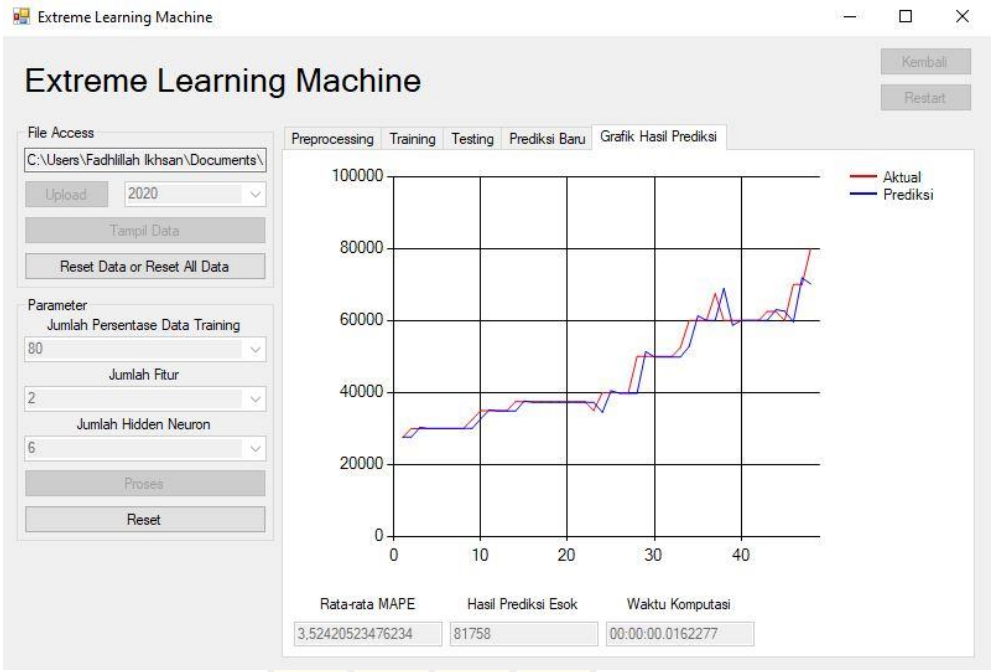
menghasilkan nilai H *dagger* melalui perhitungan *inverse* matriks dan *transpose hidden layer*, dan perhitungan nilai *output weight* sebagai proses terakhir pada tahap training. Berikut merupakan tampilan pada halaman prediksi baru ditunjukan pada Gambar 4.4.



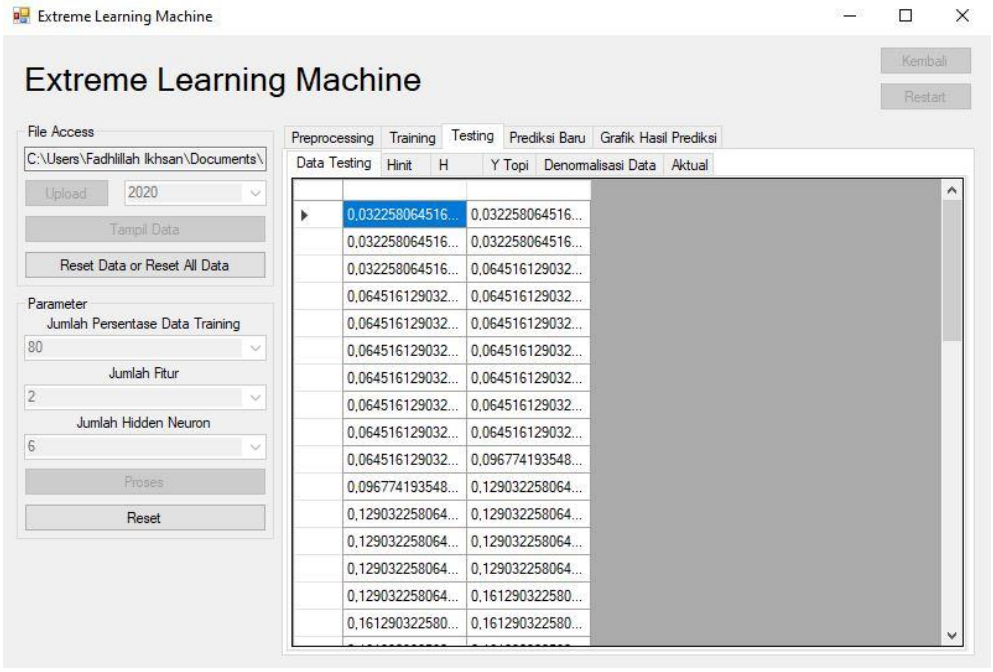
Gambar 9.3 Tampilan Halaman Prediksi Baru

9.1.1.5 Tampilan Halaman Grafik Hasil Prediksi

Bagian grafik hasil prediksi akan menampilkan grafik perbandingan hasil prediksi dengan data target, nilai MAPE dari hasil prediksi sebagai tingkat untuk mengukur seberapa baik algoritme *Extreme Learning Machine* dalam memprediksi data baru, nilai prediksi untuk satu hari kedepan dan waktu komputasi yang dibutuhkan untuk pembelajaran sistem. Berikut merupakan tampilan pada halaman grafik hasil prediksi ditunjukan pada Gambar 4.5.



Gambar 9.4 Tampilan Halaman Grafik Hasil Prediksi



Gambar 9.5 Tampilan Halaman Testing

9.1.2 Implementasi Antarmuka Algoritme *Support Vector Regression*

Implementasi antarmuka ditampilkan menggunakan aplikasi serupa yang digunakan untuk menampilkan halaman pada implementasi antarmuka sebelumnya. Pada tampilan sistem algoritme SVR terdapat 3 bagian yang sama, pada algoritme ELM yaitu file akses, input parameter, dan tampilan data dari hasil perhitungan. Bagian input parameter merupakan masukan awal yang akan diberikan kepada algoritme untuk dihitung. Dan bagian tampilan data akan menampilkan data pada setiap proses perhitungan algoritme. Pada implementasi antarmuka, bagian yang akan ditampilkan disesuaikan dengan perancangan yang telah dibentuk. Bagian yang akan ditampilkan pada implementasi antarmuka merupakan tahap yang dilalui pada perhitungan algoritme meliputi tahap preprocessing, training, testing, prediksi, dan grafik hasil prediksi.

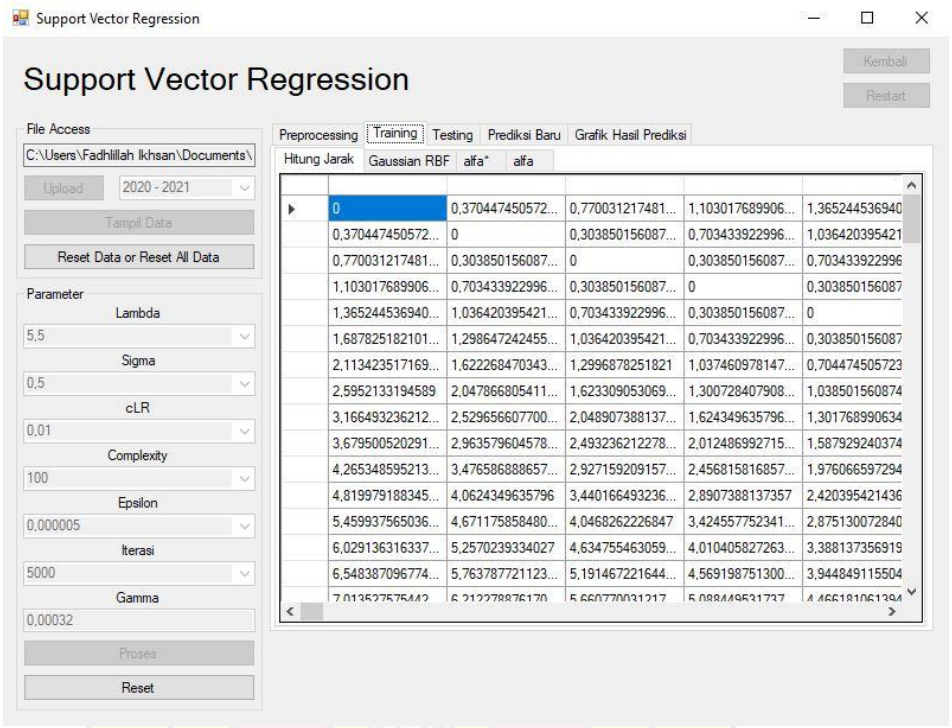
9.1.2.1 Tampilan Halaman Preprocessing

Bagian preprocessing akan menampilkan dataset dari hasil pembacaan data excel, kemudian hasil pembentukan data fitur menggunakan dataset, hasil perhitungan normalisasi data, dan hasil pembuatan bobot dan bias menggunakan nilai random sesuai dengan rentang nilai yang telah ditentukan. Berikut merupakan tampilan pada halaman preprocessing ditunjukan pada Gambar 4.6.

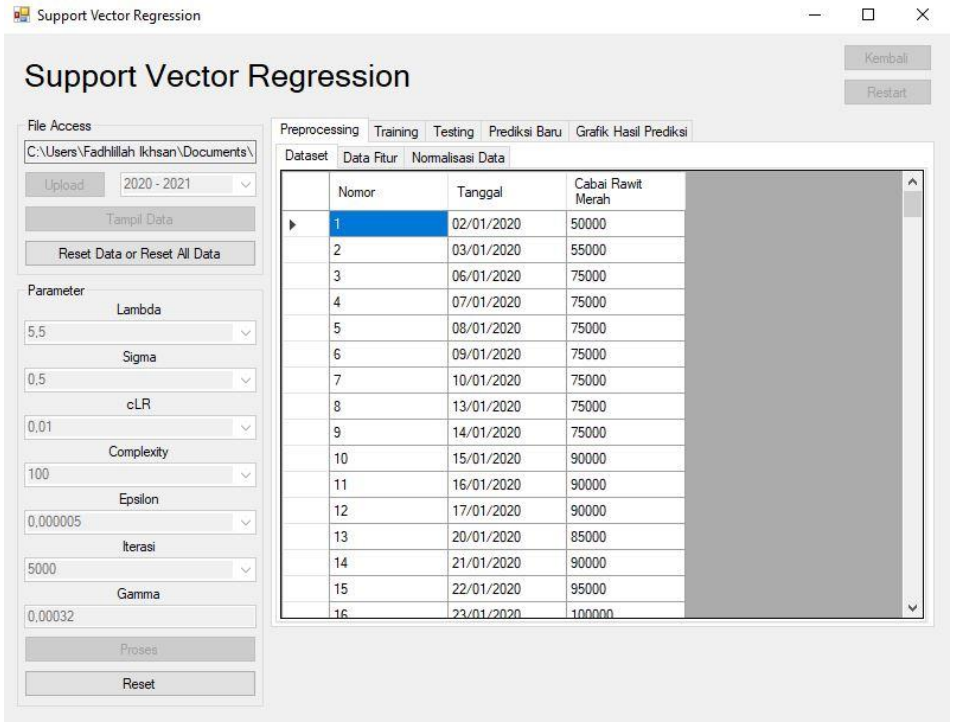
9.1.2.2 Tampilan Halaman Training

Bagian training akan menampilkan hasil perhitungan jarak pada setiap data menggunakan data normalisasi, kemudian hasil perhitungan matriks hessian menggunakan fungsi kernel *gaussian rbf*, dan hasil perubahan nilai *lagrange multiplier* setelah dilakukan iterasi sampai batas iterasi maksimum yang telah

ditentukan. Berikut merupakan tampilan pada halaman *training* ditunjukan pada Gambar 4.7.



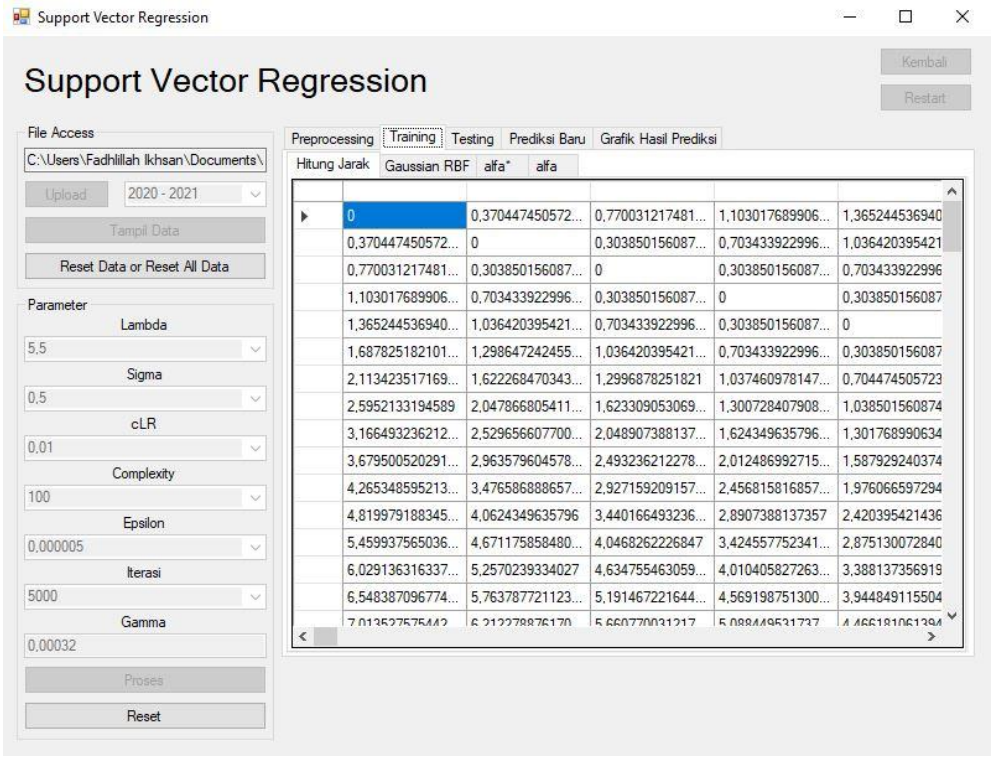
Gambar 9.6 Tampilan Halaman Training



Gambar 9.7 Tampilan Halaman Preprocessing

9.1.2.3 Tampilan Halaman Testing

Bagian halaman testing akan menampilkan seluruh proses perhitungan yang dilalui pada tahap *testing* mulai dari perhitungan prediksi menggunakan fungsi regresi, perhitungan denormalisasi data untuk mengembalikan nilai prediksi kedalam bentuk yang sebenarnya, dan data target untuk membandingkan kedekatan nilai prediksi dengan data yang sebenarnya. Berikut merupakan tampilan pada halaman testing ditunjukan pada Gambar 4.8.

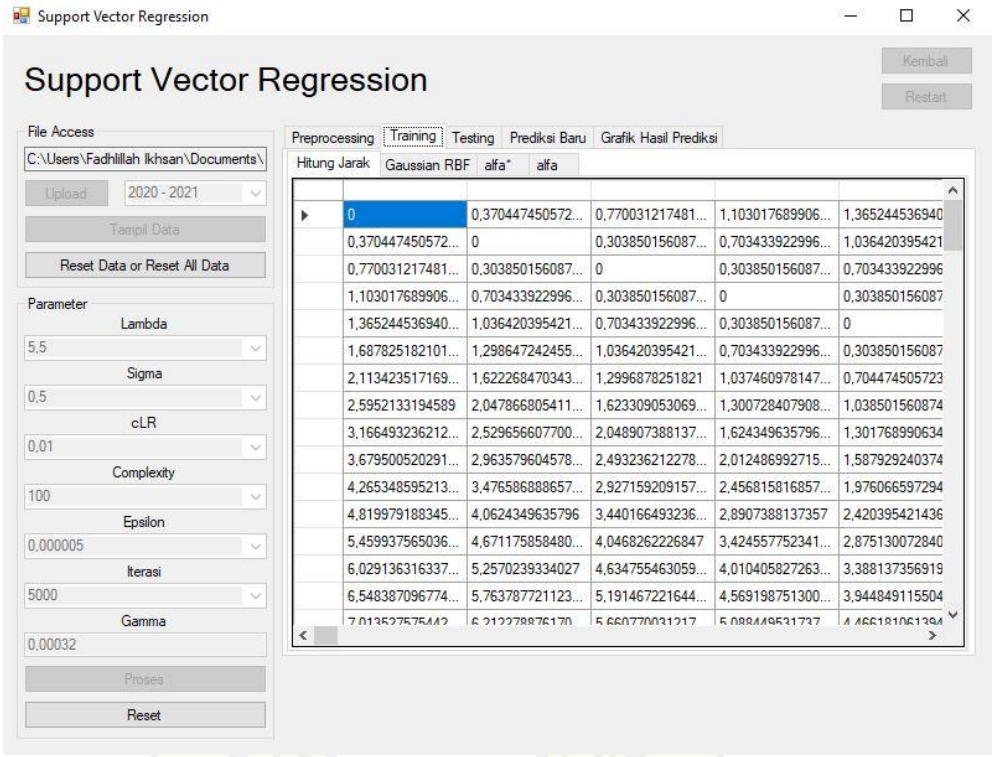


Gambar 9.8 Tampilan Halaman Testing

9.1.2.4 Tampilan Halaman Prediksi Baru

Bagian prediksi baru akan menampilkan seluruh proses perhitungan algoritme mulai dari pengambilan data prediksi, normalisasi data, perhitungan jarak untuk setiap data, perhitungan matriks hessian menggunakan fungsi kernel gaussian rbf,

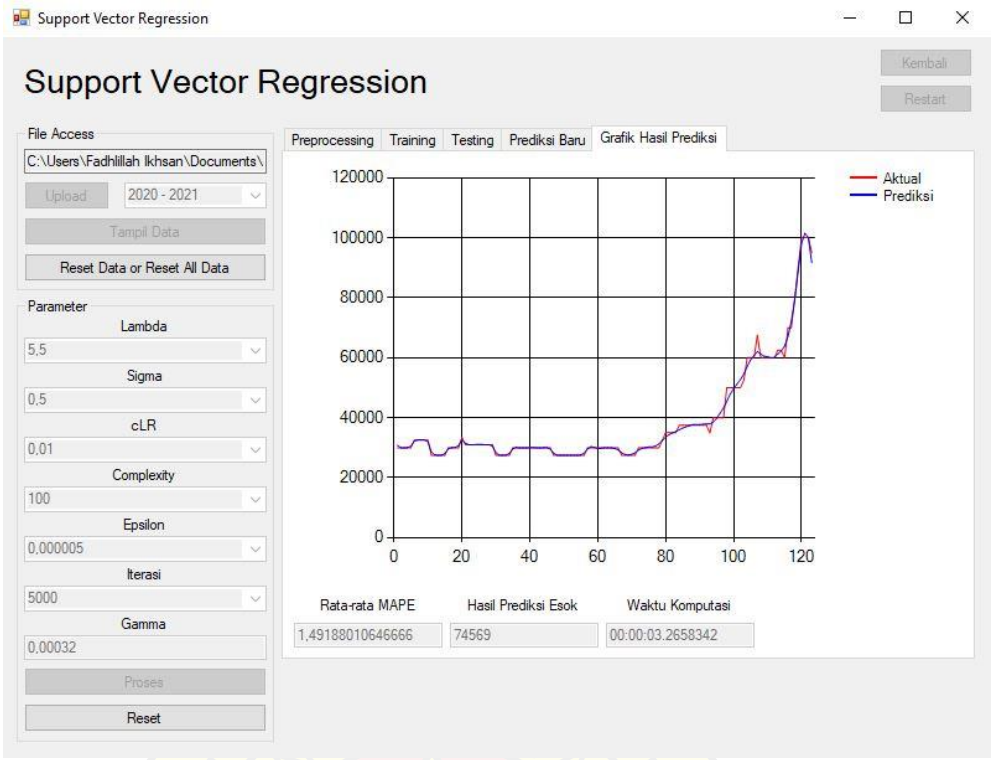
menentukan nilai prediksi menggunakan fungsi regresi, dan mengembalikan hasil prediksi kedalam bentuk yang sebenarnya menggunakan perhitungan denormalisasi data. Berikut merupakan tampilan pada halaman prediksi baru ditunjukan pada Gambar 4.9.



Gambar 9.9 Tampilan Halaman Prediksi Baru

9.1.2.5 Tampilan Halaman Grafik Hasil Prediksi

Bagian grafik hasil prediksi akan menampilkan grafik perbandingan hasil prediksi dengan data target, nilai MAPE dari hasil prediksi sebagai tingkat untuk mengukur seberapa baik algoritme *Support Vector Regression* dalam memprediksi data baru, nilai prediksi untuk satu hari kedepan dan waktu komputasi yang dibutuhkan untuk pembelajaran sistem. Berikut merupakan tampilan pada halaman grafik hasil prediksi ditunjukan pada Gambar 4.10.



Gambar 9.10 Tampilan Halaman Grafik Hasil Prediksi

9.2 Pengujian

Bagian pengujian akan mengevaluasi hasil prediksi yang telah didapatkan menggunakan sebuah perhitungan untuk mengukur keberhasilan sebuah algoritme berdasarkan tingkat kesalahan yang dihasilkan. Pengujian dilakukan sesuai dengan perancangan yang telah dibentuk.

9.2.1 Pengujian Algoritme *Extreme Learning Machine*

Pengujian parameter pada algoritme ELM ditunjukan untuk mengetahui parameter terbaik yang menghasilkan tingkat kesalahan terendah dengan menggunakan satuan rata-rata nilai MAPE. Parameter yang diuji pada algoritme ELM adalah jumlah fitur, jumlah *neuron* dan persentase jumlah data *training*.

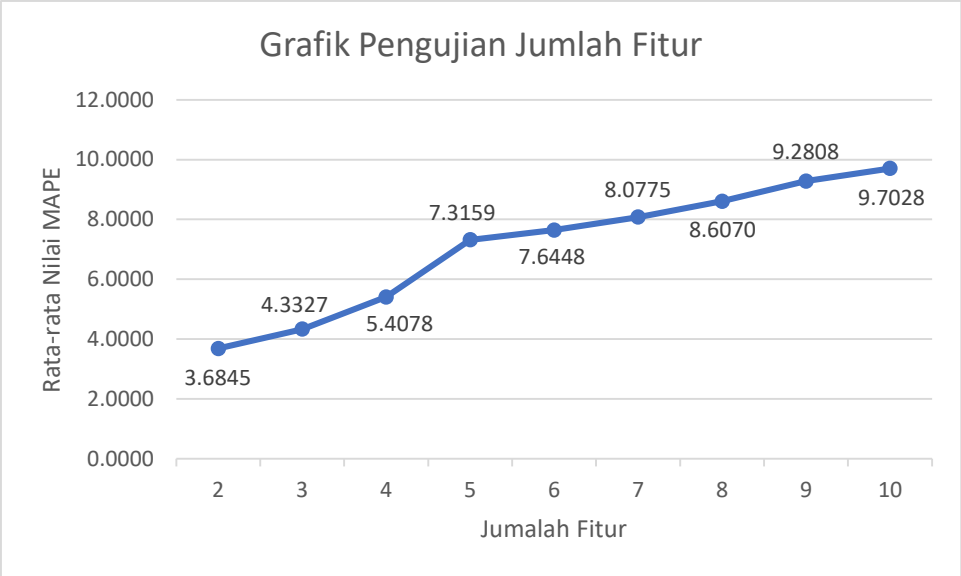
9.2.1.1 Pengujian Jumlah Fitur

Fitur merupakan faktor yang mempengaruhi data target. Semakin banyak fitur yang digunakan, maka faktor pembentuk data target juga semakin banyak. Selain mempengaruhi data target, jumlah fitur yang digunakan juga mempengaruhi jumlah data yang akan terbentuk. Semakin banyak fitur yang digunakan, maka semakin banyak juga data yang akan berkurang. Hal tersebut sangat mempengaruhi proses pembelajaran sehingga dibutuhkan suatu pengujian untuk mendapatkan jumlah fitur terbaik yang menghasilkan tingkat kesalahan terendah. Pada pengujian jumlah fitur, jumlah *neuron* yang digunakan adalah 5 *neuron* dan perbandingan persentase jumlah data *training* dan data *testing* yang digunakan adalah 80%:20%. Tabel hasil pengujian jumlah fitur ditunjukan pada Tabel 4.1.

Hasil perhitungan tabel perlu diubah dalam bentuk diagram garis untuk mendapatkan analisa dari hasil pengujian. Berikut adalah diagram garis dari hasil pengujian jumlah fitur ditunjukan pada Gambar 4.11.

Tabel 9.1 Hasil Pengujian Jumlah Fitur

Jumlah Fitur	MAPE Percobaan ke-					Rata-rata MAPE
	1	2	3	4	5	
2	**3,6413**	**3,5702**	**3,7638**	**3,7564**	**3,6909**	**3,6845**
3	4,5287	4,7835	4,1488	4,4683	3,7343	4,3327
4	5,0596	6,5564	5,1707	4,3108	5,9415	5,4078
5	5,7366	9,5226	7,1078	6,3116	7,9007	7,3159
6	6,5927	8,8906	9,1258	6,0160	7,5990	7,6448
7	7,3283	10,5665	8,0213	5,3528	9,1188	8,0775
8	11,3798	11,2877	8,5036	5,2136	6,6501	8,6070
9	13,5886	11,8661	7,1720	7,9219	5,8554	9,2808
10	9,1649	14,2343	7,9610	7,4511	14,2169	9,7028



Gambar 9.11 Grafik Hasil Pengujian Jumlah Fitur

Berdasarkan hasil pengujian yang telah dilakukan pada Tabel 4.1, jumlah fitur yang menghasilkan rata-rata tingkat kesalahan terendah dari 5 kali percobaan adalah 2 fitur dengan nilai *error* sebesar 3,6845%. Berdasarkan grafik pengujian jumlah fitur pada Gambar 4.11, dapat disimpulkan bahwa semakin bertambahnya jumlah fitur yang digunakan maka nilai *error* yang dihasilkan juga semakin tinggi. Hal tersebut terjadi karena semakin banyak faktor yang digunakan untuk mempengaruhi data target, maka jumlah data yang akan diproses juga akan semakin berkurang. Jumlah data yang digunakan sangat mempengaruhi proses pembelajaran pada sistem, jika jumlah data yang digunakan berkurang makan sistem hanya mendapatkan sedikit pembelajaran untuk menghasilkan prediksi. Hal tersebut membuat nilai prediksi yang dihasilkan memiliki tingkat kesalahan yang lebih tinggi.

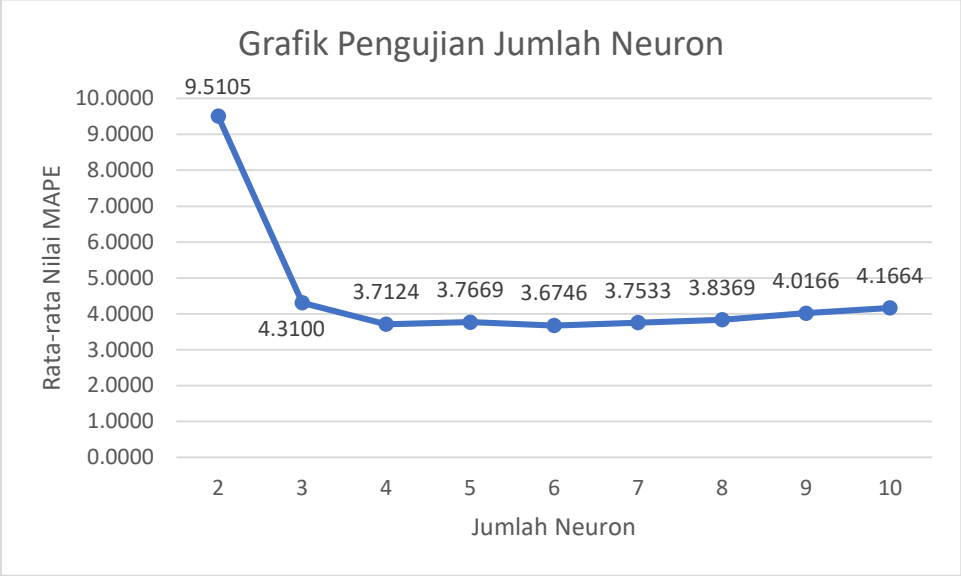
9.2.1.2 Pengujian Jumlah Neuron

Jumlah *neuron* menentukan kompleksitas jaringan pada *hidden layer*. Semakin banyak jumlah *neuron* yang digunakan maka tingkat kompleksitas jaringan pada *hidden layer* akan semakin tinggi. Jaringan yang dibentuk pada *hidden layer* mempengaruhi proses pembelajaran pada sistem sehingga tingkat kerumitan pada proses pembelajaran akan terjadi pada *hidden layer* jika jumlah *neuron* yang digunakan semakin banyak. Jumlah *neuron* yang terlalu banyak akan memicu terjadinya *overfitting*. *Overvitting* akan membuat algoritme tidak dapat mengenali data lain selain data yang dipelajarinya. Untuk itu perlu dilakukan pengujian jumlah *neuron* untuk mendapatkan jumlah *neuron* yang tepat yang menghasilkan tingkat kesalahan terendah. Pada pengujian jumlah *neuron*, jumlah fitur yang digunakan adalah 2 fitur dan persentase jumlah data *training* dan data *testing* yang dugunakan adalah 80%:20%. Berikut adalah table hasil pengujian jumlah *neuron* ditunjukan pada Table 4.2.

Tabel 9.2 Hasil Pengujian Jumlah Neuron

Jumlah Neuron	MAPE Percobaan ke-					Rata-rata MAPE
	1	2	3	4	5	
2	5,2399	6,8980	4,3818	19,8453	11,1874	9,5105
3	3,8331	5,6861	3,7654	4,2475	4,0177	4,3100
4	3,7681	3,5912	3,8553	3,6740	3,6735	3,7124
5	3,7245	3,7645	3,7207	3,8439	3,7809	3,7669
6	**3,6187**	**3,6597**	**3,6963**	**3,6858**	**3,7127**	**3,6746**
7	3,8622	3,6396	3,6933	3,6452	3,9261	3,7533
8	3,6218	3,9947	3,8800	3,7602	3,9280	3,8369
9	3,7918	4,2022	3,8477	4,2583	3,9829	4,0166
10	4,1090	4,0583	4,0467	4,3473	4,2708	4,1664

Hasil perhitungan tabel perlu diubah dalam bentuk diagram garis untuk mendapatkan analisa dari hasil pengujian. Berikut adalah diagram garis dari hasil pengujian jumlah *neuron* ditunjukan pada Gambar 4.12.



Gambar 9.12 Grafik Hasil Pengujian Jumlah Neuron

Berdasarkan hasil pengujian yang telah dilakukan pada Tabel 4.2, jumlah *neuron* yang menghasilkan rata-rata tingkat kesalahan terendah pada 5 kali percobaan adalah 6 *neuron* dengan nilai *error* sebesar 3,6746%. Berdasarkan grafik pengujian jumlah *neuron* pada Gambar 4.12, dapat disimpulkan bahwa semakin bertambahnya jumlah *neuron* yang digunakan maka nilai *error* yang dihasilkan juga semakin tinggi. Namun, jika jumlah *neuron* yang digunakan terlalu sedikit maka nilai *error* yang akan dihasilkan juga semakin tinggi. Hal tersebut terjadi karena kompleksitas jaringan yang terbentuk pada *hidden layer* sangat sederhana sehingga mengakibatkan sistem tidak dapat mempelajari pola data dengan baik dan menghasilkan nilai prediksi dengan tingkat kesalahan yang tinggi.

9.2.1.3 Pengujian Persentase Jumlah Data Training

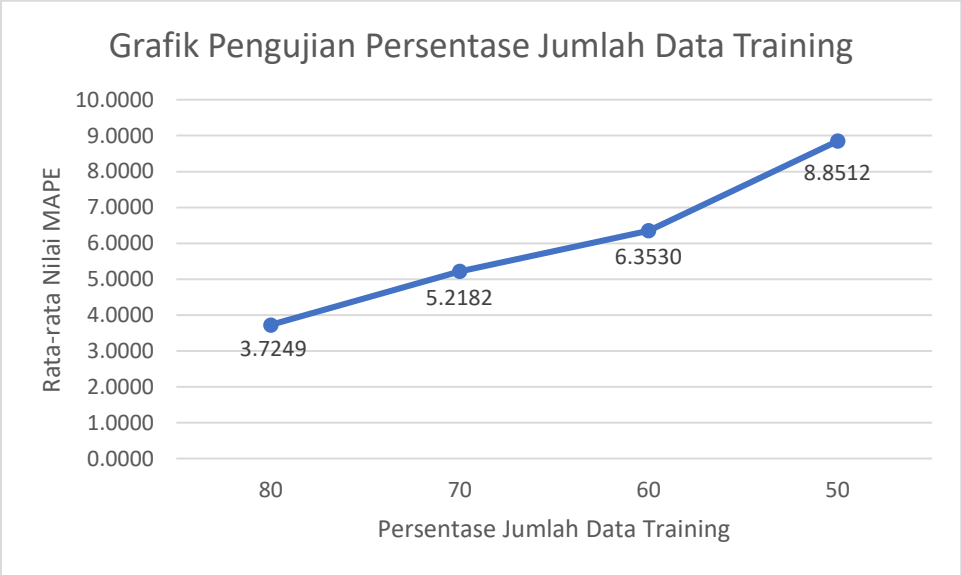
Persentase jumlah data *training* menentukan jumlah data yang masuk pada proses pembelajaran sistem. Semakin banyak jumlah data yang digunakan maka semakin banyak pola data yang harus dipelajari pada proses pembelajaran sistem untuk menghasilkan prediksi berdasarkan pola pergerakan data sebelumnya. Pengujian persentase jumlah data *training* dilakukan untuk menguji kelebihan algoritme ELM pada proses pembelajaran dalam mengatasi permasalahan *overfitting* dan *underfitting* yang terjadi pada algoritme Jaringan Saraf Tiruan (JST) lainnya. Algoritme ELM akan diuji pada jumlah data yang banyak untuk memicu terjadinya *overfitting* pada sistem, yaitu perlakuan sistem yang diakibatkan karena banyaknya jumlah data yang masuk pada proses pembelajaran sehingga sistem tidak dapat mengenali bentuk data lain selain data yang dipelajarinya dan menghasilkan prediksi dengan tingkat kesalahan yang tinggi. Algoritme ELM juga akan diuji pada jumlah data yang sedikit untuk memicu terjadinya *underfitting* pada sistem, yaitu perlakuan sistem yang menghasilkan prediksi data acak karena sistem tidak dapat mempelajari bentuk pola data dengan baik. Pada pengujian persentase jumlah data *training*, jumlah fitur yang digunakan adalah 2 fitur dan jumlah *neuron* yang digunakan adalah 6 *neuron*. Berikut adalah table hasil pengujian persentase jumlah data *training* ditunjukan pada Table 4.3.

Tabel 9.3 Hasil Pengujian Persentase Jumlah Data Training

Persentase Jumlah Data *Training*	Percobaan Ke-					Rata-rata MAPE
	1	2	3	4	5	
80%:20%	**3,6276**	**3,7747**	**3,7307**	**3,8201**	**3,6713**	**3,7249**

70%:20%	4,9665	5,2210	5,1300	5,8716	4,9020	5,2182
60%:20%	6,7699	6,3187	7,2773	6,7990	4,6003	6,3530
50%:20%	8,4118	9,1312	7,1723	5,9228	13,6177	8,8512

Hasil perhitungan tabel perlu diubah dalam bentuk diagram garis untuk mendapatkan analisa dari hasil pengujian. Berikut adalah diagram garis dari hasil pengujian persentase jumlah data *training* ditunjukan pada Gambar 4.13.



Gambar 9.13 Grafik Hasil Pengujian Persentase Jumlah Data Training

Berdasarkan hasil pengujian yang telah dilakukan pada Tabel 4.3, perbandingan persentase jumlah data *training* dan data *testing* yang menghasilkan rata-rata tingkat kesalahan terendah pada 5 kali percobaan adalah 80%:20% dengan nilai *error* sebesar 3,7249%. Berdasarkan grafik pengujian persentase jumlah data *training* pada Gambar 4.12, persentase jumlah data *training* sebanyak 80% dan 50% mampu menghasilkan tingkat kesalahan yang kurang dari 10%, hal tersebut menunjukan bahwa algoritme ELM mampu mengatasi permalasahan *overfitting* dan *underfitting* melalui proses pembelajaran sistem. Persentase jumlah data

training sangat mempengarui proses pembelajaran sistem sehingga untuk menghasilkan prediksi dengan tingkat kesalahan terendah dibutuhkan jumlah data yang banyak pada proses pembelajaran sistem.

9.2.2 Pengujian Algoritme *Support Vector Regression*

Pengujian parameter pada algoritme SVR ditunjukan untuk mengetahui parameter terbaik yang menghasilkan tingkat kesalahan terendah dengan menggunakan satuan nilai MAPE. Parameter yang diuji pada algoritme SVR adalah nilai *lambda, sigma, coefisien Learning Rate* (cLR), kompleksitas, *epsilon*, dan jumlah iterasi.

9.2.2.1 Pengujian Nilai *Lambda*

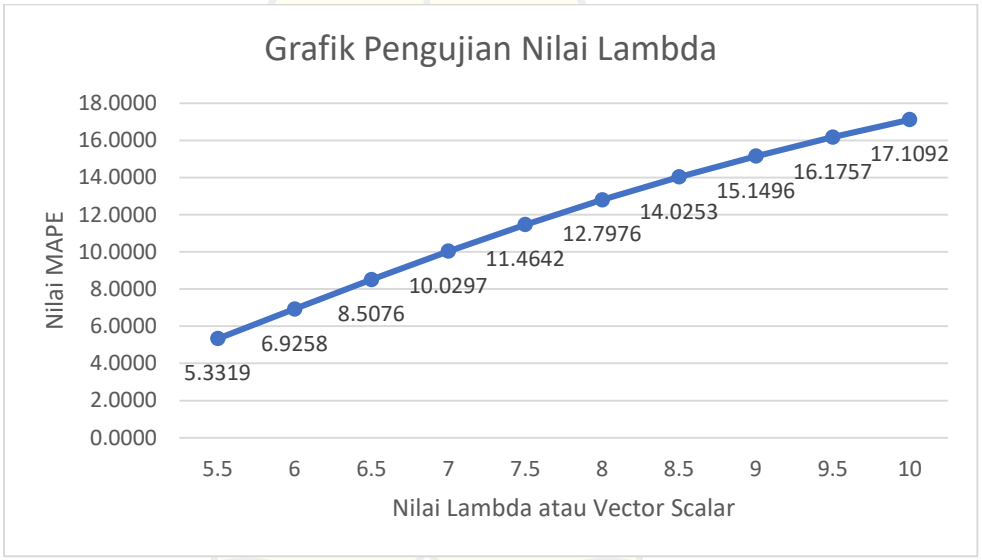
Parameter nilai *lambda* merupakan vector skalar yang mempengaruhi dimensi ruang pemetaan pada fungsi kernel. Pengujian ini dilakukan untuk mengetahui pengaruh perubahan nilai *lambda* pada fungsi kernel. Pada pengujian nila *lambda*, nilai *sigma* yang digunakan adalah 0,5, kemudian nilai *cLR* sebesar 0,01, nilai kompleksitas sebesar 100, nilai *epsilon* sebesar 0,000005, dan jumlah iterasi sebanyak 5000 iterasi. Tabel hasil pengujian nilai *lambda* ditunjukan pada Tabel 4.4.

Tabel 9.4 Hasil Pengujian Nilai *Lambda*

Nilai Lambda	Nilai MAPE
5,5	**5,3319093**
6	6,9258201
6,5	8,5075752
7	10,0296640

7,5	11,4642413
8	12,7975596
8,5	14,0252574
9	15,1496149
9,5	16,1757147
10	17,1091779

Hasil perhitungan tabel perlu diubah dalam bentuk diagram garis untuk mendapatkan analisa dari hasil pengujian. Berikut adalah diagram garis dari hasil pengujian nilai *lambda* ditunjukan pada Gambar 4.14.



Gambar 9.14 Grafik Hasil Pengujian Nilai *Lambda*

Berdasarkan hasil pengujian yang telah dilakukan pada Tabel 4.4, nilai *lambda* yang menghasilkan tingkat kesalahan terendah adalah 5,5 dengan nilai *error* sebesar 5,3319%. Berdasarkan grafik pengujian nilai *lambda* pada Gambar 4.14, dapat disimpulkan bahwa semakin bertambahnya nilai *lambda* yang digunakan maka nilai *error* yang dihasilkan juga semakin tinggi. Hal tersebut terjadi karena semakin tinggi nilai *lambda* yang digunakan maka dimensi yang terbentuk pada

ruang pemetaan fungsi kernel juga semakin tinggi. Jika ruang pemetaan pada fungsi kernel semakin tinggi maka jarak yang mengubungkan garis pemisah atau *hyplerlane* dengan data terdekat akan semakin jauh sehingga garis *hyperlane* yang terbentuk tidak dapat menyesuaikan bentuk pola data pada data target dan menghasilkan prediksi dengan tingkat kesalahan yang tinggi.

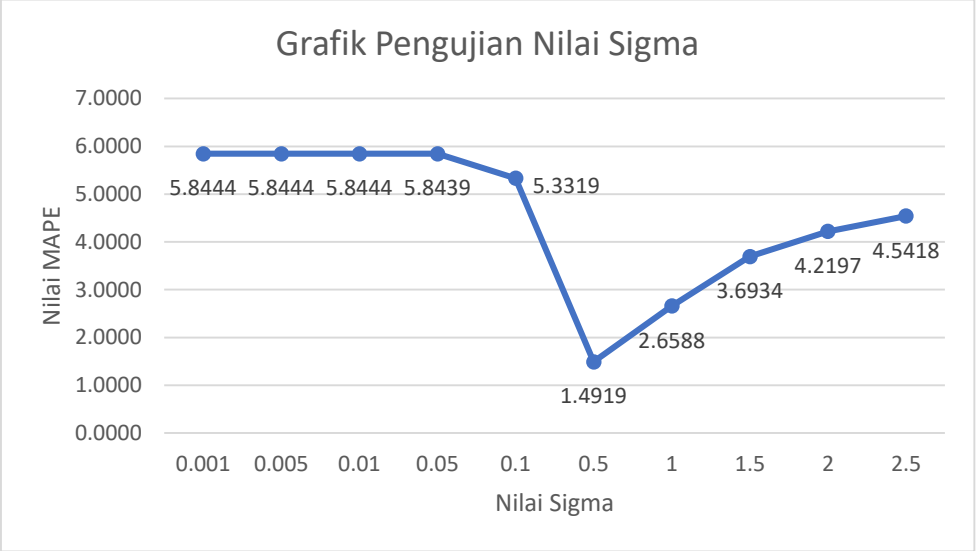
9.2.2.2 Pengujian Nilai *Sigma*

Parameter *sigma* digunakan untuk mengatur penyebaran data kedalam dimensi ruang pemetaan pada fungsi kernel. Pengujian ini dilakukan untuk mengetahui nilai konstanta yang optimal dalam mengatur penyebaran data pada fungsi kernel. Pada pengujian nilai *sigma*, nilai *lambda* yang digunakan adalah 5,5, kemudian nilai cLR sebesar 0,01, nilai kompleksitas sebesar 100, nilai *epsilon* sebesar 0,000005, dan jumlah iterasi sebanyak 5000 iterasi. Tabel hasil pengujian nilai *sigma* ditunjukan pada Tabel 4.5.

Tabel 9.5 Hasil Pengujian Nilai *Sigma*

Nilai Sigma	Nilai MAPE
0,001	5,8443904
0,005	5,8443904
0,01	5,8443904
0,05	5,8438641
0,1	5,3319093
0,5	**1,4918801**
1	2,6587975
1,5	3,6934217
2	4,2196620
2,5	4,5418307

Hasil perhitungan tabel perlu diubah dalam bentuk diagram garis untuk mendapatkan analisa dari hasil pengujian. Berikut adalah diagram garis dari hasil pengujian nilai *sigma* ditunjukan pada Gambar 4.15.



Gambar 9.15 Grafik Hasil Pengujian Nilai *Sigma*

Berdasarkan hasil pengujian yang telah dilakukan pada Tabel 4.5, nilai *sigma* yang menghasilkan tingkat kesalahan terendah adalah 0,5 dengan nilai *error* sebesar 1,4919%. Berdasarkan grafik pengujian nilai *sigma* pada Gambar 4.15, dapat disimpulkan bahwa nilai *sigma* yang semakin kecil menyebabkan persebaran data yang tidak sesuai sehingga garis *hyperlane* dihasilkan jauh dari data aktual.

9.2.2.3 Pengujian Nilai *coefisien Learning Rate* (cLR)

Parameter cLR digunakan mengatur laju proses pembelajaran pada algoritme SVR. Semakin tinggi nilai cLR yang digunakan maka semakin cepat laju proses pembelajaran pada algoritme SVR. Namun, hal tersebut dapat beresiko untuk menghasilkan nilai prediksi yang buruk. Untuk itu perlu dilakukan pengujian untuk mengetahui laju proses pembelajaran yang tepat yang mampu menghasilkan tingkat

kesalahan terendah. Pada pengujian nilai cLR, nilai *lambda* yang digunakan adalah 5,5, kemudian nilai *sigma* sebesar 0,5, nilai kompleksitas sebesar 100, nilai *epsilon* sebesar 0,000005 dan jumlah iterasi sebanyak 5000 iterasi. Tabel hasil pengujian nilai cLR ditunjukan pada Tabel 4.6.

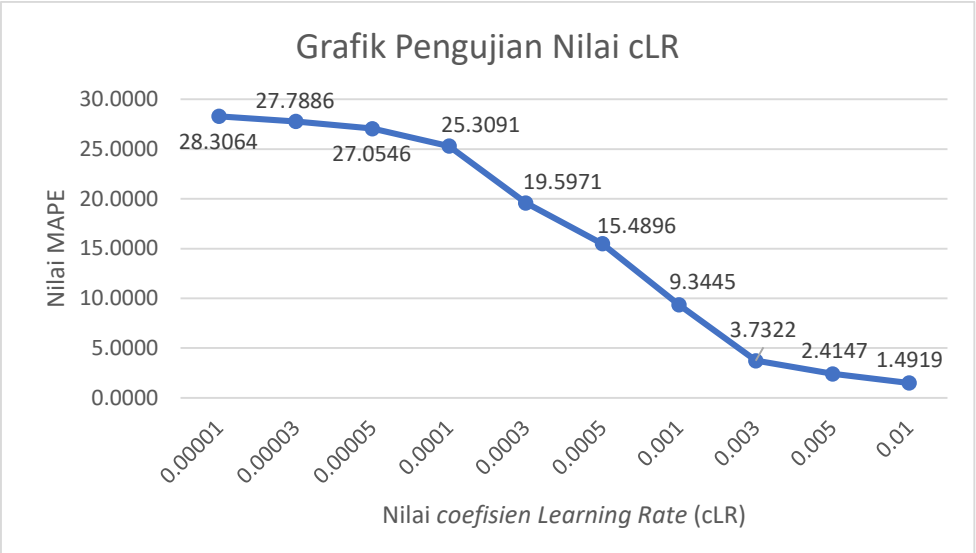
Tabel 9.6 Hasil Pengujian Nilai cLR

Nilai cLR	Nilai MAPE
0,00001	28,3064
0,00003	27,7886
0,00005	27,0546
0,0001	25,3091
0,0003	19,5971
0,0005	15,4896
0,001	9,3445
0,003	3,7322
0,005	2,4147
0,01	**1,4919**

Hasil perhitungan tabel perlu diubah dalam bentuk diagram garis untuk mendapatkan analisa dari hasil pengujian. Berikut adalah diagram garis dari hasil pengujian nilai cLR ditunjukan pada Gambar 4.16.

Berdasarkan hasil pengujian yang telah dilakukan pada Tabel 4.6, nilai cLR yang menghasilkan tingkat kesalahan terendah adalah 0,01 dengan nilai *error* sebesar 1,4919%. Berdasarkan grafik pengujian nilai cLR pada Gambar 4.16, dapat disimpulkan bahwa semakin lambat laju pembelajaran yang diberikan pada algoritme SVR maka nilai prediksi yang dihasilkan akan semakin optimal. Namun sebaliknya, semakin cepat laju pembelajaran yang diberikan pada algoritme SVR

maka nilai prediksi yang dihasilkan akan semakin buruk. Hal tersebut menandakan bahwa algoritme SVR memiliki kekurangan pada proses pembelajarannya.



Gambar 9.16 Grafik Hasil Pengujian Nilai cLR

9.2.2.4 Pengujian Nilai Kompleksitas (C)

Parameter kompleksitas (C) merupakan nilai batas toleransi digunakan algoritme SVR untuk mengatur kesalahan yang dihasilkan pada nilai alfa. Semakin besar nilai C yang digunakan maka besar toleransi tingkat kesalahan yang diberikan pada nilai alfa. Semakin kecil nilai C yang digunakan maka toleransi yang diberikan pada nilai alfa akan semakin ketat sehingga nilai alfa berpotensi untuk mendapatkan pinalti jika nilai alfa melewati batas nilai *error* yang telah ditentukan. Pinalti tersebut akan menggantikan nilai alfa dengan nilai kompleksitas. Jika hal ini terjadi maka nilai prediksi yang dihasilkan cenderung stabil dan tidak menyesuaikan pergerakan pola data pada data target karena nilai alfa akan selalu digantikan dengan nilai kompleksitas. Untuk itu, perlu dilakukan pengujian untuk mengetahui nilai toleransi yang tepat yang mampu menghasilkan tingkat kesalahan terendah.

Pada pengujian nilai kompleksitas, nilai *lambda* yang digunakan adalah 5,5, kemudian nilai *sigma* sebesar 0,5, nilai cLR sebesar 0,01, nilai *epsilon* sebesar 0,000005, dan jumlah iterasi sebanyak 5000 iterasi. Tabel hasil pengujian nilai kompleksitas ditunjukan pada Tabel 4.7.

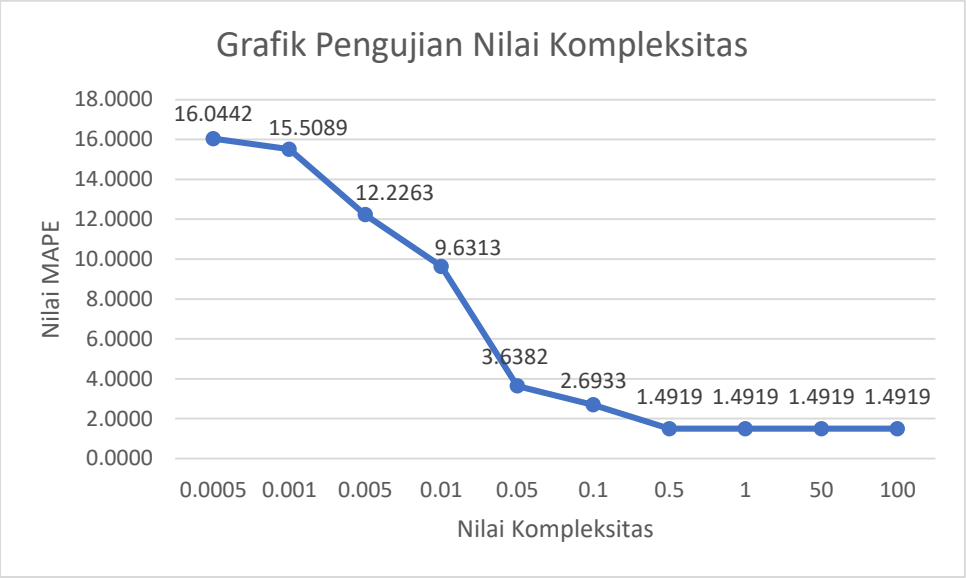
Tabel 9.7 Hasil Pengujian Nilai Kompleksitas

Nilai Kompleksitas	Nilai MAPE
0,0005	16,0442
0,001	15,5089
0,005	12,2263
0,01	9,6313
0,05	3,6382
0,1	2,6933
0,5	1,4919
1	1,4919
50	1,4919
100	**1,4919**

Hasil perhitungan tabel perlu diubah dalam bentuk diagram garis untuk mendapatkan analisa dari hasil pengujian. Berikut adalah diagram garis dari hasil pengujian nilai kompleksitas ditunjukan pada Gambar 4.17.

Berdasarkan hasil pengujian yang telah dilakukan pada Tabel 4.7, nilai kompleksitas yang menghasilkan tingkat kesalahan terendah adalah 100 dengan nilai *error* sebesar 1,4919%. Berdasarkan grafik pengujian nilai kompleksitas pada Gambar 4.17, dapat disimpulkan bahwa semakin besar nilai C yang digunakan menjadikan algoritme SVR tidak mentoleransi kesalahan yang dihasilkan nilai alfa sehingga menghasilkan nilai prediksi yang cenderung mengikuti data target dengan

tingkat kesalahan yang lebih rendah. Namun sebaliknya, semakin kecil nilai C yang digunakan maka akan menjadikan algoritme SVR membuat toleransi yang ketat pada nilai alfa sehingga nilai alfa akan selalu tergantikan dengan nilai kompleksitas dan menghasilkan nilai prediksi yang cenderung stabil dengan tingkat kesalahan yang tinggi.



Gambar 9.17 Grafik Hasil Pengujian Nilai Kompleksitas

9.2.2.5 Pengujian Nilai *Epsilon*

Parameter epsilon merupakan nilai yang mentukan batas *error* pada nilai alfa. Semakin kecil nilai *epsilon* yang digunakan maka semakin besar toleransi yang diberikan pada nilai alfa agar tidak terkena pinalti sehingga nilai alfa tidak digantikan dengan nilai kompleksitas. Nilai *epsilon* yang semakin kecil akan mempengaruhi laju proses pembelajaran pada algoritme SVR sehingga membuat laju proses pembelajaran akan semakin lambat. Namun, jika nilai *epsilon* yang digunakan terlalu besar dapat meningkatkan resiko tergantinya nilai alfa dengan nilai kompleksitas walaupun laju proses pembelajarannya akan menjadi semakin

cepat. Untuk itu perlu dilakukan pengujian untuk mengetahui batas nilai toleransi yang tepat yang mampu menghasilkan tingkat kesalahan terendah. Pada pengujian nilai *epsilon*, nilai *lambda* yang digunakan adalah 5,5, kemudian nilai *sigma* sebesar 0,5, nilai cLR sebesar 0,01, nilai kompleksitas sebesar 100, dan jumlah iterasi sebanyak 5000 iterasi. Tabel hasil pengujian nilai *epsilon* ditunjukan pada Tabel 4.7.

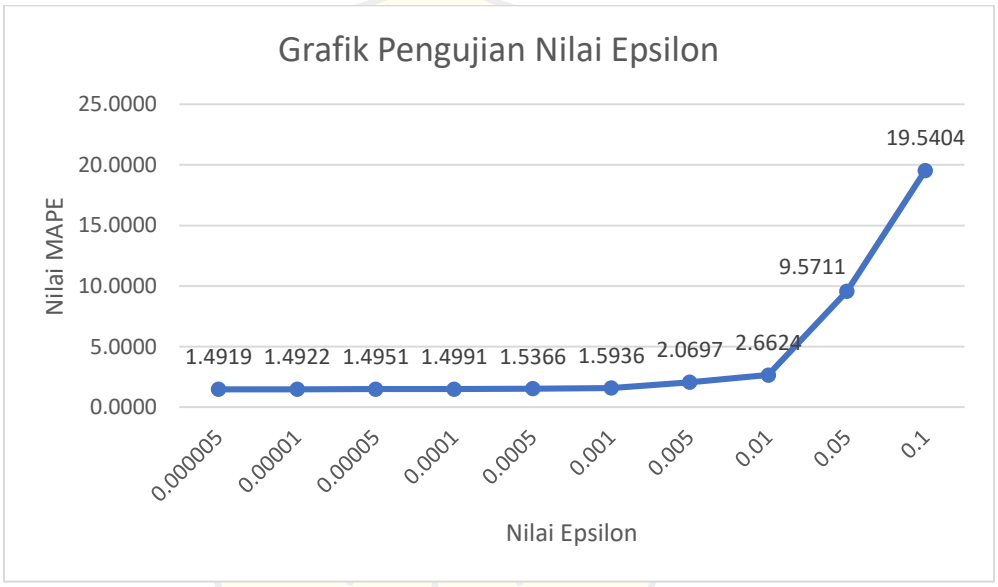
Tabel 9.8 Hasil Pengujian Nilai *Epsilon*

Nilai Epsilon	Nilai MAPE
0,000005	**1,4919**
0,00001	1,4922
0,00005	1,4951
0,0001	1,4991
0,0005	1,5366
0,001	1,5936
0,005	2,0697
0,01	2,6624
0,05	9,5711
0,1	19,5404

Hasil perhitungan tabel perlu diubah dalam bentuk diagram garis untuk mendapatkan analisa dari hasil pengujian. Berikut adalah diagram garis dari hasil pengujian nilai *epsilon* ditunjukan pada Gambar 4.18.

Berdasarkan hasil pengujian yang telah dilakukan pada Tabel 4.8, nilai *epsilon* yang menghasilkan tingkat kesalahan terendah adalah 0,000005 dengan nilai *error* sebesar 1,4919%. Berdasarkan grafik pengujian nilai *epsilon* pada Gambar 4.18, dapat disimpulkan bahwa semakin kecil nilai *epsilon* yang digunakan menjadikan

nilai alfa yang dihasilkan bebas dari toleransi nilai *error* sehingga nilai alfa tidak berpotensi untuk digantikan dengan nilai kompleksitas. Hal tersebut membuat nilai prediksi yang dihasilkan cenderung mengikuti data target dengan tingkat kesalahan yang lebih rendah. Namun sebaliknya, semakin besar *epsilon* yang digunakan menjadikan toleransi yang diberikan pada nilai alfa sangat ketat dan berpotensi untuk digantikan dengan nilai kompleksitas. Hal tersebut memaksa nilai alfa untuk digantikan dengan nilai kompleksitas sehingga nilai prediksi yang dihasilkan akan cenderung stabil dengan tingkat kesalahan yang tinggi.



Gambar 9.18 Grafik Hasil Pengujian Nilai *Epsilon*

9.2.2.6 Pengujian Jumlah Iterasi

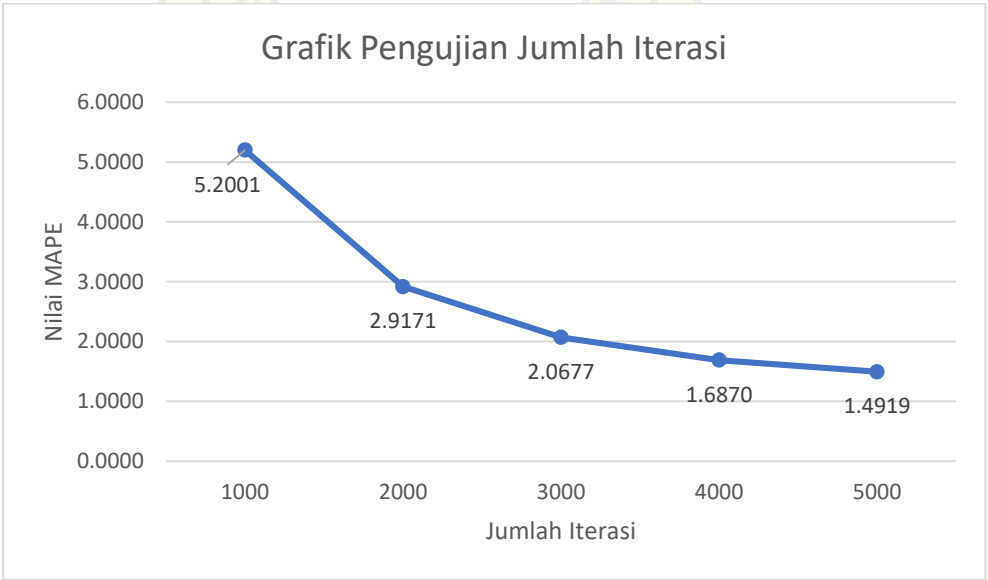
Pengujian jumlah iterasi dilakukan untuk menentukan jumlah iterasi terbaik yang menghasilkan nilai prediksi dengan tingkat kesalahan terendah. Jika dalam iterasi tertentu nilai prediksi telah mencapai konvergensi maka jumlah iterasi pertama sebelum terjadi konvergensi merupakan jumlah iterasi yang optimal pada algoritme SVR. Pada pengujian jumlah iterasi, nilai *lambda* yang digunakan adalah

5,5, kemudian nilai *sigma* sebesar 0,5, nilai cLR sebesar 0,01, nilai kompleksitas sebesar 100, dan nilai epsilon sebesar 0,000005. Tabel hasil pengujian jumlah iterasi ditunjukan pada Tabel 4.9.

Tabel 9.9 Hasil Pengujian Jumlah Iterasi

Jumlah Iterasi	Nilai MAPE
1000	5,2001
2000	2,9171
3000	2,0677
4000	1,6870
5000	**1,4919**

Hasil perhitungan tabel perlu diubah dalam bentuk diagram garis untuk mendapatkan analisa dari hasil pengujian. Berikut adalah diagram garis dari hasil pengujian jumlah iterasi ditunjukan pada Gambar 4.19.



Gambar 9.19 Grafik Hasil Pengujian Jumlah Iterasi

Berdasarkan hasil pengujian yang telah dilakukan pada Tabel 4.9, jumlah iterasi yang menghasilkan tingkat kesalahan terendah adalah 5000 iterasi dengan nilai *error* sebesar 1,4919%. Berdasarkan grafik pengujian jumlah iterasi pada Gambar 4.19, dapat disimpulkan bahwa semakin besar jumlah iterasi yang digunakan maka semakin besar kemampuan algoritme SVR dalam melakukan observasi terhadap pergerakan pola data sehingga algoritme SVR mampu menghasilkan nilai prediksi dengan tingkat kesalahan yang lebih rendah. Namun sebaliknya, semakin kecil jumlah iterasi yang digunakan maka semakin sedikit observasi yang dilakukan algoritme SVR terhadap pergerakan pola data sehingga membuat nilai prediksi yang dihasilkan memiliki tingkat kesalahan yang tidak lebih baik dibandingkan tingkat kasalahan dengan menggunakan jumlah iterasi yang lebih banyak. Akan tetapi, semakin banyak jumlah iterasi yang digunakan maka semakin banyak waktu komputasi yang dibutuhkan pada proses pembelajaran sampai memenuhi kondisi konvergensi atau iterasi maksimum. Hal tersebut menjadi kelemahan algoritme SVR pada proses pembelajaran untuk menghasilkan nilai prediksi yang optimal.

TEKNOLOGI INFORMASI

UNIVERSITAS DARMA PERSADA

BAB 10

KESIMPULAN DAN SARAN

10.1 Kesimpulan

Berdasarkan pembahasan dari hasil pengujian yang telah dilakukan, terdapat beberapa kesimpulan sebagai berikut.

1. Pada prediksi harga cabai rawit di pasar jatinegara, Jakarta Timur, algoritme ELM mampu menghasilkan rata-rata nilai MAPE terendah sebesar 3,6746% dengan waktu komputasi yang dibutuhkan pada proses pembelajaran sebesar 0,0110832 *seconds*. Hasil tersebut didapat dengan menggunakan parameter optimal dari hasil pengujian, yaitu 2 fitur, 6 *neuron* dan perbandingan persentase jumlah data training dan data testing sebesar 80%:20%. Kemudian algoritme SVR mampu menghasilkan nilai MAPE terendah sebesar 1,4919% dengan waktu komputasi yang dibutuhkan pada proses pembelajaran sebesar 3,0919356 *seconds*. Hasil tersebut didapat dengan menggunakan parameter optimal dari hasil pengujian. Parameter tersebut adalah nilai *lambda* sebesar 5,5, kemudian nilai *sigma* sebesar 0,5, nilai cLR sebesar 0,01, nilai kompleksitas sebesar 100, nilai *epsilon* sebesar 0,000005, dan jumlah iterasi sebanyak 5000 iterasi.
2. Dari hasil pengujian yang telah dilakukan dapat disimpulkan bahwa algoritme ELM unggul dalam proses pembelajaran sedangkan algoritme SVR unggul dalam akurasi tingkat kesalahan. Namun, kedua algoritme tersebut mampu menghasilkan nilai prediksi dengan tingkat kesalahan yang kurang dari 10%.

Hal tersebut menunjukan bahwa nilai prediksi yang dihasilkan termasuk dalam kategori yang sangat baik.

10.2 Saran

Berdasarkan penelitian yang dilakukan, terdapat saran sebagai tindakan lanjut untuk mengembangkan penelitian antara lain.

1. Jumlah data yang digunakan pada penelitian untuk prediksi harga cabai rawit di pasar jatinegara sangat terbatas. Untuk itu diperlukan peningkatkan pada jumlah data karena sangat berpengaruh terhadap proses pembelajaran sistem. Semakin banyak jumlah data yang digunakan maka nilai prediksi yang dihasilkan memiliki tingkat kesalahan yang lebih rendah sehingga hasil tersebut dapat dijadikan sebagai acuan bagi banyak masyarakat luas untuk membantu mengatasi pemasalahan harga cabai rawit yang fluktuatif.

DAFTAR PUSTAKA

Abadi, I. and Soeprijanto, A., 2014, November. Extreme learning machine approach to estimate hourly solar radiation on horizontal surface (PV) in Surabaya-East java. In Information Technology, Computer and Electrical Engineering (ICITACEE), 2014 1st International Conference on (pp. 372-376). IEEE.

Adiatmaja, P. B., Setiawan, B. D., & Wihandika, R. C. (2019). Peramalan Harga Cabai Merah Besar Wilayah Jawa Timur Menggunakan Metode Extreme Learning Machine. *Jurnal Pengembangan Teknologi Informasi Dan Ilmu Komputer*, *3*(6), 5444–5449.

Andini, T. D. & Auristandi, P., 2016. Peramalan Jumlah Stok Alat Tulis Kantor Di UD Achmad Jaya Menggunakan Metode Double Exponential Smooting. Jurnal Ilmiah Teknologi dan Informasi ASIA (JITIKA).

Ariwanda, G., Cholissodin, I., & Tibyani. (2019). Prediksi Harga Cabai Rawit di Kota Malang Menggunakan Algoritme Extreme Learning Machine (ELM). *Jurnal Pengembangan Teknologi Informasi Dan Ilmu Komputer*, *3*(6), 5291–5298.

Cheng, G. J., Cai, L., & Pan, H. X., 2009. Comparison of extreme learning machine with support vector regression for reservoir permeability prediction. *CIS 2009 - 2009 International Conference on Computational Intelligence and Security*, *2*, 173–176. https://doi.org/10.1109/CIS.2009.124.

Fachrony, A., Cholissodin, I. & Santoso, E., 2018. Implementasi Algoritme

Extreme Learning Machine (ELM) untuk Prediksi Beban Pemanasan dan Pendinginan Bangunan. Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer, 2(9), pp.3043–3049.

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

Furi, R. P., Jondri & Saepudin, D., 2015. Peramalan Financial Time Series Menggunakan Independent Component Analysis dan Support Vector Regression (Studi Kasus: IHSG dan Jll). S1. Telkom University.

Herdianto, 2013, Prediksi Kerusakan Motor Induksi Menggunakan Metode Jaringan Saraf Tiruan Backpropagation, Tesis, Universitas Sumatera Utara, Medan.

Huang, G. Bin, Zhu, Q. Y., & Siew, C. K., 2004. Extreme learning machine: A new learning scheme of feedforward neural networks. *IEEE International Conference on Neural Networks - Conference Proceedings*, *2*, 985–990. https://doi.org/10.1109/IJCNN.2004.1380068.

Ikhsan, F., & Setiawan, B. D. (2019). Prediksi Penjualan Seblak menggunakan Algoritme Extreme Learning Machine di Seblak Malabar, *3*(11), 10630–10637.

Maharesi, R., 2013. Penggunaan Support Vector Regression (SVR) Pada Prediksi Return Saham Syariah BEI. Proceeding PESAT, Volume 5.

Mustaffa, Z. & Yusof, Y., 2011. A comparison of normalization techniques in predicting dengue outbreaik. *International Conference on Business and Economics Research*, 1, pp.345–349.

Puspaningrum, M. D., Santoso, E., & Yudistira, N. (2020). Prediksi Persentase Penyelesaian Permohonan Hak Milik menggunakan Metode Extreme Learning Machine (ELM) (Studi Kasus : Badan Pertanahan Nasional Kabupaten Malang), *4*(7), 2236–2242.

Rahmadiani, A. & Anggraeni, W., 2012. Implementasi Fuzzy Neural Network untuk Memperkirakan Jumlah Kunjungan Pasien Poli Bedah di Rumah Sakit Onkologi Surabaya. Jurnal Tekni POMITS, I(1), pp. 1- 5.

Rifqi, M. R., Setiawan, B. D., & Bacthiar, F. A. (2018). Support Vector Regression untuk Peramalan Permintaan Darah : Studi Kasus Unit Transfusi Darah Cabang – PMI Kota Malang. *Jurnal Pengembangan Teknologi Informasi Dan Ilmu Komputer (J-PTIIK) Universitas Brawijaya*, *2*(10), 3332–3342.

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

Sepri, D., Fauzi, A., Wandira, R., Riza, O. S., & Fitri, Y. (2020). Prediksi Harga Cabai Merah Menggunakan Support Vector Regression. *Ejournal.Upbatam.Ac.Id*, (October), 0–5. Retrieved from http://ejournal.upbatam.ac.id/index.php/cbis/article/view/1921.

Singh, R. and Balasundaram, S., 2007. Application of extreme learning machine method for time series analysis. International Journal of Intelligent Technology, 2(4), pp.256-262.

Srimuang, W., & Intarasothonchun, S., 2015. Classification Model of Network Intrusion using Weighted Extreme Learning Machine. 12th International

Joint Conference on Computer Science and Software Engineering (JCSSE), 190-194.

Sujitno, E dan M. Dianawati. 2015. Produksi Panen berbagai Varietas Unggul Baru Cabai Rawit (*Capsicum frutescens*) di Lahan Kering Kabupaten Garut, Jawa Barat. *Pros Seminar Nasional Masyarakat Biodiversitas Indonesial* 1(4): 874-877.

Tjandra, E., 2011, Panen Cabai Rawit Di Polybag, Cahaya Atma Pustaka, Yogyakarta

Vijayakumar, S. & Wu, S., 1999. Sequential Support Vector Classifiers and Regression. Genoa, Italy, Saitama: RIKEN Brain Science Institute, The Institute for Physical and Chemical Research.

Wang, G., Zhao, Yi., & Wang, Di., 2008. Aprotein Secondary Structure Prediction Framework Based On The Extreme Learning Machine. Journal Neurocomputing, 262-268.

LAMPIRAN A

SOURCE CODE ALGORITME EXTREME

LEARNING MACHINE

Fungsi Main ELM

```
1   private void button4_Click(object sender, EventArgs e)
        {
2           timerr.Start();
3           long minMax = 2500;
4           //PengambilanParameter
5           int persentaseDataTraining =
    Convert.ToInt32(comboBox2.Text);
6           int persentaseDataTesting = 20;
7           int fitur = Convert.ToInt32(comboBox3.Text);
8           int neuron = Convert.ToInt32(comboBox4.Text);
9           Random ran = new Random();
10          //Preprocessing
11          long[,] dataFitur = membuatFitur(dataset,
    fitur);
12          long findMin = searchMin(dataFitur, minMax);
13          long findMax = searchMax(dataFitur, minMax);
14          double[,] normalisasi =
    normalisasiData(dataFitur, findMin, findMax);
15          double[,] bobot = generateBobot(fitur, neuron,
    ran);
16          double[,] bias = generateBias(neuron, ran);
17          //Proses Training
18          double[,] dataTrain = dataTraining(normalisasi,
    persentaseDataTraining, persentaseDataTesting);
19          double[,] aktualDataTrain =
    aktualDataTraining(normalisasi, persentaseDataTraining,
    persentaseDataTesting);
20          double[,] hinit = getHinit(dataTrain, bobot);
21          double[,] h = fungsiAktivasiHiddenLayer(hinit,
    bias);
22          double[,] hTranspose = transposeMatriks(h);
23          double[,] beforeInverse =
    perkalianMatriks(hTranspose, h);
24          double[,] inverseMatriks =
    inverseOBE(beforeInverse);
25          double[,] hDagger =
    perkalianMatriks(inverseMatriks, hTranspose);
26          double[,] beta = outputWeight(hDagger,
    aktualDataTrain);
27          //Proses Testing
28          double[,] dataTest = dataTesting(normalisasi,
    persentaseDataTesting);
29          long[,] aktualDataTest =
    aktualDataTesting(dataFitur, persentaseDataTesting);
30          double[,] hinitTest = getHinit(dataTest,
    bobot);
```

```
31              double[,] hTest =
    fungsiAktivasiHiddenLayer(hinitTest, bias);
32              double[,] yTopi = outputLayer(hTest, beta);
33              //Denormalisasi & Evaluasi MAPE
34              double[,] denormalisasi =
    denormalisasiData(yTopi, findMin, findMax);
35              double mape = evaluasiMAPE(denormalisasi,
    aktualDataTest);
36              //PrediksiBaru
37              long[,] dataPred = dataPrediksi(dataFitur);
38              double[,] normalisasiDataPred =
    normalisasiData(dataPred, findMin, findMax);
39              double[,] hinitPred =
    getHinit(normalisasiDataPred, bobot);
40              double[,] hPred =
    fungsiAktivasiHiddenLayer(hinitPred, bias);
41              double[,] yTopiPred = outputLayer(hPred, beta);
42              //Denormalisasi Hasil Prediksi
43              double[,] denormalisasiPred =
    denormalisasiData(yTopiPred, findMin, findMax);
44              long ambilPrediksi =
    Convert.ToInt64(denormalisasiPred[0, 0]);
45              timerr.Stop();
46  }
```

LAMPIRAN B

SOURCE CODE ALGORITME SUPPORT VECTOR REGRESSION

Fungsi Main SVR

```
1   private void button4_Click(object sender, EventArgs e)
2           {
3               timerr.Start();
4               long minMax = 2500;
5               //PengambilanParameter
6               double lamda =
    Convert.ToDouble(comboBox2.Text);
7               double sigma =
    Convert.ToDouble(comboBox3.Text);
8               double cLR = Convert.ToDouble(comboBox4.Text);
9               double complexity =
    Convert.ToDouble(comboBox5.Text);
10              double epsilon =
    Convert.ToDouble(comboBox6.Text);
11              long iterasi = Convert.ToInt64(comboBox7.Text);
12              int fitur = (dataset.GetLength(0) - 1) / 2;
13              //PreProcessing
14              double gamma = findGamma(cLR, lamda);
15              long[,] dataFitur = membuatFitur(dataset,
    fitur);
16              long findMin = searchMin(dataFitur, minMax);
17              long findMax = searchMax(dataFitur, minMax);
18              double[,] normalisasi =
    normalisasiData(dataFitur, findMin, findMax);
19              //ProsesTraining
20              double[,] jarak = hitungJarak(normalisasi);
21              double[,] rbf = gaussianRBF(jarak, sigma,
    lamda);
22              double[,] hasilAlfa = findAlfa(normalisasi,
    rbf, gamma, epsilon, complexity, iterasi);
23              double[,] alfa1 = ambilAlfa1(hasilAlfa);
24              double[,] alfa2 = ambilAlfa2(hasilAlfa);
25              //ProsesTesting & Evaluasi MAPE
26              long[,] dataAktual =
    ambilDataAktual(dataFitur);
27              double[,] hasilPrediksi = prediksi(rbf,
    hasilAlfa);
28              double[,] denormalisasi =
    denormalisasiData(hasilPrediksi, findMin, findMax);
29              double mape = evaluasiMAPE(denormalisasi,
    dataAktual);
30              //PrediksiBaru
31              long[,] dataBaru = dataPrediksi(dataFitur);
32              double[,] normalisasiDataBaru =
    normalisasiData(dataBaru, findMin, findMax);
33
```

```
            double[,] jarakBaru =
34  hitungJarakDataPrediksi(normalisasi, normalisasiDataBaru);
            double[,] rbfBaru = gaussianRBF(jarakBaru,
35  sigma, lamda);
            double[,] hasilPrediksiBaru = prediksi(rbfBaru,
36  hasilAlfa);
            double[,] denormalisasiBaru =
37  denormalisasiData(hasilPrediksiBaru, findMin, findMax);
            long ambilPrediksi =
38  Convert.ToInt64(denormalisasiBaru[0, 0]);
39          timerr.Stop();
40  }
```

LAMPIRAN C

DATA HARGA CABAI RAWIT

No	Tanggal	Cabai Rawit Merah
1	02/01/2020	50000
2	03/01/2020	55000
3	06/01/2020	75000
4	07/01/2020	75000
5	08/01/2020	75000
6	09/01/2020	75000
7	10/01/2020	75000
8	13/01/2020	75000
9	14/01/2020	75000
10	15/01/2020	90000
11	16/01/2020	90000
12	17/01/2020	90000
13	20/01/2020	85000
14	21/01/2020	90000
15	22/01/2020	95000
16	23/01/2020	100000
17	24/01/2020	100000
18	27/01/2020	100000
19	28/01/2020	100000
20	29/01/2020	100000
21	30/01/2020	100000
22	31/01/2020	100000
23	03/02/2020	100000
24	04/02/2020	100000
25	05/02/2020	100000
26	06/02/2020	100000

No	Tanggal	Cabai Rawit Merah
27	07/02/2020	100000
28	10/02/2020	80000
29	11/02/2020	80000
30	12/02/2020	80000
31	13/02/2020	75000
32	14/02/2020	70000
33	17/02/2020	70000
34	18/02/2020	70000
35	19/02/2020	70000
36	20/02/2020	60000
37	21/02/2020	50000
38	24/02/2020	55000
39	25/02/2020	55000
40	26/02/2020	55000
41	27/02/2020	55000
42	28/02/2020	50000
…	…	…
239	23/12/2020	60000
240	28/12/2020	70000
241	29/12/2020	70000
242	30/12/2020	80000
243	04/01/2021	90000
244	05/01/2021	100000
245	06/01/2021	100000
246	07/01/2021	100000
247	08/01/2021	95000

LAMPIRAN D

HASIL DATA PREDIKSI DAN DATA AKTUAL

MENGGUNAKAN ALGORITME EXTREME

LEARNING MACHINE

No	Data Prediksi	Data Akual
1	30019,78	30000
2	30019,78	30000
3	30019,78	30000
4	30019,78	30000
5	30019,78	32500
6	32866,17	35000
7	35392,06	35000
8	35040,36	35000
9	35040,36	35000
10	35040,36	37500
11	37915,73	37500
12	37548,89	37500
13	37548,89	37500
14	37548,89	37500
15	37548,89	37500
16	37548,89	37500
17	37548,89	37500
18	37548,89	37500
19	37548,89	35000
20	34684,82	40000
21	40815,16	40000
22	40055,3	40000
23	40055,3	40000

No	Data Prediksi	Data Aktual
24	40055,3	50000
25	51773,74	50000
26	50048,63	50000
27	50048,63	50000
28	50048,63	50000
29	50048,63	52500
30	52973,73	60000
31	61363,68	60000
32	59970,53	60000
33	59970,53	67500
34	68783,51	60000
35	58659,58	60000
36	59970,53	60000
37	59970,53	60000
38	59970,53	60000
39	59970,53	62500
…	…	…
44	69803,15	80000
45	81452,75	90000
46	90973,54	100000
47	100290,9	100000
48	98607,42	100000
49	98607,42	95000

LAMPIRAN E

HASIL DATA PREDIKSI DAN DATA AKTUAL MENGGUNAKAN ALGORITME SUPPORT VECTOR REGRESSION

No	Data Prediksi	Data Akual
1	30793,08	30000
2	30056,51	30000
3	30054,37	30000
4	30003,15	30000
5	30046,29	30000
6	30543,34	30000
7	32250,83	32500
8	32608,13	32500
9	32638,68	32500
10	32659,71	32500
11	31937,25	32500
12	28614,03	27500
13	27585,22	27500
14	27477,94	27500
15	27536,36	27500
16	28063,26	27500
17	29670,29	30000
18	30047,48	30000
19	30076,75	30000
20	30726,62	30000
21	32515,12	33500
22	31519,36	31000
23	31045,63	31000

No	Data Prediksi	Data Aktual
24	31019	31000
25	31055,94	31000
26	31075,9	31000
27	31086,42	31000
28	31019,72	31000
29	31061,1	31000
30	30495,14	31000
31	28384,87	27500
32	27554,85	27500
33	27452,1	27500
34	27555,2	27500
35	28157,73	27500
36	29624,23	30000
37	30112,1	30000
38	29989,32	30000
39	29996,52	30000
40	30006,89	30000
...	…	…
120	88919,61	90000
121	98162,47	100000
122	101480,9	100000
123	100067,1	100000
124	91608,07	95000

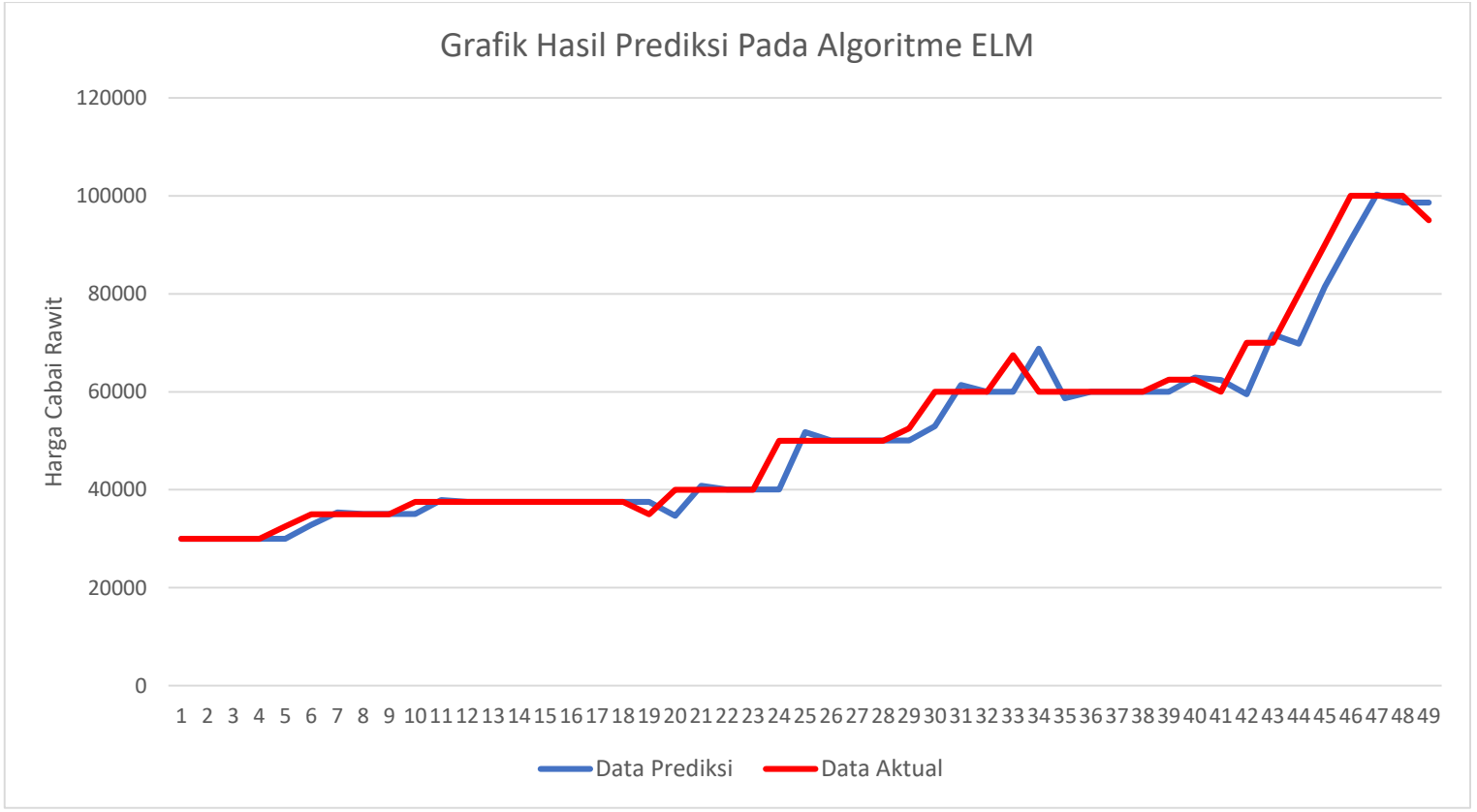
LAMPIRAN F

GRAFIK DATA PREDIKSI DAN DATA AKTUAL MENGGUNAKAN ALGORITME EXTREME LEARNING MACHINE

Grafik hasil prediksi algoritme ELM dihitung menggunakan parameter terbaik yang telah dilakukan uji coba. Berikut adalah grafik hasil prediksi pada algoritme ELM yang menghasilkan tingkat kesalahan terendah.

Jumlah Fitur	: 2
Jumlah *Hidden Neuron*	: 6
Fungsi Aktivasi	: *Sigmoid Biner*
Persentase Jumlah Data *Training*	: 80%
Persentase Jumlah Data *Testing*	: 20%
Rata-rata nilai MAPE	: 3,7249%

Waktu Komputasi : 0,0110832 *seconds*



LAMPIRAN G

GRAFIK DATA PREDIKSI DAN DATA AKTUAL MENGGUNAKAN ALGORITME SUPPORT VECTOR REGRESSION

Grafik hasil prediksi algoritme SVR dihitung menggunakan parameter terbaik yang telah dilakukan uji coba. Berikut adalah grafik hasil prediksi pada algoritme SVR yang menghasilkan tingkat kesalahan terendah.

Nilai *Lambda*	: 5,5
Nilai *Sigma*	: 0,5
Nilai *coefisien Learning Rate* (cLR)	: 0,01
Nilai Kompleksitas	: 100
Nilai *Epsilon*	: 0,000005
Jumlah iterasi	: 5000

Rata-rata nilai MAPE : 1,4919%

Waktu Komputasi : 3,0919356 *seconds*