

**IMPLEMENTASI METODE *SUPPORT VECTOR REGRESSION*
(SVR) DALAM PERAMALAN PENJUALAN ROTI
(STUDI KASUS: HARUM BAKERY)**

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:
Noval Dini Maulana
NIM: 165150209111002



PROGRAM STUDI TEKNIK INFORMATIKA
JURUSAN TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2018



PENGESAHAN

IMPLEMENTASI METODE *SUPPORT VECTOR REGRESSION (SVR)* DALAM PERAMALAN PENJUALAN ROTI (STUDI KASUS: HARUM BAKERY)

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan memperoleh gelar Sarjana Komputer

Disusun Oleh :
Noval Dini Maulana
NIM: 165150209111002

Skrripsi ini telah diuji dan dinyatakan lulus pada
28 Desember 2018

Telah diperiksa dan disetujui oleh:

Dosen Pembimbing 1

Budi Darma Setiawan, S.Kom, M.Cs
NIP: 198410152014041002

Dosen Pembimbing 2

Candra Dewi, S.Kom, M.Sc
NIP: 197711142003122001

Mengetahui
Ketua Jurusan Teknik Informatika



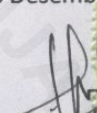
Tri Astoto Kurniawan, S.T, M.T, Ph.D
NIP: 197105182003121001

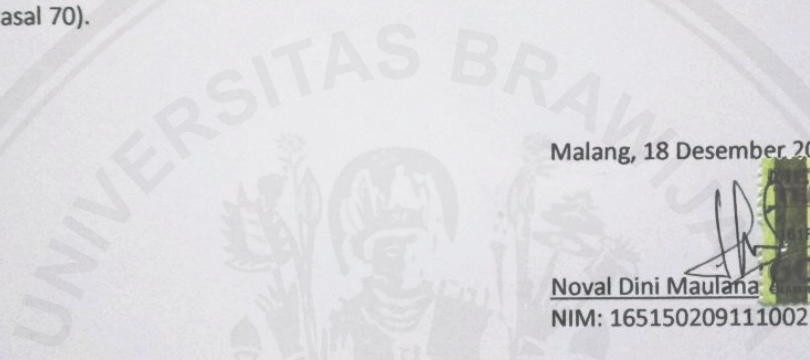
PERNYATAAN ORISINALITAS

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

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

Malang, 18 Desember 2018


Noval Dini Maulana
NIM: 165150209111002



KATA PENGANTAR

Puji syukur penulis panjatkan ke hadirat Allah SWT yang telah memberikan rahmat, karunia dan hidayah-Nya sehingga penulis dapat menyelesaikan skripsi dengan judul “**IMPLEMENTASI METODE *SUPPORT VECTOR REGRESSION (SVR)* DALAM PERAMALAN PENJUALAN ROTI (STUDI KASUS: HARUM BAKERY)**”. Skripsi ini diajukan sebagai ujian seminar skripsi dalam rangka untuk memperoleh gelar Sarjana Komputer di Fakultas Ilmu Komputer (FILKOM), Program Studi Teknik Informatika, Jurusan Teknik Informatika, Universitas Brawijaya, Malang. Atas terselesaikannya skripsi ini, penulis mengucapkan banyak terima kasih kepada:

1. Allah SWT, yang telah memberikan segala hidayah dan rezeki-Nya termasuk kesehatan yang tidak ternilai harganya sehingga penulis mampu menyelesaikan tugas akhir skripsi ini.
2. Budi Darma Setiawan, S.Kom, M.Cs selaku dosen pembimbing pertama skripsi yang telah meluangkan waktu dan juga memberikan banyak pengarahan dan dukungan bagi penulis.
3. Candra Dewi, S.Kom, M.Sc selaku dosen pembimbing kedua skripsi yang telah meluangkan waktu dan juga memberikan banyak pengarahan dan masukan bagi penulis.
4. Wayan Firdaus Mahmudy, S.Si., M.T., Ph.D selaku Dekan Fakultas Ilmu Komputer Universitas Brawijaya.
5. Segenap bapak dan ibu dosen yang telah mendidik dan mengajarkan ilmunya kepada penulis selama menempuh pendidikan di Fakultas Ilmu Komputer Universitas Brawijaya.
6. Kedua Orang Tua, Siadi Prayitno dan Supraptiningsih, atas dukungan dan perhatiannya terhadap penulis selama proses perkuliahan hingga terselesaikannya tugas akhir skripsi ini.
7. Kakak kandung penulis, Shemi Aditya Maulana, Kakak ipar, Fatihatul Maghfiroh, serta keponakan tercinta Sekha Kian Athara yang menjadi motivasi besar dalam penyelesaian tugas akhir skripsi ini.
8. Seluruh teman – teman SAP 2016 Fakultas Ilmu Komputer Universitas Brawijaya yang telah memberi doa, dorongan, dan semangat.
9. Segenap keluarga besar Toko Roti Harum Bakery yang telah mengizinkan saya menggunakan data penjualannya untuk digunakan dalam penelitian tugas akhir skripsi ini.
10. Segenap staff dan karyawan di Fakultas Ilmu Komputer Universitas Brawijaya yang telah banyak membantu penulis dalam pelaksanaan penyusunan skripsi.

Penulis menyadari bahwa dalam menyelesaikan tugas akhir ini, tentunya masih jauh dari kesempurnaan. Maka dari itu kritik dan saran yang sifatnya membangun sangat diharapkan untuk kesempurnaan karya berikutnya. Semoga penyusunan skripsi ini dapat berguna dan bermanfaat bagi penulis maupun pembaca.

Malang, 18 Desember 2018

Penulis

novaldinimaulana@gmail.com



ABSTRAK

Roti merupakan salah satu jenis makanan yang digemari oleh Masyarakat Indonesia. Salah satu bukti pentingnya roti bagi masyarakat Indonesia adalah impor tepung terigu yang terus meningkat. Salah satu perusahaan pembuat roti yang saat ini sedang berkembang adalah Harum Bakery. Kendala yang sering dihadapi Harum Bakery adalah sistem peramalan permintaan pelanggan yang masih manual dan terkesan mengira-ngira. Proses peramalan tersebut sangat jelas berpengaruh besar terhadap proses penjualan. Dengan adanya peramalan penjualan roti, diharapkan dapat membantu toko roti Harum Bakery dalam mempersiapkan bahan baku dan segala sesuatu yang diperlukan untuk pembuatan roti. *Support Vector Regression* (SVR) adalah salah satu metode yang bisa digunakan dalam melakukan peramalan. Data yang digunakan adalah data penjualan roti manis, cake dan tawar dengan tipe data time series dan menggunakan 4 fitur. Pada penelitian ini metode SVR yang digunakan untuk meramal hasil penjualan menghasilkan nilai evaluasi RMSE untuk roti manis sebesar 0,00176, roti cake sebesar 0,00019, dan roti tawar sebesar 0,00010.

Kata Kunci: *Support Vector Regression*, Peramalan, *Time Series*

ABSTRACT

Bread is one of the favorite foods of the Indonesian people, the proof is the increasing import of wheat flour. One of the bakery companies that is currently developing is Harum Bakery. Constraints that are often faced by Harum Bakery are customer demand forecasting systems that are still manual and seem to be guessing. The forecasting process give a big impact on the sales process. With the forecasting of bread sales, it is hoped that Harum Bakery can help bakeries in preparing raw materials and everything needed for bread making. Support Vector Regression (SVR) is one method that can be used in forecasting. The data used is data on sales of sweet bread, cake and white bread with time series data types and uses 4 features. In this study the SVR method used to predict the results of the sale resulted in an evaluation value of RMSE for sweet breads is 0.00176, bread cake is 0.00019, and large breads is 0.00010.

Keyword: Support Vector Regression, Forecasting, Time Series



DAFTAR ISI

PENGESAHAN	ii
PERNYATAAN ORISINALITAS	iii
KATA PENGANTAR.....	iv
ABSTRAK.....	vi
<i>ABSTRACT</i>	vii
DAFTAR ISI.....	viii
DAFTAR TABEL.....	xii
DAFTAR GAMBAR.....	xiv
DAFTAR LAMPIRAN	xvi
BAB 1 PENDAHULUAN.....	1
1.1 Latar belakang.....	1
1.2 Rumusan masalah.....	2
1.3 Tujuan	2
1.4 Manfaat.....	3
1.5 Batasan masalah	3
1.6 Sistematika pembahasan.....	3
BAB 2 LANDASAN KEPUSTAKAAN	5
2.1 Kajian Pustaka	5
2.2 Dasar Teori.....	6
2.2.1 Peramalan	6
2.2.2 Support Vector Regression (SVR).....	7
2.2.3 Algoritme <i>Sequential Learning</i>	7
2.2.4 Fungsi Kernel.....	9
2.2.5 Normalisasi dan Denormalisasi.....	10
2.2.6 <i>Root Mean Square Error (RMSE)</i>	10
BAB 3 METODOLOGI	11
3.1 Tipe Penelitian	11
3.2 Lokasi Penelitian	12
3.3 Pengumpulan Data	12
3.4 Perancangan Sistem.....	12

3.5 Implementasi Algoritme	13
3.6 Kebutuhan Sistem	14
3.7 Pengujian Algoritme SVR	15
3.8 Analisis Hasil / Pembahasan	15
3.9 Penarikan Kesimpulan	15
BAB 4 PERANCANGAN	16
4.1 Desain Algoritme.....	16
4.2 Flowchart	16
4.2.1 Normaliasi	16
4.2.2 Hitung Jarak.....	18
4.2.3 Hitung Kernel RBF	20
4.2.4 Matriks Hessian	21
4.2.5 Hitung Nilai Error.....	22
4.2.6 Hitung nilai delta <i>alpha star</i> dan delta <i>alpha</i>	23
4.2.7 Hitung nilai <i>alpha star</i> dan <i>alpha</i>	24
4.2.8 Fungsi Regresi	25
4.2.9 Denormalisasi.....	26
4.2.10 Evaluasi (<i>RMSE</i>).....	27
4.3 Perhitungan Manual	28
4.3.1 Data Uji / Data <i>Testing</i>	28
4.3.2 Normalisasi Data	29
4.3.3 Inisialisasi Parameter SVR	30
4.3.4 Menghitung Jarak.....	30
4.3.5 Menghitung Kernel	31
4.3.6 Menghitung Matriks Hessian	31
4.3.7 Menghitung Nilai <i>αi</i> * dan <i>αi</i>	32
4.3.8 Menentukan Fungsi Regresi <i>f(x)</i>	35
4.3.9 Proses Denormalisasi	36
4.3.10 Proses Evaluasi (<i>RMSE</i>)	37
4.4 Desain Antarmuka	37
BAB 5 IMPLEMENTASI	39
5.1 Implementasi Sistem	39



5.1.1 Implementasi Proses Normaliasi.....	39
5.1.2 Implementasi Proses Menghitung Kernel RBF.....	41
5.1.3 Implementasi Proses Menghitung Matriks Hessien	41
5.1.4 Implementasi Proses Menghitung Nilai Regresi	42
5.1.5 Implementasi Proses Denormalisasi	43
5.2 Implementasi Antarmuka	44
5.2.1 Antarmuka Halaman <i>Input</i>	44
5.2.2 Antarmuka Halaman <i>Output</i>	45
BAB 6 HASIL DAN PEMBAHASAN	46
6.1 Hasil Pengujian Roti Manis	46
6.1.1 Hasil Pengujian Parameter Nilai <i>Lambda</i> (λ).....	46
6.1.2 Hasil Pengujian Parameter Nilai <i>Sigma</i> (σ).....	47
6.1.3 Hasil Pengujian Parameter Nilai <i>Coefisien Learning Rate</i> (<i>cLR</i>) ..	49
6.1.4 Hasil Pengujian Parameter Nilai <i>Complexity</i> (<i>C</i>)	50
6.1.5 Hasil Pengujian Parameter Nilai <i>Epsilon</i>	52
6.1.6 Hasil Pengujian Jumlah Iterasi.....	53
6.1.7 Kesimpulan Hasil Pengujian	55
6.2 Hasil Pengujian Roti Cake	55
6.2.1 Hasil Pengujian Parameter Nilai <i>Lambda</i> (λ).....	55
6.2.2 Hasil Pengujian Parameter Nilai <i>Sigma</i> (σ).....	57
6.2.3 Hasil Pengujian Parameter Nilai <i>Coefisien Learning Rate</i> (<i>cLR</i>) ..	58
6.2.4 Hasil Pengujian Parameter Nilai <i>Complexity</i> (<i>C</i>)	60
6.2.5 Hasil Pengujian Parameter Nilai <i>Epsilon</i>	61
6.2.6 Hasil Pengujian Jumlah Iterasi.....	63
6.2.7 Kesimpulan Hasil Pengujian	64
6.3 Hasil Pengujian Roti Tawar	65
6.3.1 Hasil Pengujian Parameter Nilai <i>Lambda</i> (λ).....	65
6.3.2 Hasil Pengujian Parameter Nilai <i>Sigma</i> (σ).....	66
6.3.3 Hasil Pengujian Parameter Nilai <i>Coefisien Learning Rate</i> (<i>cLR</i>) ..	68
6.3.4 Hasil Pengujian Parameter Nilai <i>Complexity</i> (<i>C</i>)	69
6.3.5 Hasil Pengujian Parameter Nilai <i>Epsilon</i>	71
6.3.6 Hasil Pengujian Jumlah Iterasi.....	72



6.3.7 Kesimpulan Hasil Pengujian	74
BAB 7 PENUTUP	75
7.1 Kesimpulan.....	75
7.2 Saran	75
DAFTAR PUSTAKA.....	xvii
LAMPIRAN	76



DAFTAR TABEL

Tabel 1.1 Hasil Penjualan dan Sisa Roti Manis pada Bulan Januari 2017	1
Tabel 4.1 Data Latih dan Data Uji.....	28
Tabel 4.2 Data <i>Testing</i> Roti Manis	29
Tabel 4.3 Normalisasi Data Penjualan Roti Manis	29
Tabel 4.4 <i>Range</i> Parameter SVR	30
Tabel 4.5 Contoh nilai Parameter SVR	30
Tabel 4.6 Hasil Hitung Jarak	30
Tabel 4.7 Hasil Hitung Kernel RBF.....	31
Tabel 4.8 Hasil Hitung Matriks Hessian.....	32
Tabel 4.9 Nilai α_i * dan α_i pada iterasi pertama	32
Tabel 4.10 Hasil Hitung Nilai <i>Error</i>	33
Tabel 4.11 Hasil hitung nilai $\delta\alpha_i$ * dan $\delta\alpha_i$	34
Tabel 4.12 Hasil hitung nilai α_i * dan α_i	35
Tabel 4.13 Nilai α_i * dan α_i iterasi ke 2	35
Tabel 4.14 Hasil Hitung Fungsi Regresi $f(x)$	36
Tabel 4.15 Hasil Perhitungan Denormalisasi	37
Tabel 6.1 Hasil Pengujian Nilai Parameter <i>Lambda</i> (λ) untuk Roti Manis	46
Tabel 6.2 Hasil Pengujian Nilai Parameter <i>Sigma</i> (σ) untuk Roti Manis.....	48
Tabel 6.3 Hasil Pengujian Nilai Parameter <i>cLR</i> untuk Roti Manis.....	49
Tabel 6.4 Hasil Pengujian Nilai Parameter <i>Complexity</i> untuk Roti Manis	51
Tabel 6.5 Hasil Pengujian Nilai Parameter <i>Epsilon</i> untuk Roti Manis.....	52
Tabel 6.6 Hasil Pengujian Iterasi untuk Roti Manis.....	54
Tabel 6.7 Perbandingan Nilai Prediksi dan Nilai Aktual Penjualan Roti Manis.....	55
Tabel 6.8 Hasil Pengujian Nilai Parameter <i>Lambda</i> (λ) untuk Roti Cake	56
Tabel 6.9 Hasil Pengujian Nilai Parameter <i>Sigma</i> (σ) untuk Roti Cake.....	57
Tabel 6.10 Hasil Pengujian Nilai Parameter <i>cLR</i> untuk Roti Cake	59
Tabel 6.11 Hasil Pengujian Nilai Parameter <i>Complexity</i> untuk Roti Cake	60
Tabel 6.12 Hasil Pengujian Nilai Parameter <i>Epsilon</i> untuk Roti Cake.....	62
Tabel 6.13 Hasil Pengujian Iterasi untuk Roti Cake.....	63
Tabel 6.14 Perbandingan Nilai Prediksi dan Nilai Aktual Penjualan Roti Cake.....	64

Tabel 6.15 Hasil Pengujian Nilai Parameter *Lambda* (λ) untuk Roti Tawar 65

Tabel 6.16 Hasil Pengujian Nilai Parameter *Sigma* (σ) untuk Roti Tawar 67

Tabel 6.17 Hasil Pengujian Nilai Parameter *cLR* untuk Roti Tawar..... 68

Tabel 6.18 Hasil Pengujian Nilai Parameter *Complexity* untuk Roti Tawar 70

Tabel 6.19 Hasil Pengujian Nilai Parameter *Epsilon* untuk Roti Tawar..... 71

Tabel 6.20 Hasil Pengujian Iterasi untuk Roti Tawar 73

Tabel 6.21 Perbandingan Nilai Prediksi dan Nilai Aktual Penjualan Roti Cake 74



DAFTAR GAMBAR

Gambar 3.1 Blok Diagram Tahapan Penelitian	11
Gambar 3.2 Diagram Alir Proses SVR.....	13
Gambar 3.3 Gambaran Umum Perancangan Sistem	14
Gambar 4.1 Diagram Alir Proses Normalisasi	17
Gambar 4.2 Diagram Alir Proses Hitung Jarak	19
Gambar 4. 3 Diagram Alir Proses Hitung Kernel RBF	20
Gambar 4.4 Diagram Alir Proses Hitung Matriks Hessian.....	21
Gambar 4.5 Diagram Alir Proses Hitung Nilai Error	22
Gambar 4.6 Diagram Alir Proses Hitung <i>Delta Alpha</i> dan <i>Delta Alpha Star</i>	23
Gambar 4.7 Diagram Alir Proses Hitung Nilai <i>Alpha</i> dan <i>Alpha Star</i>	24
Gambar 4.8 Diagram Alir Proses Hoitung Nilai Regresi	25
Gambar 4.9 Diagram Alir Proses Denormalisasi	26
Gambar 4.10 Diagram Alir Proses Hitung RMSE	27
Gambar 4.11 Desain Antarmuka Halaman <i>Input</i>	38
Gambar 4.12 Desain Antarmuka Halaman <i>Output</i>	38
Gambar 5.1 Antarmuka Halaman <i>Input</i>	44
Gambar 5.2 Antarmuka Halaman <i>Output</i>	45
Gambar 6.1 Grafik Nilai RMSE untuk Pengujian Nilai <i>Lambda</i> pada Roti Manis ..	47
Gambar 6.2 Grafik Nilai RMSE untuk Pengujian Nilai <i>Sigma</i> pada Roti Manis	48
Gambar 6.3 Grafik Nilai RMSE untuk Pengujian Nilai <i>cLR</i> pada Roti Manis	50
Gambar 6.4 Grafik Nilai RMSE Pengujian Nilai <i>Complexity</i> pada Roti Manis	51
Gambar 6.5 Grafik Nilai RMSE untuk Pengujian Nilai <i>Epsilon</i> pada Roti Manis ...	53
Gambar 6.6 Grafik Nilai RMSE untuk Pengujian Jumlah Iterasi pada Roti Manis.	54
Gambar 6.7 Grafik Nilai RMSE untuk Pengujian <i>Lambda</i> pada Roti Cake	56
Gambar 6.8 Grafik Nilai RMSE untuk Pengujian <i>Sigma</i> pada Roti Cake	58
Gambar 6.9 Grafik Nilai RMSE untuk Pengujian Nilai <i>cLR</i> pada Roti Cake.....	59
Gambar 6.10 Grafik Nilai RMSE Pengujian Nilai <i>Complexity</i> pada Roti Cake	61
Gambar 6.11 Grafik Nilai RMSE untuk Pengujian Nilai <i>Epsilon</i> pada Roti cake	62
Gambar 6.12 Grafik Nilai RMSE untuk Pengujian Jumlah Iterasi pada Roti Cake.	64
Gambar 6.13 Grafik Nilai RMSE untuk Pengujian <i>Lambda</i> pada Roti Tawar	66

Gambar 6.14 Grafik Nilai RMSE untuk Pengujian *Sigma* pada Roti Tawar 67

Gambar 6.15 Grafik Nilai RMSE untuk Pengujian Nilai *cLR* pada Roti Tawar 69

Gambar 6.16 Grafik Nilai RMSE Pengujian Nilai *Complexity* pada Roti Tawar 70

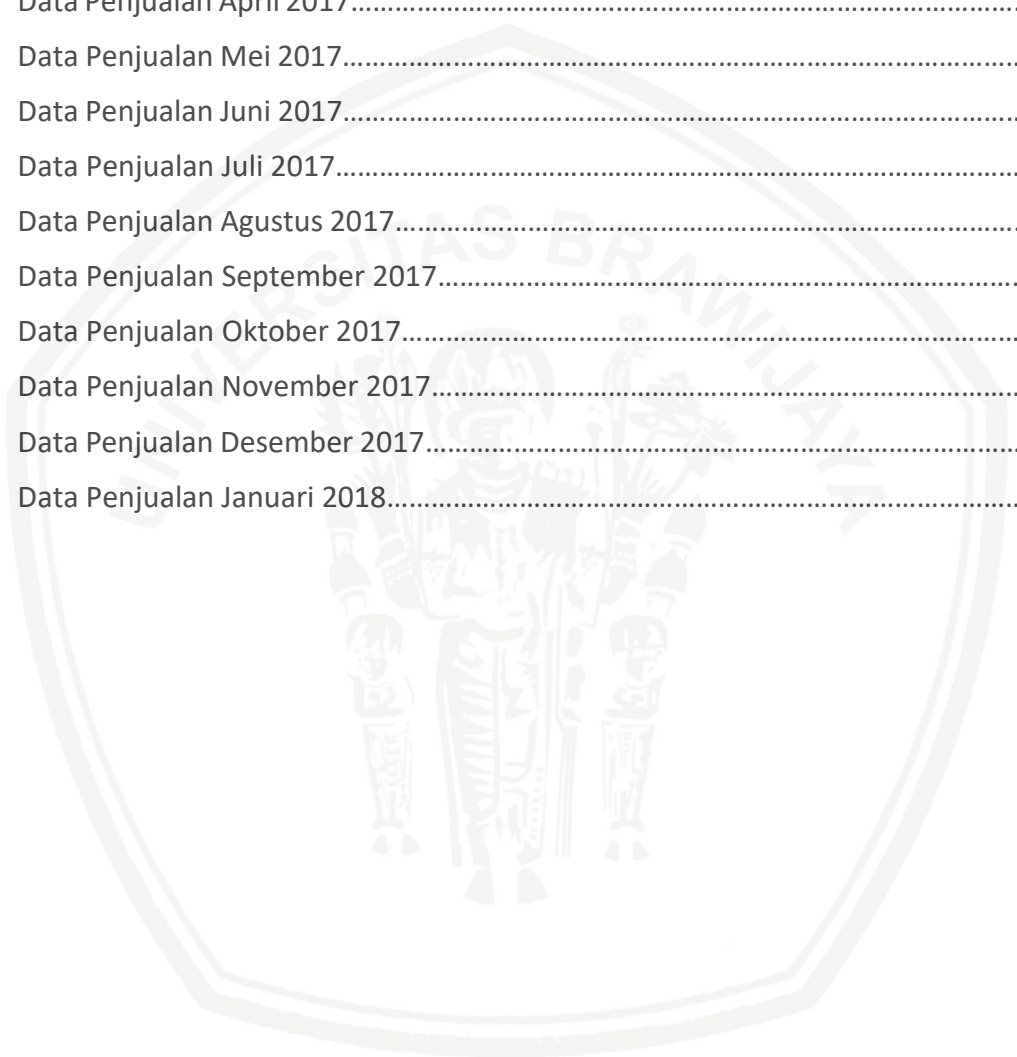
Gambar 6.17 Grafik Nilai RMSE untuk Pengujian Nilai *Epsilon* pada Roti Tawar . 72

Gambar 6.18 Grafik Nilai RMSE Pengujian Jumlah Iterasi pada Roti Tawar 73



DAFTAR LAMPIRAN

Data Penjualan Januari 2017.....	76
Data Penjualan Februari 2017.....	77
Data Penjualan Maret 2017.....	78
Data Penjualan April 2017.....	79
Data Penjualan Mei 2017.....	80
Data Penjualan Juni 2017.....	81
Data Penjualan Juli 2017.....	82
Data Penjualan Agustus 2017.....	84
Data Penjualan September 2017.....	85
Data Penjualan Oktober 2017.....	86
Data Penjualan November 2017.....	87
Data Penjualan Desember 2017.....	88
Data Penjualan Januari 2018.....	89



BAB 1 PENDAHULUAN

1.1 Latar belakang

Roti merupakan salah satu jenis makanan yang digemari oleh Masyarakat Indonesia. Salah satu bukti pentingnya roti bagi masyarakat Indonesia adalah impor tepung terigu yang terus meningkat. Salah satu perusahaan pembuat roti yang saat ini sedang berkembang adalah Harum Bakery. Harum Bakery ini terletak di Kabupaten Malang, tepatnya di Jalan Raya Sengkaling no. 217. Kendala yang sering dihadapi Harum Bakery adalah sistem peramalan permintaan pelanggan yang masih manual dan terkesan mengira-ngira. Proses peramalan tersebut sangat jelas berpengaruh besar terhadap proses penjualan, dimana perusahaan harus cepat dalam menyiapkan bahan baku dan proses pembuatan roti apabila ada permintaan pelanggan dalam jumlah besar. Salah satu contoh peramalan permintaan pelanggan yang salah oleh Harum Bakery ditunjukkan pada Tabel 1.1 berikut:

Tabel 1.1 Hasil Penjualan dan Sisa Roti Manis pada Bulan Januari 2017

Tanggal	Manis	Sisa
1	110	10
2	90	15
3	123	2
4	200	0
5	102	8
6	134	1
7	142	0

Oleh karena hal itu, dibutuhkan proses peramalan permintaan pelanggan yang lebih ilmiah dan akurat, sehingga dapat berdampak efektif terhadap stok bahan baku dan dapat memaksimalkan hasil penjualan.

Ada beberapa metode peramalan dengan model peramalan kuantitatif, salah satunya dengan metode *Support Vector Regression*. Metode *Support Vector Regression* yang diterapkan oleh Sethu Vijayakumar dan Si Wu (1999) untuk menyelesaikan permasalahan peramalan dalam bentuk regresi (Vijayakumar & Wu, 1999). Metode *Support Vector Regression* telah banyak digunakan untuk membantu peneliti di bagian peramalan atau peramalan dan terbukti menghasilkan peramalan dengan tingkat kesalahan cukup rendah. Serta pada algoritme SVR cocok menggunakan data yang nilainya acak atau data yang bermodel non-linear.

Dalam kasus peramalan atau peramalan seperti masalah diatas, terdapat penelitian yang pernah dilakukan oleh M. Raabith Rifqi, dengan penelitiannya yang berjudul “Support Vector Regression Untuk Peramalan Permintaan Darah: Studi Kasus Unit Transfusi Darah Cabang – PMI Kota Malang”. Pada penelitian tersebut digunakan *input* berupa sata permintaan darah bulan September 2013 – Agustus 2014 yang kemudian diproses menggunakan metode *Support Vector Regression* dan meghasilkan nilai MAPE yang paling minimum yaitu 3,899% (Rifqi, 2018). Penelitian lainnya yang pernah dilakukan menggunakan metode *SVR* ini adalah penelitian yang pernah dilakukan oleh Mimin Putri Raharyani dengan judul penelitian “Implementasi Algoritme Support Vector Regression pada Peramalan Jumlah pengunjung Pariwisata” dengan *input* berupa Data jumlah pengunjung pada bulan Januari 2013 hingga Desember 2013, menghasilkan nilai MAPE kurang dari 10% dan dapat dikategorikan baik untuk memperamalan jumlah pengunjung pariwisata. Berdasarkan hasil pengujian pengaruh data yang dinormalisasi dan data yang tidak dinormalisasi terhadap proses algoritme *SVR*, rata-rata nilai MAPE minimum yang diperoleh adalah 5,93% dengan data yang dinormalisasi dan rata-rata nilai MAPE minimum 3,98% untuk data yang tidak dinormalisasi. Nilai MAPE tersebut dapat disimpulkan bahwa normaliasi data belum tentu menghasilkan hasil peramalan yang lebih baik (Raharyani, 2017).

Berdasarkan latar belakang diatas, penulis ingin menuliskan sebuah penelitian dengan judul “Implementasi Metode *Support Vector Regression (SVR)* dalam Peramalan Penjualan Roti (Studi Kasus: Harum Bakery)”, yang diharapkan dapat membantu perusahaan tersebut dalam memperamalan permintaan pelanggan kedepannya, sehingga dapat memaksimalkan proses dan hasil penjualan.

1.2 Rumusan masalah

Berdasarkan latar belakang yang telah penulis paparkan, maka penulis merumuskan beberapa rumusan masalah yang akan diselesaikan dalam penelitian ini yaitu :

1. Berapa nilai *coefisien Learning Rate, Complexity, Gamma, Lambda*, dan *sigma* yang menghasilkan akurasi terbaik.
2. Bagaimana akurasi sistem peramalan penjualan roti dengan metode *Support Vector Regression (SVR)* di Perusahaan Harum Bakery jika ditinjau dari tingkat kesalahan menggunakan perhitungan *Root Mean Square Error (RMSE)*.

1.3 Tujuan

Tujuan dari penelitian dari penelitian ini adalah :

1. Menerapkan metode *Support Vector Regression (SVR)* untuk melakukan peramalan pada penjualan roti di Harum bakery.
2. Menghitung nilai tingkat kesalahan peramalan penjualan roti di Harum Bakery menggunakan *Root Mean Square Error (RMSE)*.

1.4 Manfaat

Manfaat dari penelitian ini adalah :

1. Dapat menyimpulkan apakah metode *Support Vector Regression (SVR)* cocok digunakan untuk tipe data seperti pada penelitian ini.
2. Dengan adanya hasil dari penelitian ini, diharapkan dapat berguna bagi Harum Bakery sehingga dapat memaksimalkan proses dan hasil penjualan.

1.5 Batasan masalah

Untuk menghindari pembahasan yang terlalu luas, maka dari rumusan masalah penulis membatasi penelitian ini dengan ketentuan sebagai berikut:

1. Ada tiga jenis roti yang peneliti gunakan, yaitu roti manis, roti tawar, dan roti cake.
2. Data yang digunakan dalam penelitian ini adalah data penjualan roti per hari selama kurun waktu satu tahun (Januari 2018 sampai dengan Desember 2018), yang kemudian menghasilkan 1095 data penjualan dengan rincian 365 data penjualan roti manis, 365 data penjualan roti cake, dan 365 data penjualan roti tawar.
3. *Output* yang dihasilkan merupakan peramalan penjualan roti tawar, roti manis, dan roti cake pada setiap hari pada bulan Januari 2018.

Hasil akhir perhitungan peramalan penjualan roti ini tidak dibandingkan dengan metode peramalan lainnya.

1.6 Sistematika pembahasan

Sistematika pembahasan yang diterapkan pada penelitian ini adalah sebagai berikut:

BAB I Pendahuluan

Berisi tentang uraian umum yang terdiri dari latar belakang, rumusan masalah, batasan masalah, tujuan penelitian, manfaat bagi peneliti dan perusahaan, dan metodologi penulisan laporan penelitian ini.

BAB II Tinjauan Pustaka

Membahas tentang teori-teori yang digunakan dalam penelitian serta kajian pustaka yang mendukung metode yang penulis gunakan demi terwujudnya Implementasi Metode *Support Vector Regression (SVR)* dengan studi kasus Harum Bakery.

BAB III Metodologi

Membahas tentang metode-metode yang digunakan dalam penelitian Implementasi Metode *Support Vector Regression (SVR)* dalam Peramalan Penjualan Roti (Studi Kasus: Harum Bakery).

BAB IV Perancangan

Membahas tentang analisis dari Pengembangan Sistem untuk Peramalan Penjualan Roti menggunakan Metode *Support Vector Regression (SVR)* (Studi Kasus: Harum Bakery). Serta merancang kebutuhan sistem yang sesuai dengan kebutuhan tersebut.

BAB V Implementasi

Membahas tentang hasil rancangan dari analisis kebutuhan serta implementasi dari rancangan tersebut pada Implementasi Metode *Support Vector Regression (SVR)* dalam Peramalan Penjualan Roti (Studi Kasus: Harum Bakery).

BAB VI Pembahasan

Memaparkan tentang proses dan pengujian sistem dari Implementasi Metode *Support Vector Regression (SVR)* dalam Peramalan Penjualan Roti (Studi Kasus: Harum Bakery).

BAB VII Penutup

Memaparkan kesimpulan dari penelitian Implementasi Metode *Support Vector Regression (SVR)* dalam Peramalan Penjualan Roti (Studi Kasus: Harum Bakery) yang telah diselesaikan, serta saran dari penulis untuk peneliti atau penulis yang ingin mengembangkan penelitian ini lebih lanjut.

BAB 2 LANDASAN KEPUSTAKAAN

Kajian pustaka yang digunakan pada penelitian ini antara lain: Support Vector Regression Untuk Peramalan Permintaan Darah: Studi Kasus Unit Transfusi Darah Cabang – PMI Kota Malang (Rifqi, 2018); Normalisasi SVR dengan Ant Colony Optimization untuk peramalan tingkat produksi susu segar (Studi Kasus pada koperasi susu Sae Pujon, Malang) (Susanto, 2018); Peramalan Jumlah Kunjungan Wisatawan Mancanegara ke Bali Menggunakan Support Vector Regression dengan Algoritme Genetika (Surtiningsih, 2017); Implementasi Algoritme Support Vector Regression pada Peramalan Jumlah pengunjung Pariwisata (Raharyani, 2017). Dasar Teori yang digunakan berdasarkan latar belakang dan topik permasalahan adalah Peramalan, Roti, *Support Vector Regression (SVR)*, *Algoritme Sequential Learning*, Fungsi Kernel, Normalisasi dan Denormalisasi, dan *Mean Square Error (MSE)*.

2.1 Kajian Pustaka

Kajian pustaka yang digunakan adalah penelitian yang pernah dilakukan sebelumnya dengan menggunakan metode yang sama tetapi tentunya dengan topik yang berbeda. Pembahasan kajian pustaka ini meliputi masukan (*input*), metode, dan hasil akurasi yang didapatkan.

Kajian pustaka yang pertama adalah penelitian yang dilakukan oleh M. Raabith Rifqi pada tahun 2018 dengan judul Support Vector Regression Untuk Peramalan Permintaan Darah: Studi Kasus Unit Transfusi Darah Cabang – PMI Kota Malang. Masukan (*input*) yang digunakan adalah Data permintaan darah bulan September 2013 – Agustus 2014. Evaluasi kinerja algoritme SVR pada penulisan ini menggunakan MAPE (Mean Absolute Percentage Error), dan dihasilkan nilai MAPE yang paling minimum yaitu 3,899%, dengan nilai $\lambda = 10$, $\sigma = 0,5$, $cLR = 0,01$, $\epsilon = 0,01$, jumlah fitur data = 4 dan jumlah iterasi sebanyak 5000 yang menggunakan data testing sebanyak 12 data. Penelitian selanjutnya dilakukan oleh Karuniawan Susanto pada tahun 2018 dengan judul Normalisasi SVR dengan *Ant Colony Optimization* untuk Peramalan Tingkat Produksi Susu Segar (Studi Kasus pada koperasi susu Sae Pujon, Malang). *Input* yang digunakan dalam penelitian ini adalah Data penjualan susu segar mulai bulan Januari 2014 – Desember 2015. Sistem peramalan produksi susu segar koperasi susu Sae Pujon menggunakan metode SVR dan *Ant Colony Optimization* menghasilkan error rate nilai MAPE sebesar 3,30425%. Hasil tersebut dapat disimpulkan memiliki akurasi tinggi karena nilai MAPE kurang dari 10%. Selanjutnya penelitian yang dilakukan oleh Listiya Surtiningsih pada tahun 2017, judul yang digunakan adalah Peramalan Jumlah Kunjungan Wisatawan Mancanegara ke Bali Menggunakan Support Vector Regression dengan Algoritme Genetika. *Input* dari penelitian ini adalah Data Jumlah kunjungan wisatawan mancanegara (WisMan) ke Bali per bulan dari Januari 2001 hingga Desember 2001. Berdasarkan pengujian yang telah dilakukan dengan menggunakan data jumlah kunjungan wisatawan mancanegara ke Bali pada tahun 2001 hingga tahun 2016, didapatkan parameter terbaik dari $\lambda =$

1 – 10, kompleksitas (C) = 1 – 100, Epsilon = 0,00001 – 0,001, Gamma = 0,00001 – 0,001, Sigma = 0,01 – 3,5, Iterasi SVR = 1250, generasi GA = 90, Populasi = 70, Kombinasi Crossover Rate dan Mutation Rate = 0,6 dan 0,4, jumlah fitur 2, dan jumlah periode peramalan 1 bulan. Nilai error rate / MAPE terkecil didapatkan yaitu sebesar 2,513%. Terakhir penelitian yang dilakukan oleh Mimin Putri Raharyani pada tahun 2017 dengan judul penelitian Implementasi Algoritme Support Vector Regression pada Peramalan Jumlah pengunjung Pariwisata. *Input* yang digunakan adalah Data jumlah pengunjung pada bulan Januari 2013 hingga Desember 2013. Nilai MAPE yang dihasilkan < 10% dan dapat dikategorikan baik untuk memperamalan jumlah pengunjung pariwisata. Berdasarkan nilai MAPE tersebut rata-rata selisih antara hasil peramalan dengan data aktual adalah sebesar 115 jumlah pengunjung. Berdasarkan hasil pengujian pengaruh data yang dinormalisasi dan data yang tidak dinormalisasi terhadap proses algoritme SVR, rata-rata nilai MAPE minimum yang diperoleh adalah 5,93% dengan data yang dinormalisasi dan rata-rata nilai MAPE minimum 3,98% untuk data yang tidak dinormalisasi. Nilai MAPE tersebut dapat disimpulkan bahwa normalisasi data belum tentu menghasilkan hasil peramalan yang lebih baik.

2.2 Dasar Teori

2.2.1 Peramalan

Menurut Makridakis (1991), peramalan (*forecasting*) adalah peramalan nilai-nilai dari sebuah peubah berdasarkan terhadap nilai yang diketahui dari peubah tersebut atau dari sebuah peubah yang berhubungan. Meramal atau memperamalan juga bisa berdasarkan kepada keahlian keputusan, yang juga didasarkan kepada data yang sudah ada sebelumnya.

Sedangkan menurut Heizer dan Render (2009), peramalan atau peramalan adalah sebuah seni dan ilmu untuk memperamalkan kejadian di masa mendatang. Peramalan atau peramalan dapat dilakukan dengan melibatkan data masa lalu (*historis*) lalu memproyeksikannya ke masa depan dengan suatu bentuk model yang matematis.

Dalam dunia bisnis, peramalan permintaan pelanggan atau *customer* merupakan suatu hal yang sangat penting, sehingga bisa memangkas biaya produksi dan meningkatkan hasil penjualan. Perencanaan kapasitas produksi yang fleksibel adalah perencanaan kapasitas produksi yang sesuai dengan besarnya kebutuhan permintaan. Perusahaan akan mengalami kerugian apabila kapasitas produksi yang direncanakan terlalu besar sehingga melebihi kebutuhan permintaan yang sebenarnya. Melakukan analisis dan mengestimasi penjualan (*sales forecasting*) merupakan salah satu kegiatan yang sangat penting bagi perusahaan dalam menentukan jumlah produksi yang disesuaikan dengan kapasitas produksi yang dimiliki perusahaan. Selain itu peramalan penting artinya karena dengan peramalan yang tepat-guna diharapkan akan meningkatkan efisiensi produksi (Freddy Rangkuti, 2005).

2.2.2 Support Vector Regression (SVR)

Algoritme *Support Vector Regression (SVR)* adalah teori yang diadaptasi dari teori *machine learning* yang sudah digunakan untuk memecahkan masalah klasifikasi, yaitu *Support Vector Machine (SVM)*. SVR ini adalah penerapan algoritme SVM untuk kasus regresi. Pada SVM merupakan penerapan teori *machine learning* kasus klasifikasi yang menghasilkan nilai bulat atau nilai diskrit, sedangkan pada algoritme SVR yaitu untuk penerapan kasus regresi yang menghasilkan output berupa bilangan riil atau kontinu (Furi, Jordi, & Saepudin, 2015). Konsep algoritme SVR memiliki sebuah kemampuan untuk mengatasi masalah *overfitting*, sehingga berdampak menghasilkan nilai peramalan yang bagus (Furi, Jordi, & Saepudin, 2015). *Overfitting* adalah perilaku data yang pada saat fase training menghasilkan nilai akurasi peramalan yang hampir sempurna atau sama dengan nilai aktual data (Yasin, Prahutama, & Utami, 2014). Tujuan algoritme SVR adalah menemukan suatu garis pemisah atau yang bisa disebut dengan *Hyperplane* terbaik. *Hyperplane* terbaik dapat ditemukan dari mengukur margin dari *hyperplane* tersebut. Margin merupakan jarak antara *hyperplane* dengan data terdekat. Data yang terdekat dari margin disebut dengan *support vector* (Furi, Jordi, & Saepudin, 2015).

Dasar ide algoritme *Support Vector* untuk estimasi regresi adalah menghitung nilai fungsi linier, dimana α_i , α_i^* adalah pengali *Lagrange non-negative*. Solusi untuk masalah ini secara tradisional diperoleh dengan menggunakan paket pemrograman kuadrat. Permukaan aproksimasi optimal menggunakan formulasi yang telah dimodifikasi, setelah memperpanjang SVM menjadi *Non-linear* ditunjukkan dengan persamaan berikut:

$$F(x) = \sum_{i=1}^1 (\alpha_j^* - \alpha_j) (K(x_i, x) + \lambda^2) \quad (2.1)$$

Seperti halnya kasus klasifikasi, hanya beberapa koefisien ($\alpha_j^* - \alpha_j$) yang hasilnya tidak bernilai nol, titik data yang sesuai disebut *Support Vector* (Vijayakumar & Wu, 1999)

2.2.3 Algoritme *Sequential Learning*

Menurut Vijayakumar & Wu (1999), proses *Sequential Learning* adalah proses yang terdapat di setiap perhitungan fungsi SVR, dimana proses ini berguna untuk mendapatkan garis pemisah / *hyperplane* yang optimal. Berikut adalah langkah-langkahnya: (Vijayakumar & Wu, 1999).

1. Inisialisasi $\alpha_i = 0$, $\alpha_i^* = 0$, Hitung matriks R_{ij}

$$R_{ij} = (K(x_i, x) + \lambda^2) \text{ untuk } i, j = 1, \dots, n \quad (2.2)$$

Keterangan:

R_{ij} = matriks *hessian*

K = fungsi kernel

X_i = data ke – i

X_j = data ke – j

λ = Variabel Skalar

Parameter *lambda* (λ) atau variabel skalar menunjukkan ukuran skalar untuk pemetaan ruang pada kernel SVR (Vijayakumar & Wu, 1999).

2. Untuk setiap data training, $i = 1$ sampai n dihitung:

$$a. E_i = y_i - \sum_{j=1}^1 (\alpha_j^* - \alpha_j) R_{ij} \quad (2.3)$$

$$b. \delta\alpha_i^* = \min\{\max[\gamma (E_i - \epsilon), -\alpha_i^*], C - \alpha_i^*\}$$

$$\delta\alpha_i = \min\{\max[\gamma (-E_i - \epsilon), -\alpha_i], C - \alpha_i\} \quad (2.4)$$

$$c. \alpha_i^* = \alpha_i^* + \delta\alpha_i^*$$

$$\alpha_i = \alpha_i + \delta\alpha_i \quad (2.5)$$

Keterangan:

E_i = Nilai Error

y_i = Nilai data normalisasi

α_i^* = Lagrange multiplier

α_i = Lagrange multiplier

R_{ij} = Matriks Hessian

$\delta\alpha_i^*$ = Variabel tunggal, bukan bentuk dari perkalian δ dengan α_i^*

$\delta\alpha_i$ = Variabel tunggal, bukan bentuk dari perkalian δ dengan α_i

γ = Nilai *Learning rate*

ϵ = Parameter *epsilon*

C = Parameter kompleksitas

$$\gamma = \frac{cLR}{\text{Max (Matrik Hessian)}} \quad (2.6)$$

Keterangan:

γ = Nilai *Learning rate*

cLR = Nilai *coefisien Learning rate*

Parameter untuk menghitung nilai $\delta\alpha_i^*$ dan $\delta\alpha_i$ menggunakan 3 parameter yakni nilai parameter *gama* (γ), *epsilon* (ϵ), dan *Complexity* (C). Parameter *gama* (γ) merupakan nilai *learning rate*, untuk mendapatkan nilai *gama* harus memakai nilai parameter cLR (*coefisien Learning rate*). Parameter cLR merupakan laju

pembelajaran (Vijayakumar & Wu, 1999). Selanjutnya untuk parameter yang digunakan untuk menghitung nilai $\delta\alpha_i^*$ dan $\delta\alpha_i$ yakni parameter *epsilon* (ϵ). Parameter *epsilon* (ϵ) digunakan dalam mengukur batas kesalahan fungsi $f(x)$, nilai tersebut menyelubungi nilai dari fungsi $f(x)$ sehingga akan membentuk yang disebut daerah *error – zone*. Dan jika nilai $f(x)$ melebihi *error – zone* yang terbentuk maka akan dikenai penalti sebesar dari parameter C yang telah diatur. Parameter C berfungsi merepresentasikan batas penalti toleransi terhadap kesalahan peramalan, semakin besar nilai parameter C menjadikan model peralaman semakin tidak mentoleransi kesalahan sehingga memberikan nilai peramalan yang baik (Furi, Jordi, & Saepudin, 2015).

3. Kembali ke langkah kedua sampai kondisi iterasi maksimum atau $\max(|\delta\alpha_i|) < \epsilon$ dan $\max(|\delta\alpha_i^*|) < \epsilon$

4. Fungsi regresinya yaitu

$$f(x) = \sum_{j=1}^n (\alpha_j^* - \alpha_j) (K(x_i, x_j) + \lambda^2) \tag{2.7}$$

Keterangan:

X_i = data ke – i

X_j = data ke – j

λ = Variabel Skalar

5. Selesai.

2.2.4 Fungsi Kernel

Untuk mendukung menyelesaikan permasalahan non-linier dengan algoritme SVR, maka digunakanlah fungsi kernel. Untuk memecahkan masalah linear dalam ruang dimensi tinggi, yang harus dilakukan adalah mengganti *inner product* (x_i dan x_j) dengan fungsi kernel. keunggulan dari penggunaan fungsi kernal ini yaitu mampu berhubungan dengan ruang fitur berdimensi lebih tinggi tanpa perlu menghitung pemetaan secara eksplisit (Furi, Jodi, & Saepudin, 2015). Kinerja dari algoritme SVR ditentukan oleh jenis fungsi kernel yang akan digunakan dan pengaturan parameter kernel (Che, & Wang, 2014). Fungsi yang sering digunakan yakni fungsi kernel *Radial Basis Function* (RBF) Kernel dengan persamaan yang ditunjukkan pada persamaan 2.8 berikut (Furi, Jodi, & Saepudin, 2015):

$$K(x_i, x_j) = \exp\left(-\frac{1}{2\sigma^2}\right)(x_i - x_j) \tag{2.8}$$

Keterangan:

X_i = Data ke – i

X_j = Data ke – j

σ = Standart deviasi

Parameter *sigma* (σ) merupakan konstanta dari fungsi kernel *Gaussian* RBF untuk mengatur persebaran data kedalam dimensi fitur yang lebih tinggi (Furi, Jondri, & Saepudin, 2015).



2.2.5 Normalisasi dan Denormalisasi

Normalisasi atau normalisasi data adalah suatu cara mengubah data menjadi nilai yang memiliki kekuatan sama besar (Patel & Mehta, 2011). Normalisasi bertujuan untuk mendapatkan data dengan ukuran lebih kecil yang mewakili data yang asli tanpa kehilangan karakteristiknya sendiri. Rumus normalisasi seperti ditunjukkan pada persamaan 2.9 berikut (Parto & Sahu, 2015):

$$\text{Normalisasi} = \frac{x - \text{min}}{\text{max} - \text{min}} \quad (2.9)$$

Keterangan:

- x = data ke- i
- Min = data minimum dari data yang digunakan
- Max = data maksimum dari data yang digunakan

Sedangkan denormalisasi adalah proses mengembalikan data ke awal yang sebelumnya telah dilakukan normalisasi untuk mendapatkan data yang asli. Proses denormalisasi dilakukan pada hasil akhir atau *output* dari pelatihan peramalan. Rumus denormalisasi seperti ditunjukkan pada persamaan 2.10 berikut:

$$\text{Denormalisasi} = y(\text{max} - \text{min}) + \text{min} \quad (2.10)$$

Keterangan:

- y = hasil *output* dari pelatihan
- Min = data minimum
- Max = data maximum

2.2.6 Root Mean Square Error (RMSE)

Root Mean Square Error sangat populer untuk menilai algoritma mesin pembelajaran, termasuk algoritma yang jauh lebih canggih dari regresi linier (Conway & White, 2012). Nilai RMSE digunakan untuk membedakan kinerja model dalam periode kalibrasi dengan periode validasi serta untuk membandingkan kinerja model individual dengan model prediksi lainnya (Hosseini, Javaherian, & Movahed, 2014).

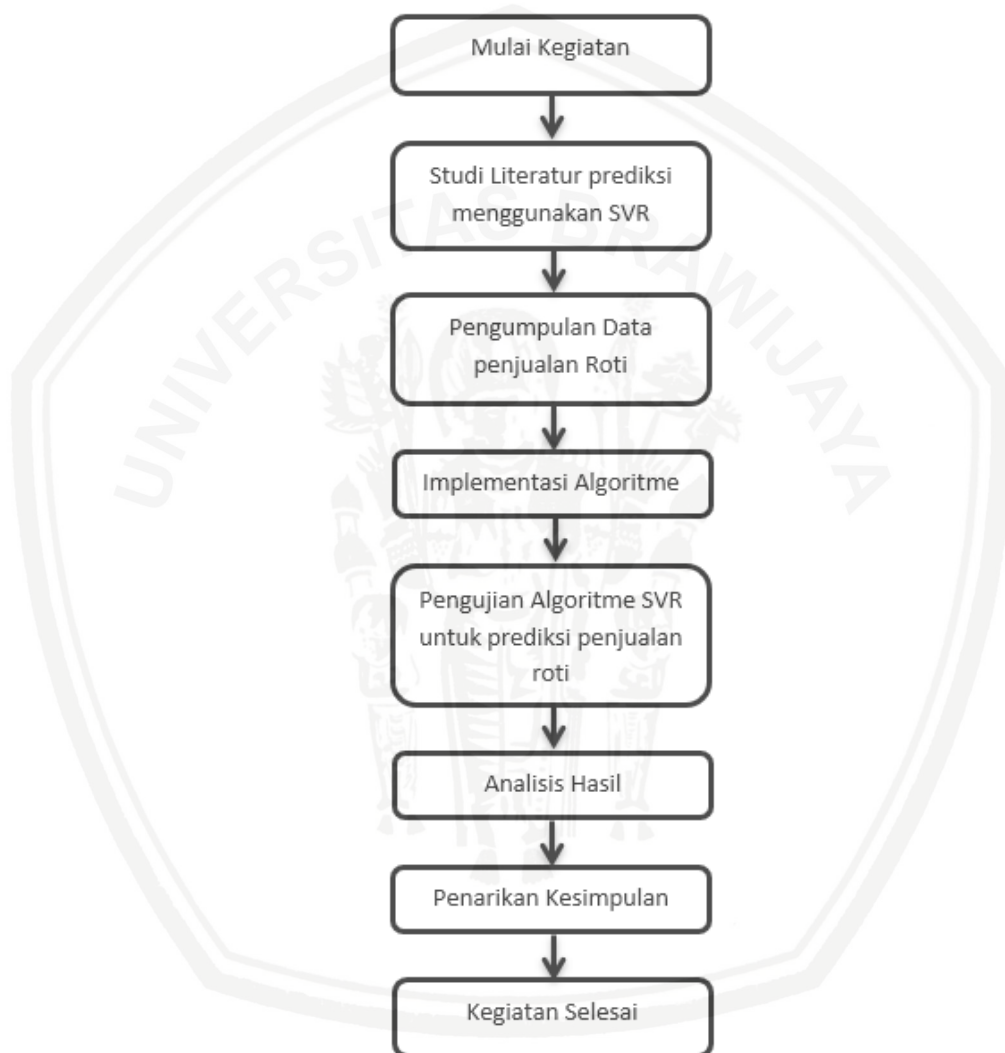
$$\text{RMSE} = \sqrt{\frac{\sum_{j=i}^n (y' - y)^2}{n}} \quad 2.11$$

Keterangan:

- n = Jumlah data.
- e = Error.
- y' = Nilai *output* (prediksi).
- y = Nilai aktual.

BAB 3 METODOLOGI

Pada bab metodologi ini, berisi tentang metode penelitian serta perancangan dari sistem yang akan dibangun. Langkah-langkah dalam penyusunan penelitian ini meliputi studi literatur, analisis kebutuhan sistem, pengumpulan data, perancangan sistem, implementasi sistem, pengujian sistem, dan penarikan kesimpulan. Untuk lebih jelasnya tahapan penelitian ditunjukkan dalam gambar 3.1



Gambar 3.1 Blok Diagram Tahapan Penelitian

3.1 Tipe Penelitian

Tipe penelitian yang sedang dilakukan merupakan kategori *non implementatif*, karena fokus terhadap fenomena yang sedang dikaji untuk menghasilkan analisis ilmiah. Metode yang digunakan untuk menghasilkan

analisis ilmiah berupa observasi, studi kasus, eksperimen, dan penelitian tindakan (*action research*).

Dari kegiatan penelitiannya, pendekatan pada penelitian ini merupakan analitik dari kegiatan penelitian *non implementatif* untuk menjelaskan hubungan antara fenomena tertentu dengan objek penelitian. Penelitian dengan pendekatan tersebut memiliki tujuan untuk menjelaskan relasi atau hubungan antara komponen dalam penelitian dengan kondisi tertentu yang sedang diteliti. Keluaran yang dihasilkan dari penelitian ini adalah hasil analisis.

3.2 Lokasi Penelitian

Penelitian yang dilakukan bertempat di laboratorium komputer cerdas, Fakultas Ilmu Komputer, Universitas Brawijaya yang berlokasi di Jl. Veteran, Kota Malang, Jawa Timur.

3.3 Pengumpulan Data

Data yang digunakan dalam penelitian ini merupakan data penjualan roti pada toko roti Harum Bakery yang terletak di Jalan Raya Sengkaling no. 217, Malang, Jawa Timur. Data didapatkan dengan cara observasi langsung ke toko roti Harum Bakery. Data penjualan meliputi 3 jenis roti terhitung mulai tanggal 1 Januari 2017 sampai dengan 31 Desember 2017, yang terkumpul sebanyak 1095 data, dengan rincian sebagai berikut:

1. 365 data penjualan roti manis
2. 365 data penjualan roti cake
3. 365 data penjualan roti tawar

Data yang digunakan merupakan jenis data *time series*, dengan menggunakan fitur data penjualan 4 hari sebelumnya untuk memprediksi penjualan hari selanjutnya.

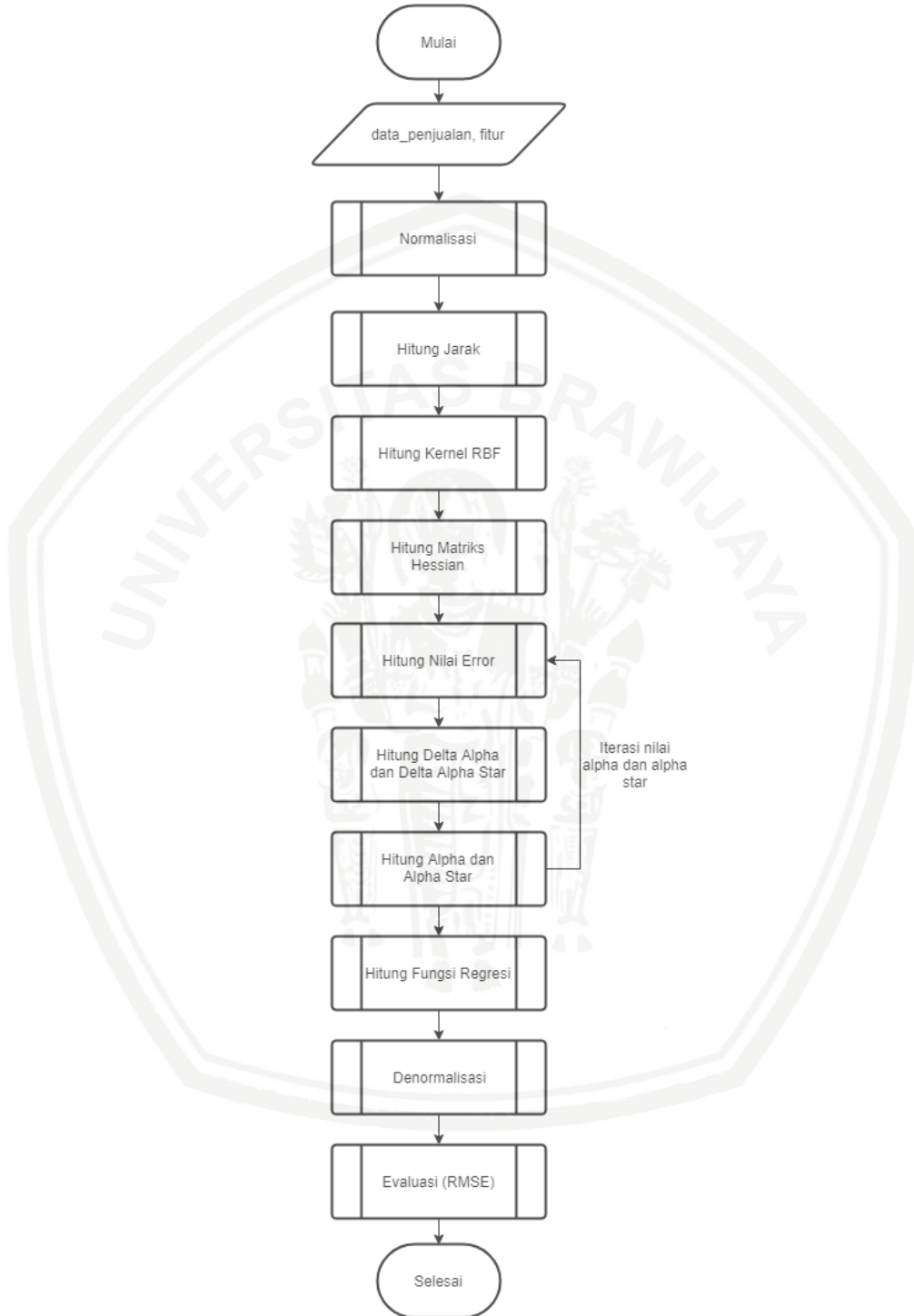
3.4 Perancangan Sistem

Perancangan sistem dilakukan untuk mempermudah implementasi pengujian serta analisis. Langkah-langkah yang dilakukan dalam perancangan sistem adalah sebagai berikut:

1. Diagram alir (*flowchart*) sistem
2. Perhitungan manual
3. Perancangan antarmuka
4. Perancangan pengujian

3.5 Implementasi Algoritme

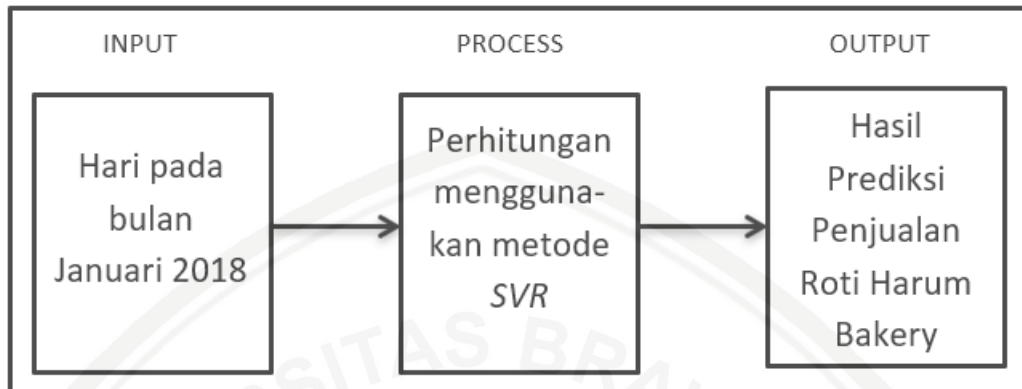
Pada subbab Implementasi Algoritme ini dijelaskan mengenai proses atau langkah-langkah dalam perhitungan metode SVR, yang akan dijabarkan dalam gambar 3.2 dibawah ini:



Gambar 3.2 Diagram Alir Proses SVR

3.6 Kebutuhan Sistem

Pada bab kebutuhan sistem ini berisi gambaran umum kerja sistem secara menyeluruh. Perancangan ini membahas tentang masukan (*input*), proses (*process*), dan keluaran (*output*) secara garis besar, sehingga dapat mempermudah proses implementasi kedalam sistem yang akan dibuat. Perancangan sistem ditunjukkan pada gambar 3.2 berikut.



Gambar 3.3 Gambaran Umum Perancangan Sistem

1. Masukan (*input*)
Masukan pada sistem berupa hari yang akan dilakukan peramalan menggunakan metode *Support Vector Regression (SVR)*.
2. Proses (*process*)
Sistem menghitung menggunakan metode *Support Vector Regression (SVR)*, yang didalamnya terdapat proses menghitung normalisasi, hitung jarak, kernel RBF, dan matriks hessian.
3. Keluaran (*output*)
Sistem menunjukkan hasil peramalan penjualan atau permintaan pelanggan pada hari tersebut.

Ada pula kebutuhan fungsional yang diperlukan untuk menjalankan sistem, antara lain:

1. Kebutuhan perangkat keras (*hardware*), meliputi:
 - Laptop dengan *Processor Intel® Core™ i5-3337U CPU @ 1.80GHz (4 CPUs), ~1.8GHz*
2. Kebutuhan perangkat lunak (*software*), meliputi:
 - Sistem Operasi Windows 10 Pro-64-bit
 - Google Chrome
 - XAMPP versi 1.8.3
 - Sublime Text 2.0.2

3.7 Pengujian Algoritme SVR

Setelah sistem selesai dibuat, maka selanjutnya adalah melakukan proses pengujian algoritme dengan menguji masing-masing nilai *learning rate* (cLR), *epsilon* (ϵ), *Complexity* (C), *lambda* (λ), dan Sigma (σ). Serta melakukan proses iterasi α_i^* dan α_i sebanyak mungkin sampai didapatkan hasil yang paling optimal.

3.8 Analisis Hasil / Pembahasan

Setelah proses pengujian nilai *learning rate* (cLR), *epsilon* (ϵ), *Complexity* (C), *lambda* (λ), dan Sigma (σ) selesai dilakukan, maka langkah selanjutnya adalah melakukan evaluasi kesalahan dengan menghitung tingkat error, yang mana dalam penelitian ini proses evaluasi kesalahannya menggunakan *Root Mean Square Error* (RMSE).

3.9 Penarikan Kesimpulan

Penarikan kesimpulan dilakukan setelah semua proses selesai dilakukan, mulai dari perancangan sistem sampai dengan pengujian dari Implementasi Metode *Support Vector Regression* (SVR) dalam Peramalan Penjualan Roti (Studi Kasus: Harum Bakery). Kesimpulan diambil dari jawaban rumusan masalah dan juga diambil dari pengujian sistem dan analisis metode. Setelah penarikan kesimpulan, selanjutnya adalah membuat saran. Saran berisi tentang masukan dari penulis tentang hasil yang sudah dicapai dalam penelitian ini dan dapat digunakan sebagai acuan untuk penulis atau peneliti selanjutnya yang ingin mengembangkan atau memperbaiki penelitian serupa.

BAB 4 PERANCANGAN

4.1 Desain Algoritme

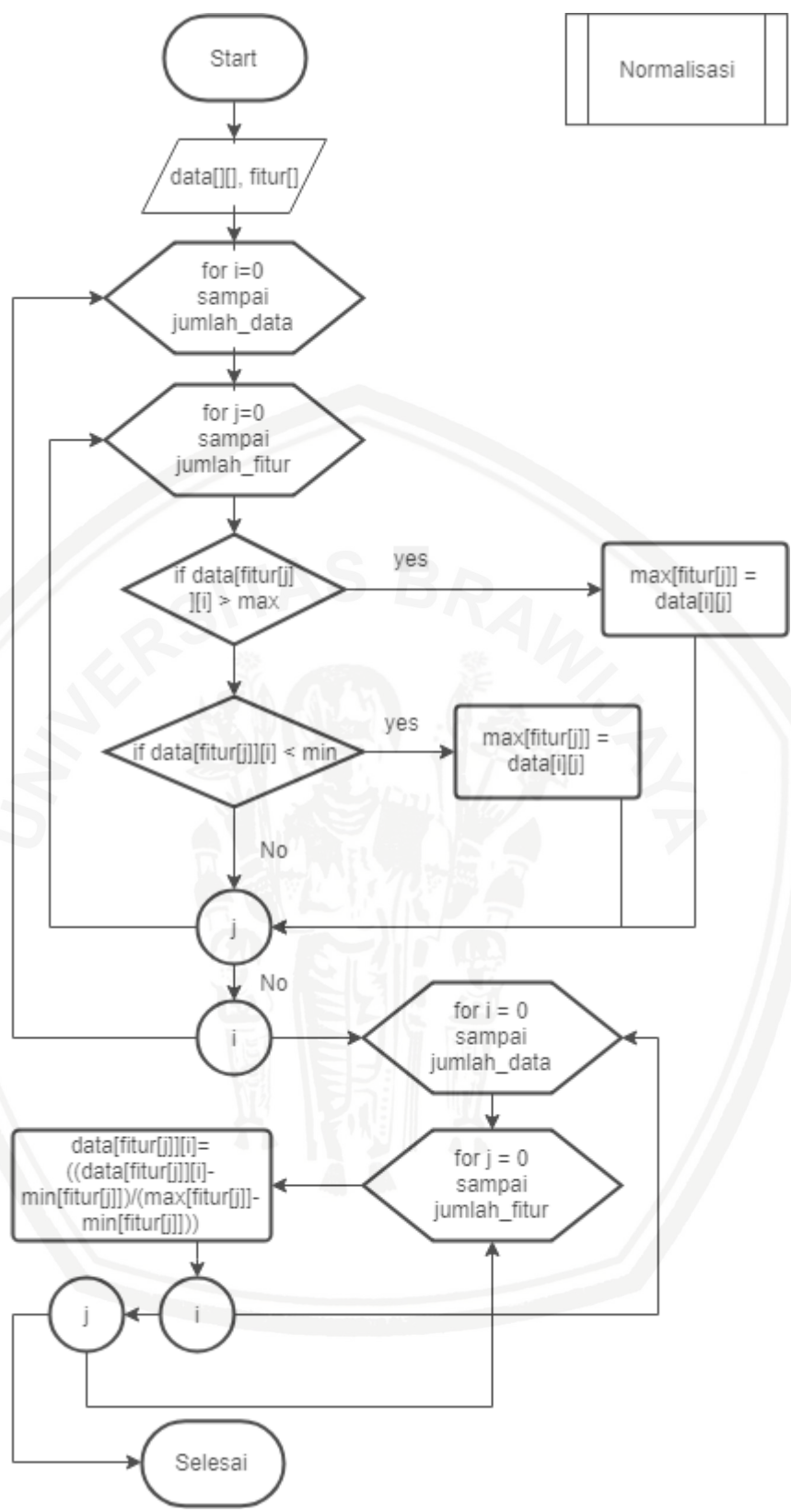
Pada permasalahan yang sudah dijabarkan dan dijelaskan, metode yang akan digunakan dalam penelitian ini adalah *Support Vector Regression*. Algoritma *Support Vector Regression* digunakan untuk memprediksi penjualan roti pada toko roti Harum Bakery.

Langkah-langkah yang akan dilakukan dalam menggunakan algoritme *Support Vector Regression* ada lima tahap, sehingga diharapkan dapat menghasilkan output yang sesuai. Langkah pertama yang dilakukan adalah menentukan data testing dan data uji yang akan digunakan dalam algoritme SVR. Langkah kedua adalah menentukan inisialisasi parameter SVR yang akan digunakan, seperti cLr , $Complexity$, $Epsilon$, $Lambda$, $Sigma$, dan $Gama$. Langkah ketiga melakukan proses perhitungan *Sequential Learning* pada algoritme SVR, proses ini dibagi menjadi beberapa proses seperti menghitung jarak data testing dan data uji, menghitung matriks R_{ij} , menghitung nilai *error*, $\delta\alpha_i^*$ dan $\delta\alpha_i$, serta α_i^* dan α_i . Setelah itu langkah keempat yaitu menghitung peramalan $f(x)$, lalu langkah kelima merupakan langkah terakhir yaitu menghitung nilai evaluasi kerja pada algoritme SVR menggunakan *Root Mean Square Error (RMSE)*.

4.2 Flowchart

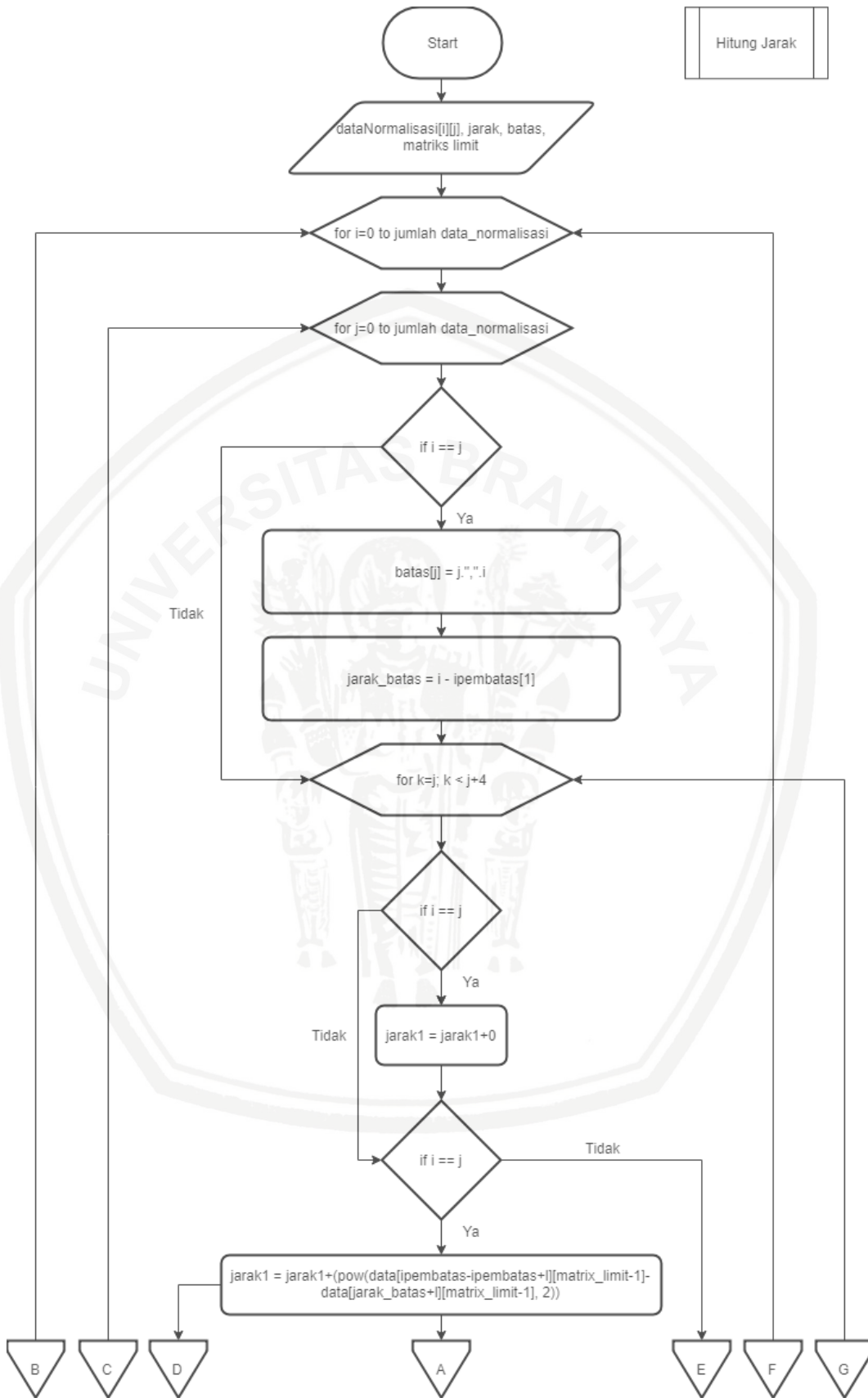
4.2.1 Normalisasi

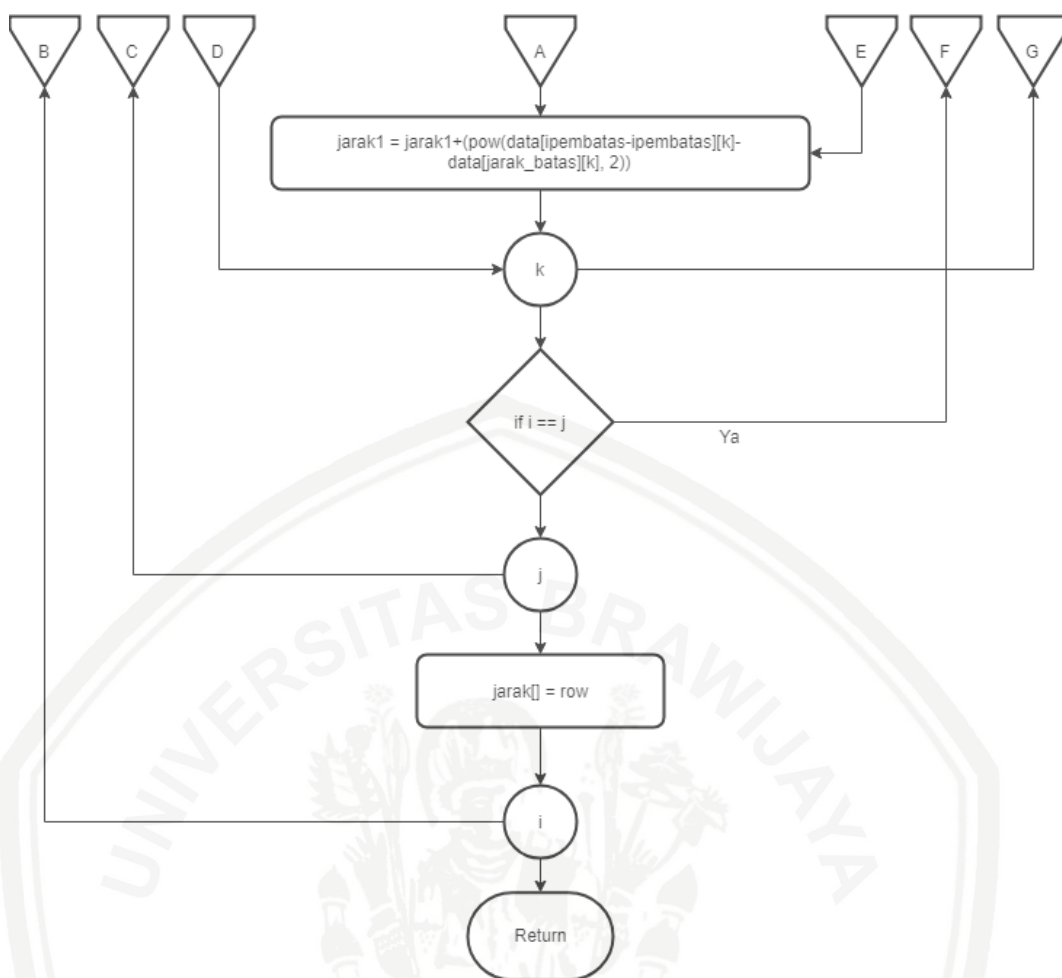
Proses pertama yang dilakukan adalah normalisasi, normalisasi dilakukan dengan tujuan untuk mendapatkan nilai dengan ukuran yang lebih kecil. Pada proses normalisasi, pertama dilakukan pembentukan matriks, i untuk baris dan j untuk kolom. Setelah itu dicari data maksimal dan minimal pada matriks sebanyak i baris dan j kolom, jika sudah didapatkan data maksimal dan minimal, maka dilakukan proses perhitungan menggunakan rumus normalisasi. Jika sudah didapatkan hasilnya, data akan dikembalikan ke *controller* untuk digunakan dalam proses selanjutnya. Diagram alir yang menjelaskan proses normalisasi ditunjukkan pada gambar 4.1.



Gambar 4.1 Diagram Alir Proses Normalisasi

4.2.2 Hitung Jarak

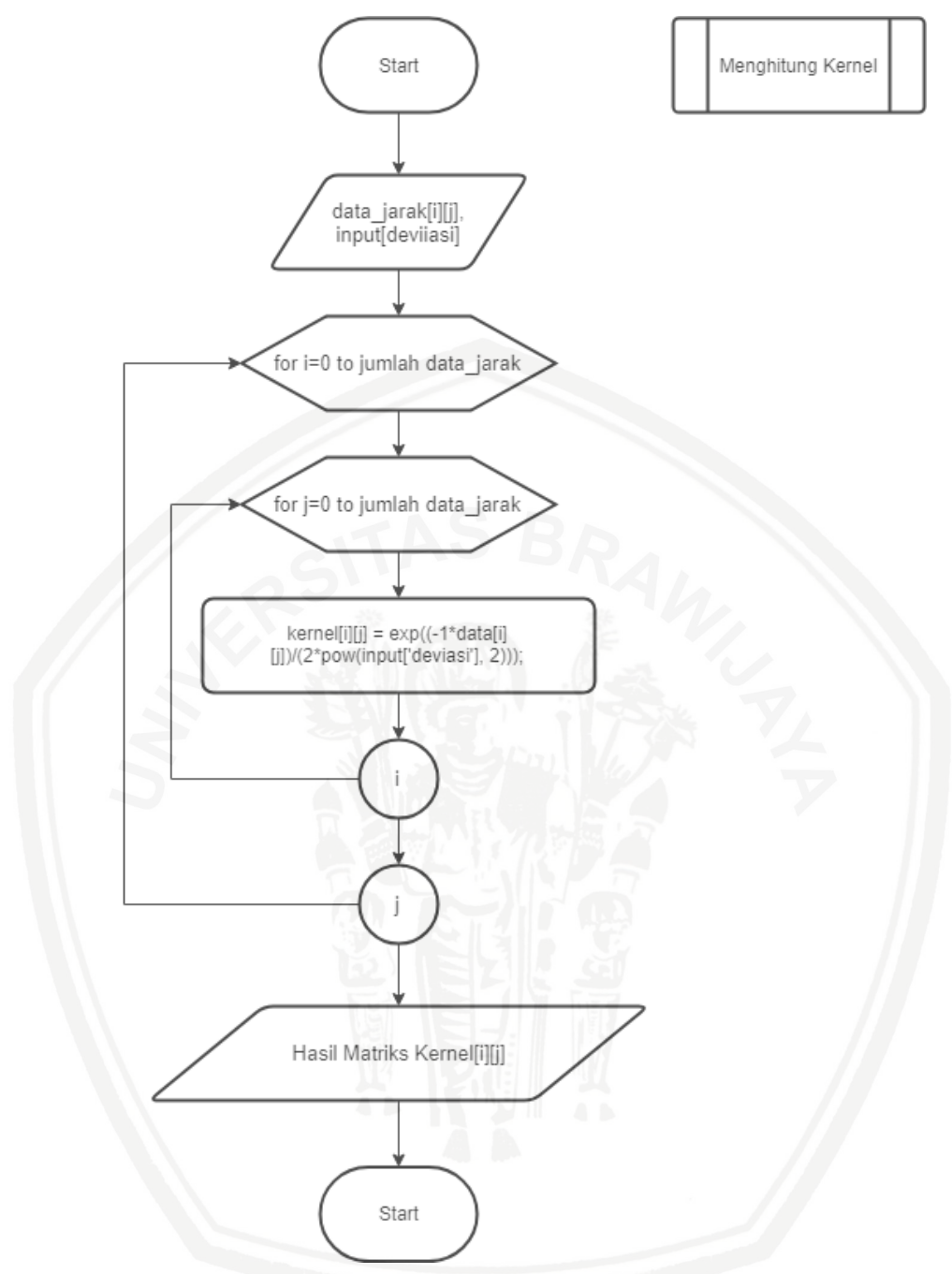




Gambar 4.2 Diagram Alir Proses Hitung Jarak

Gambar 4.2 menunjukkan diagram alir atau *flowchart* proses menghitung jarak. Untuk proses perhitungan jarak, data yang dibutuhkan adalah data normalisasi, variabel jarak, variabel batas, dan variabel matriks limit. Setelah itu dibentuk matriks dengan kolom dan baris sebanyak matriks normalisasi. Lalu dibuat batas nilai 0 secara diagonal, lalu dilakukan perhitungan menggunakan rumus perhitungan jarak. Jika semua data sudah didapatkan sebanyak *i* kolom dan *j* baris, maka data dikembalikan ke *controller* untuk kemudian digunakan dalam proses selanjutnya.

4.2.3 Hitung Kernel RBF



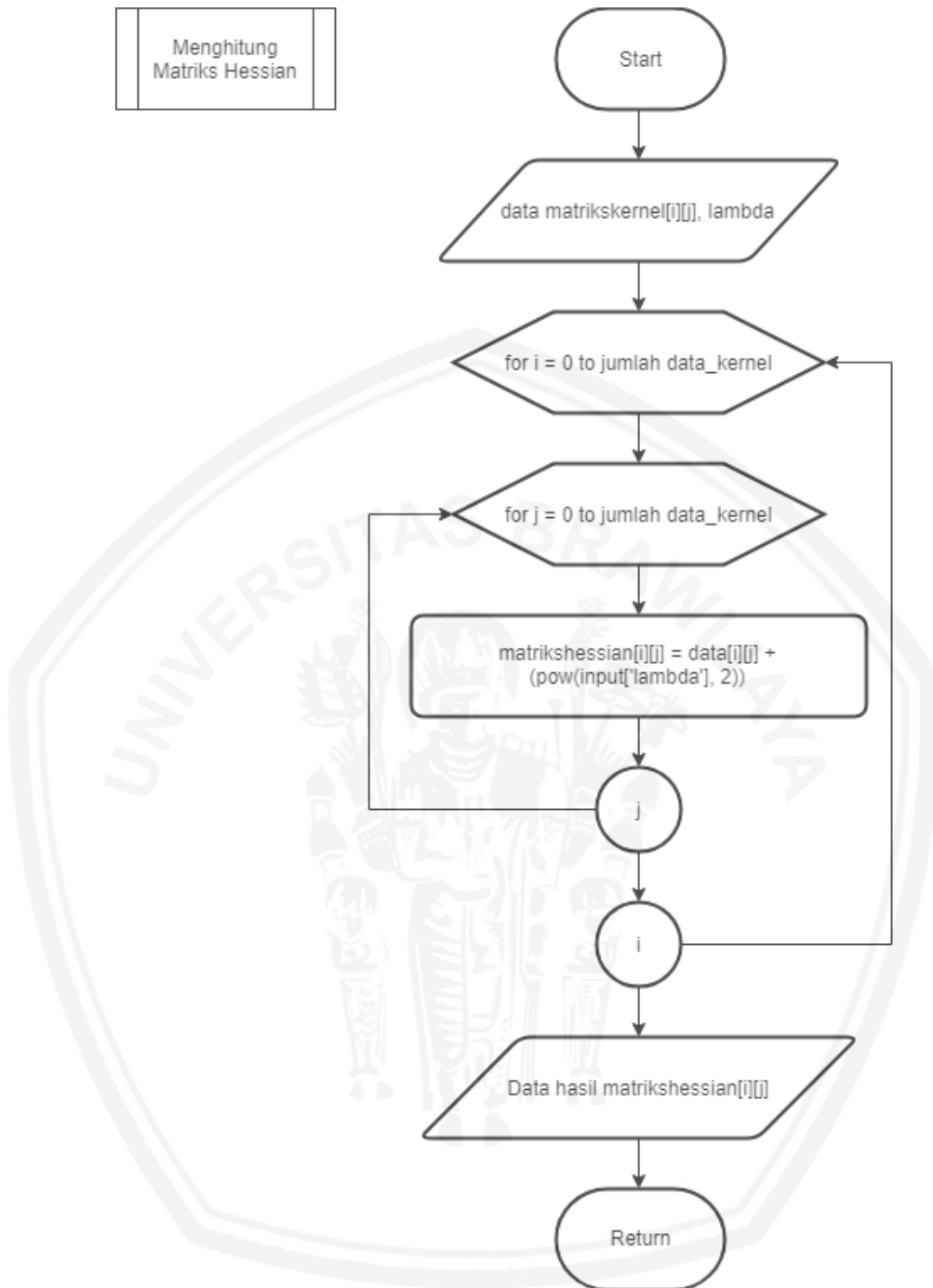
Gambar 4. 3 Diagram Alir Proses Hitung Kernel RBF

Untuk proses perhitungan Kernel ditunjukkan pada gambar 4.3, yang dibutuhkan adalah matriks data jarak dan input dari form deviasi. Lalu dibentuk matriks dengan i dan j sebanyak matriks jarak, setelah itu dimasukkan rumus hitung kernel, maka didapatkan data matriks Kernel RBF. Setelah itu data dikembalikan ke *controller* untuk digunakan dalam proses perhitungan selanjutnya.



4.2.4 Matriks Hessian

Menghitung Matriks Hessian

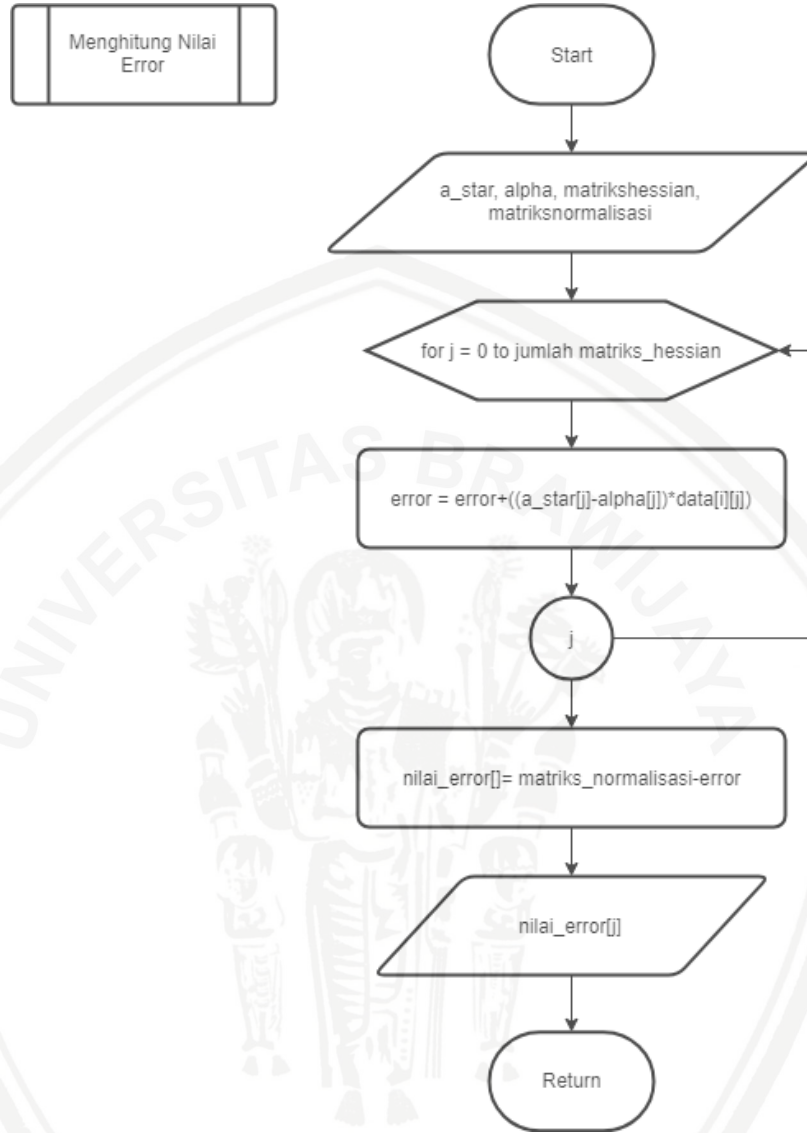


Gambar 4.4 Diagram Alir Proses Hitung Matriks Hessian

Proses selanjutnya adalah menghitung matriks Hessian. Seperti ditunjukkan pada gambar 4.4, untuk menghitung matriks Hessian dibutuhkan data matriks Kernel dan input lambda dari form yang ada di *view*. Selanjutnya dibentuk matriks dengan jumlah baris dan kolom yang sesuai dengan matriks Kernel, lalu dilakukan proses perhitungan matriks Hessian dengan rumusnya, sehingga didapatkan hasil

perhitungan matriks Hessian. Setelah matriks Hessian didapatkan, maka datanya akan dikembalikan ke *controller* untuk digunakan dalam proses selanjutnya.

4.2.5 Hitung Nilai Error

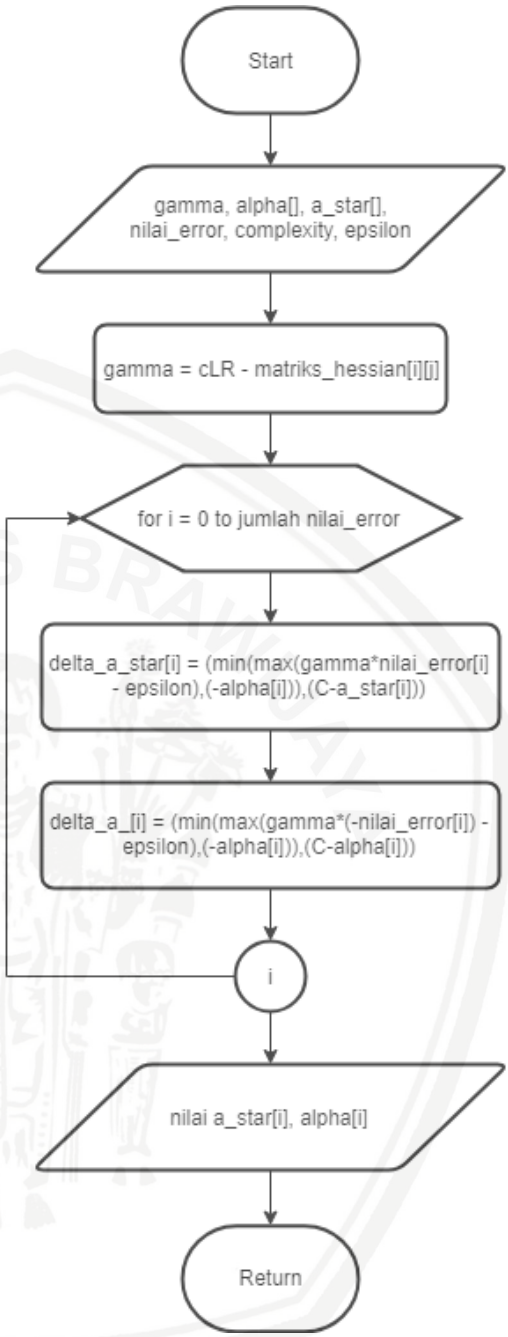


Gambar 4.5 Diagram Alir Proses Hitung Nilai Error

Jika sudah didapatkan data matriks Hessian, maka selanjutnya menghitung nilai error, seperti ditunjukkan pada gambar 4.5. Untuk menghitung nilai error dibutuhkan data *alpha star*, *alpha*, matriks Hessian, dan matriks Normalisasi. Untuk menghitung nilai error, dibutuhkan data matriks Hessian per baris yang akan diinisialisasikan dengan *j*. Setelah itu dimasukkan rumus dalam menghitung nilai error. Untuk itersi pertama digunakan nilai *alpha star* dan *alpha* yaitu 0, untuk iterasi selanjutnya akan digunakan nilai *alpha star* dan *alpha* pada iterasi sebelumnya. Setelah didapatkan nilai error, maka nilai akan dikembalikan untuk digunakan pada proses perhitungan nilai *delta alpha star* dan *delta alpha*.

4.2.6 Hitung nilai *delta alpha star* dan *delta alpha*

Menghitung nilai δa^* dan δa

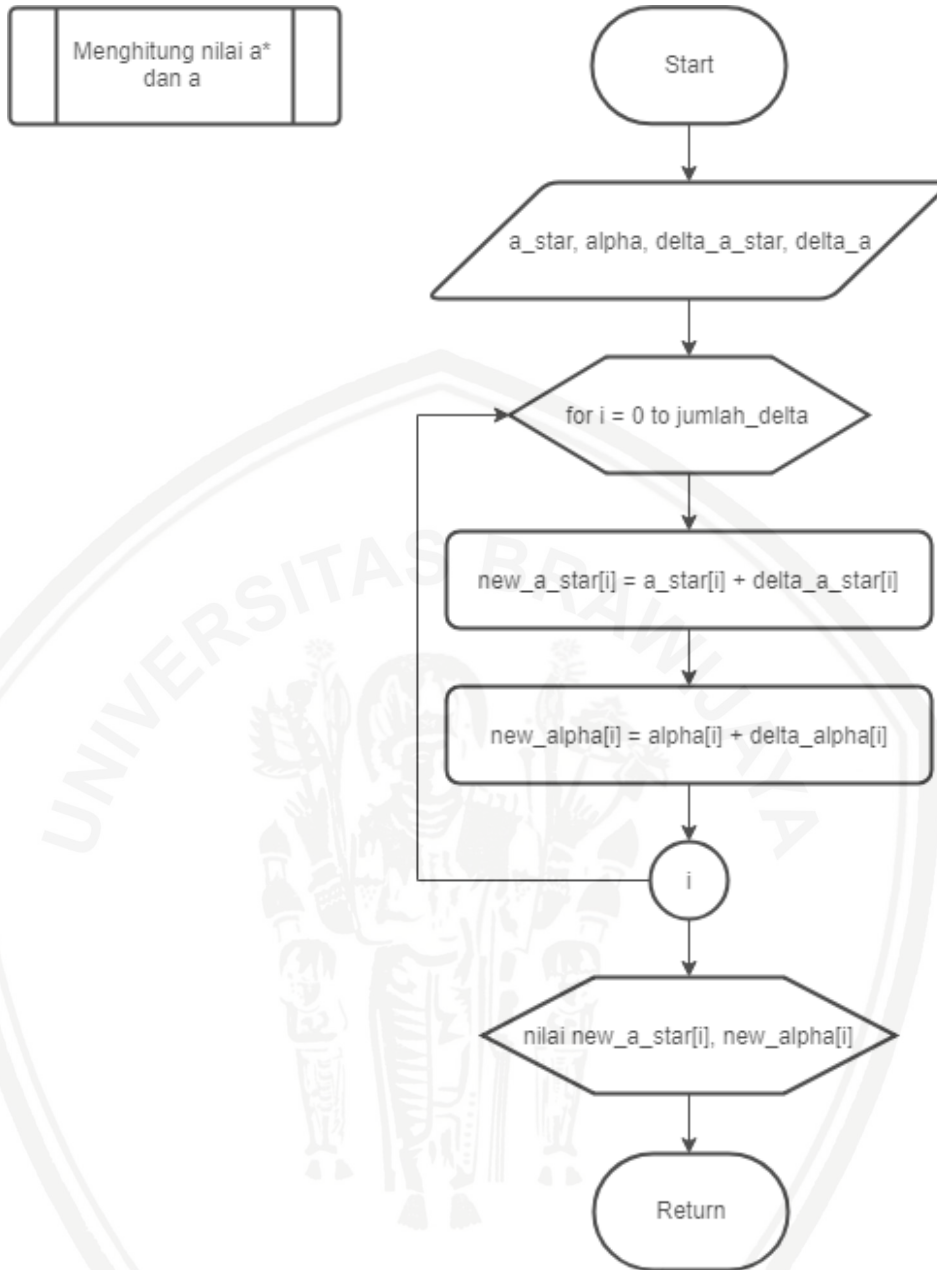


Gambar 4.6 Diagram Alir Proses Hitung *Delta Alpha* dan *Delta Alpha Star*

Setelah didapatkan nilai *error*, maka dilakukan proses perhitungan nilai *delta alpha star* dan *delta alpha* seperti ditunjukkan pada gambar 4.6. Untuk melakukan proses perhitungan ini dibutuhkan nilai *gamma*, *alpha*, *alpha star*, nilai *error*, *complexity*, dan *epsilon*. Lalu dilakukan perhitungan *delta alpha star* dan *delta alpha* dengan rumus yang berbeda. Setelah nilainya didapatkan maka akan digunakan dalam melakukan perhitungan nilai *alpha star* dan *alpha*.



4.2.7 Hitung nilai *alpha star* dan *alpha*

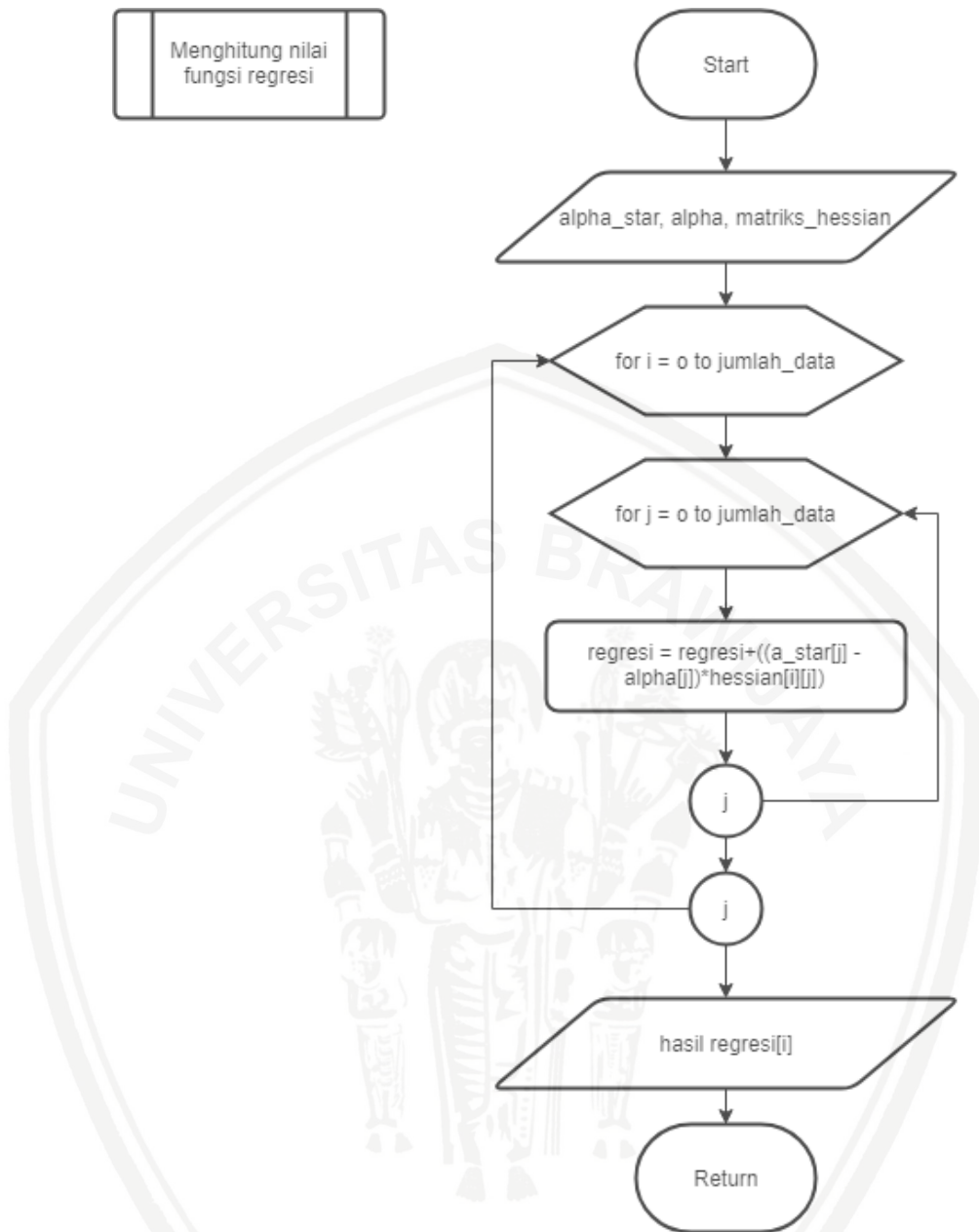


Gambar 4.7 Diagram Alir Proses Hitung Nilai *Alpha* dan *Alpha Star*

Gambar 4.7 adalah diagram alir dalam mencari nilai *alpha star* dan *alpha*. Untuk mencari nilai ini yang dibutuhkan adalah nilai *delta alpha star*, *delta alpha*, serta nilai *alpha* dan *alpha star* pada iterasi sebelumnya. Lalu dibentuk metrik sebanyak *i* baris sesuai dengan matriks *delta alpha star* dan *delta alpha*. Setelah nilainya didapatkan maka akan dikembalikan ke *controller* untuk digunakan dalam perhitungan nilai fungsi regresi $f(x)$.

4.2.8 Fungsi Regresi

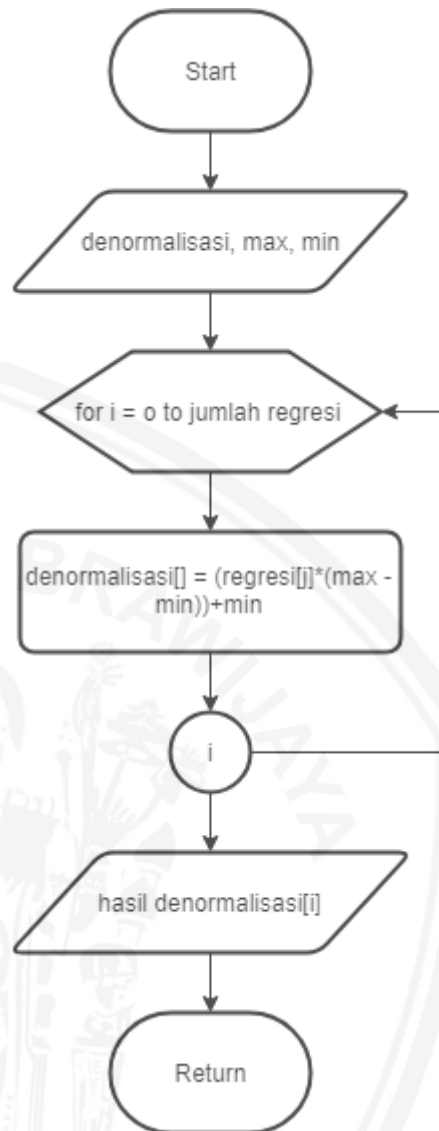
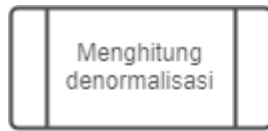
Menghitung nilai fungsi regresi



Gambar 4.8 Diagram Alir Proses Hoitung Nilai Regresi

Selanjutnya melakukan proses perhitungan nilai fungsi regresi. Untuk menghitung nilai fungsi regresi dibutuhkan nilai *alpha star*, *alpha*, dan data matriks Hessian, seperti ditunjukkan pada gambar 4.8. Setelah itu dibentuk matriks sebanyak *i* baris pada matriks *alpha star* dan *alpha*. Lalu dimasukkan rumus untuk mengitung fungsi regresi, jika nilainya sudah didapatkan maka nilainya akan di denormalisasi di proses selanjutnya.

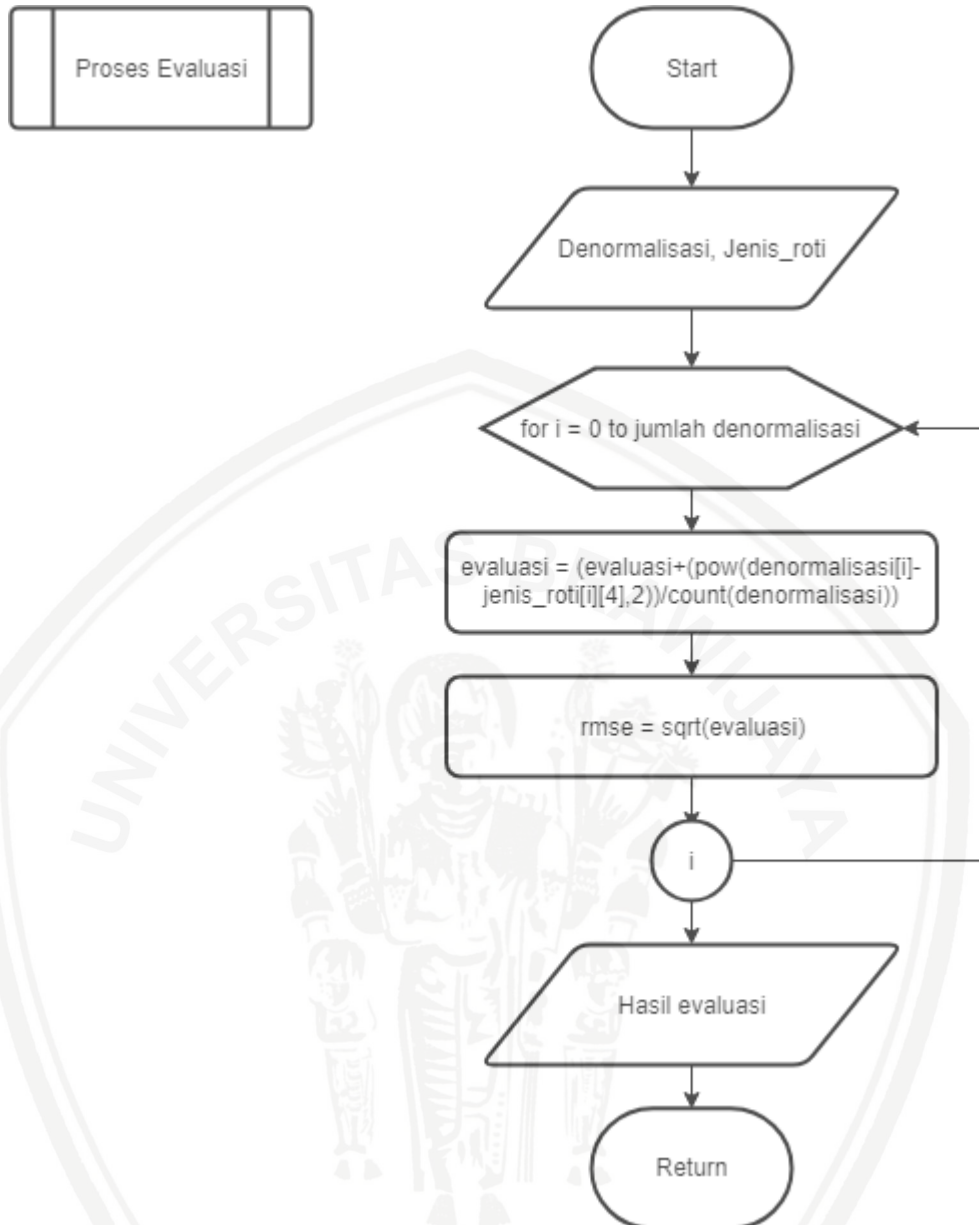
4.2.9 Denormalisasi



Gambar 4.9 Diagram Alir Proses Denormalisasi

Proses selanjutnya adalah melakukan denormalisasi, seperti ditunjukkan pada gambar 4.9. Untuk matriks yang terbentuk sebanyak i baris sesuai dengan matriks fungsi regresi. Lalu dimasukkan rumus untuk menghitung denormalisasi, jika nilainya sudah didapatkan maka selanjutnya dilakukan proses evaluasi menggunakan *Root Mean Square Error* (RMSE).

4.2.10 Evaluasi (RMSE)



Gambar 4.10 Diagram Alir Proses Hitung RMSE

Proses terakhir dari peramalan ini adalah evaluasi. Gambar 4.10 menunjukkan data yang dibutuhkan adalah data denormalisasi dan data aktual roti. Lalu untuk untuk perhitungan menggunakan semua data yang ada di matriks denormalisasi sebanyak i baris. Setelah itu dimasukkan rumus perhitungan untuk proses evaluasinya sehingga menghasilkan nilai pengukuran evaluasi dari proses peraman roti yang sudah dilakukan.

4.3 Perhitungan Manual

Peramalan penjualan roti yang akan dilakukan adalah dengan membentuk model regresi, yang akan dilanjutkan dengan pengujian sehingga mendapatkan nilai regresi dan nilai *error rate* (nilai kesalahan). Pada nilai *error rate* ini menunjukkan keakuratan dalam peramalan penjualan roti menggunakan metode SVR. Semakin besar nilai error yang diperoleh, maka hasilnya akan semakin buruk atau tidak sesuai, sebaliknya apabila nilai error yang diperoleh kecil, maka hasilnya baik dan sistem dapat dikatakan akurat. Data yang digunakan sebagai data latih dan data uji pada bab ini diambil dari data penjualan roti hari senin bulan oktober 2017 sampai dengan bulan desember 2017, yang ditunjukkan pada Tabel 4.1 berikut.

Tabel 4.1 Data Latih dan Data Uji

Tanggal	Jumlah Penjualan
9 Oktober 2017	91
16 Oktober 2017	230
23 Oktober 2017	124
30 Oktober 2017	255
6 November 2017	152
13 November 2017	261
20 November 2017	233
27 November 2017	255
4 Desember 2017	136
11 Desember 2017	157
18 Desember 2017	169
25 Desember 2017	152

4.3.1 Data Uji / Data Testing

Fitur yang digunakan dalam peramalan ini diambil dari data 4 hari sebelumnya dengan target pada hari berikutnya. Inisialisasi nilai yang digunakan adalah X_1 , X_2 , X_3 , X_4 , dan Y . X adalah data pada setiap hari sebelumnya, dan Y adalah target. Data pada Tabel 4.1 akan diubah menjadi data *testing* yang ditunjukkan seperti Tabel 4.2 berikut.

Tabel 4.2 Data *Testing* Roti Manis

No.	X1	X2	X3	X4	Y
1	91	230	124	255	152
2	230	124	255	152	261
3	124	255	152	261	233
4	255	152	261	233	255
5	152	261	233	255	136
6	261	233	255	136	157
7	233	255	136	157	169
8	255	136	157	169	152

4.3.2 Normalisasi Data

Setelah menentukan data *testing*, selanjutnya dilakukan proses normalisasi data yang bertujuan untuk mendapatkan data dengan rentang nilai yang tidak terlalu jauh, namun tidak mengubah karakteristik data itu sendiri. Untuk proses normalisasi, langkah pertama yang dilakukan adalah menentukan nilai maksimum dan minimum dari data yang bersangkutan. Data maksimum dari penjualan roti manis adalah 261, sedangkan data minimumnya 91. Hasil normalisasi data ditunjukkan pada Tabel 4.3. Berikut contoh rumus perhitungan normalisasi sesuai dengan persamaan 2.9:

$$x = \frac{230-91}{261-91}$$

$$x = 0,81765$$

Tabel 4.3 Normalisasi Data Penjualan Roti Manis

No.	X1	X2	X3	X4	Y
1	0	0.817647	0.194118	0.964706	0.358824
2	0.817647	0.194118	0.964706	0.358824	1
3	0.194118	0.964706	0.358824	1	0.835294
4	0.964706	0.358824	1	0.835294	0.964706
5	0.358824	1	0.835294	0.964706	0.264706
6	1	0.835294	0.964706	0.264706	0.388235
7	0.835294	0.964706	0.264706	0.388235	0.458824
8	0.964706	0.264706	0.388235	0.458824	0.358824

4.3.3 Inisialisasi Parameter SVR

Parameter SVR yang digunakan adalah *cLR*, *complexity (c)*, *Epsilon (ε)*, *Lambda (λ)*, *Sigma (σ)*, dan jumlah iterasi dalam sequential learning. Contoh inisialisasi parameter SVR ditunjukkan pada Tabel 4.4:

Tabel 4.4 Range Parameter SVR

Parameter	Minimal	Maksimal
<i>cLR</i>	0,006	0,015
<i>C</i>	0,0005	0,005
<i>ε</i>	0,000005	0,1
<i>λ</i>	0,001	30
<i>σ</i>	0,05	4

Tabel 4.5 Contoh nilai Parameter SVR

<i>cLR</i>	<i>C</i>	<i>ε</i>	<i>λ</i>	<i>σ</i>
0,1	100	0,001	0,5	0,5

4.3.4 Menghitung Jarak

Setelah melakukan proses normalisasi serta menentukan range parameter SVR dan nilai Parameter SVR yang digunakan dalam bab ini, selanjutnya dilakukan proses menghitung jarak dengan menggunakan data hasil normalisasi lalu dipangkatkan dua, contoh perhitungannya seperti berikut:

$$||x_i - x_j ||^2 = (0 - 0,81765)^2 + (0,81765 - 0,19412)^2 + (0,19412 - 0,96471)^2 + (0,96471 - 0,35882)^2 = \mathbf{2,01824}$$

Selanjutnya hasil perhitungan jarak ditunjukkan pada Tabel 4.6:

Tabel 4.6 Hasil Hitung Jarak

No.	1	2	3	4	5	6	7	8
1	0	2.018235	0.087682	1.80737	0.573114	2.084118	1.056644	1.53000
2	2.018235	0	1.760796	0.277024	1.243806	0.453218	1.084983	0.368927
3	0.087682	1.760796	0	1.399135	0.256644	1.573945	0.794221	1.377543
4	1.80737	0.277024	1.399135	0	0.822076	0.555087	1.12436	0.524844

No.	1	2	3	4	5	6	7	8
5	0.573114	1.243806	0.256644	0.822076	0	0.944983	0.886159	1.363529
6	2.084118	0.453218	1.573945	0.555087	0.944983	0	0.549135	0.696817
7	1.056644	1.084983	0.794221	1.12436	0.886159	0.549135	0	0.52699
8	1.53000	0.368927	1.377543	0.524844	1.363529	0.696817	0.52699	0

4.3.5 Menghitung Kernel

Dalam penelitian ini perhitungan kernel yang digunakan adalah Kernel RBF (*Radial Basis Function*), dengan rumus yang sudah ditunjukkan pada persamaan 2.8 diatas. Hasil dari perhitungan Kernel ini akan digunakan untuk menghitung Matriks Hessian. Untuk menghitung Kernel RBF ini menggunakan parameter sigma dan dengan nilai jarak yang sudah didapatkan pada Tabel 4.6. Untuk hasil perhitungan kernel ditunjukkan pada Tabel 4.7. Berikut ini adalah contoh perhitungan Kernel RBF:

$$K(x_i, x_j) = \exp\left(-\frac{1}{2(0,5)^2}\right) 2.018235$$

$$= 0,01766$$

Tabel 4.7 Hasil Hitung Kernel RBF

No.	1	2	3	4	5	6	7	8
1	1	0.01766	0.839152	0.026924	0.317833	0.01548	0.12084	0.046888
2	0.01766	1	0.029552	0.574619	0.083108	0.403961	0.114182	0.478139
3	0.839152	0.029552	1	0.060915	0.598525	0.042943	0.204243	0.063604
4	0.026924	0.574619	0.060915	1	0.193176	0.329502	0.105534	0.350047
5	0.317833	0.083108	0.598525	0.193176	1	0.151077	0.169939	0.065411
6	0.01548	0.403961	0.042943	0.329502	0.151077	1	0.333447	0.248172
7	0.12084	0.114182	0.204243	0.105534	0.169939	0.333447	1	0.348548
8	0.046888	0.478139	0.063604	0.350047	0.065411	0.248172	0.348548	1

4.3.6 Menghitung Matriks Hessian

Proses selanjutnya setelah menghitung Kernel RBF adalah menghitung Matriks Hessian, rumus dalam menghitung Matriks Hessian adalah nilai hasil hitung kernel ditambahkan dengan lambda yang dipangkatkan dengan dua, seperti ditunjukkan pada persamaan 2.2. Contoh perhitungannya seperti berikut:

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



$$= 1 + 0,5^2$$

$$= 1,25$$

Hasil perhitungan matriks hessian ditunjukkan pada Tabel 4.8:

Tabel 4.8 Hasil Hitung Matriks Hessian

No.	1	2	3	4	5	6	7	8
1	1.25	0.26766	1.089152	0.276924	0.567833	0.26548	0.37084	0.296888
2	0.26766	1.25	0.279552	0.824619	0.333108	0.653961	0.364182	0.728139
3	1.089152	0.279552	1.25	0.310915	0.848525	0.292943	0.454243	0.313604
4	0.276924	0.824619	0.310915	1.25	0.443176	0.579502	0.355534	0.600047
5	0.567833	0.333108	0.848525	0.443176	1.25	0.401077	0.419939	0.315411
6	0.26548	0.653961	0.292943	0.579502	0.401077	1.25	0.583447	0.498172
7	0.37084	0.364182	0.454243	0.355534	0.419939	0.583447	1.25	0.598548
8	0.296888	0.728139	0.313604	0.600047	0.315411	0.498172	0.598548	1.25

4.3.7 Menghitung Nilai α_i^* dan α_i

Langkah selanjutnya setelah mendapatkan nilai Matriks Hessian, dilakukan proses mencari nilai α_i^* (*alpha star*) dan α_i (*alpha*). Dalam mencari nilai *alpha star* dan *alpha* diperlukan 3 tahap. Yang pertama mencari nilai *error*, lalu selanjutnya mencari nilai $\delta\alpha_i^*$ (*delta alpha star*) dan $\delta\alpha_i$ (*delta alpha*), dan yang terakhir mencari nilai *alpha star* dan *alpha*. Setelah mendapat nilai *alpha star* dan *alpha* dilakukan proses iterasi hingga mendapatkan nilai *alpha star* dan *alpha* yang optimal.

1. Menghitung Nilai *Error*

Dalam proses menghitung nilai *error*, menggunakan rumus yang ditunjukkan pada persamaan 2.3 diatas, yaitu nilai normalisasi dikurangi oleh jumlah keseluruhan *alpha star* yang dikurangi *alpha* pada iterasi sebelumnya, lalu dikali dengan matriks hessian. Untuk inialisasi nilai awal α_i^* dan α_i adalah 0.

Tabel 4.9 Nilai α_i^* dan α_i pada iterasi pertama

No.	α_i^*	α_i
1	0.02863	0



No.	α_i^*	α_i
2	0.07992	0
3	0.06674	0
4	0.0771	0
5	0.0211	0
6	0.03098	0
7	0.03663	0
8	0,02863	0

Berikut ini adalah contoh perhitungan mencari nilai error:

$$E_i = 0,35882 - (((0,02863 - 0) \times 1,25) + ((0,07992 - 0) \times 0,26766) + \dots + ((0,02863 - 0) \times 0,29689) = \mathbf{0,16532}$$

Hasil hitung nilai *error* ditunjukkan pada Tabel 4.10:

Tabel 4.10 Hasil Hitung Nilai Error

No.	Nilai Error
1	0.165322
2	0.748736
3	0.621784
4	0.716253
5	0.067823
6	0.181327
7	0.271524
8	0.145148

2. Menghitung nilai $\delta\alpha_i^*$ dan $\delta\alpha_i$

Setelah mendapatkan nilai error, selanjutnya dilakukan perhitungan nilai $\delta\alpha_i^*$ dan $\delta\alpha_i$ pada iterasi yang sama. Untuk menghitung nilai $\delta\alpha_i^*$ dan $\delta\alpha_i$ dibutuhkan nilai error pada iterasi yang sama, alpha *star*, dan alpha pada iterasi sebelumnya. Serta dibutuhkan parameter *learning rate* (γ), *complexity* (C), dan *epsilon* (ϵ). Untuk rumus menghitung nilai $\delta\alpha_i^*$ dan $\delta\alpha_i$ dapat dilihat pada persamaan 2.4 diatas. Sedangkan untuk nilai γ didapatkan dari nilai *coefisien learning rate* (cLR) dibagi



dengan nilai maksimal pada matriks hessian, seperti ditunjukkan pada persamaan 2.6 diatas. Berikut ini adalah contoh untuk menghitung nilai $\delta\alpha_i^*$:

$$\delta\alpha_i^* = \min\{\max[0,08 \times (0,16532 - 0,001), -0,02863], 100 - 0,02863\}$$

$$= \mathbf{0,01315}$$

Lalu untuk menghitung $\delta\alpha_i$ contoh perhitungannya adalah sebagai berikut:

$$\delta\alpha_i = \min\{\max[0,08 \times (-0,16532 - 0,001), -0], 100 - 0\}$$

$$= \mathbf{0}$$

Selanjutnya hasil perhitungan dari $\delta\alpha_i^*$ dan $\delta\alpha_i$ ditunjukkan pada Tabel 4.11 berikut.

Tabel 4.11 Hasil hitung nilai $\delta\alpha_i^*$ dan $\delta\alpha_i$

No.	$\delta\alpha_i^*$	$\delta\alpha_i$
1	0.013146	0
2	0.059819	0
3	0.049663	0
4	0.05722	0
5	0.005346	0
6	0.014426	0
7	0.021642	0
8	0.011532	0

3. Menghitung nilai α_i^* dan α_i

Setelah mendapatkan nilai $\delta\alpha_i^*$ dan $\delta\alpha_i$, selanjutnya mencari nilai α_i^* dan α_i pada iterasi yang sama. Rumus untuk mendapatkan nilai α_i^* dan α_i dapat dilihat dari persamaan 2.5 diatas. Dalam mencari nilai α_i^* dan α_i diperlukan nilai dari $\delta\alpha_i^*$ dan $\delta\alpha_i$ pada iterasi yang sama dan nilai α_i^* dan α_i pada iterasi sebelumnya. Berikut ini adalah contoh perhitungan α_i^* :

$$\alpha_i^* = 0,02863 + 0,01315$$

$$= \mathbf{0,04177}$$

Berikut ini adalah contoh perhitungan α_i :

$$\alpha_i = 0 + 0$$

$$= \mathbf{0}$$

Selanjutnya hasil dari perhitungan α_i^* dan α_i akan ditunjukkan pada Tabel 4.11

Tabel 4.12 Hasil hitung nilai α_i^* dan α_i

No.	α_i^*	α_i
1	0.041772	0
2	0.139739	0
3	0.116406	0
4	0.134317	0
5	0.026442	0
6	0.045405	0
7	0.058268	0
8	0.040158	0

4.3.8 Menentukan Fungsi Regresi $f(x)$

Langkah selanjutnya setelah mendapatkan nilai α_i^* dan α_i adalah menghitung nilai fungsi regresi $f(x)$. pada perhitungan nilai fungsi regresi $f(x)$ digunakan nilai α_i^* dan α_i pada iterasi terakhir dan nilai matriks Hessian. Dalam bab ini nilai α_i^* dan α_i iterasi terakhir adalah iterasi ke 2, dan ditunjukkan pada Tabel 4.12, serta nilai matriks Hessian yang ditunjukkan pada Tabel 4.8.

Tabel 4.13 Nilai α_i^* dan α_i iterasi ke 2

No.	α_i^*	α_i
1	0.041772	0
2	0.139739	0
3	0.116406	0
4	0.134317	0
5	0.026442	0
6	0.045405	0
7	0.058268	0
8	0.040158	0

Setelah proses *sequential learning* selesai dilakukan, maka didapatkan nilai α_i^* dan α_i baru dan bisa dilakukan proses perhitungan fungsi regresi $f(x)$. Fungsi regresi

diperoleh dari jumlah keseluruhan dari *alpha star* (α_i^*) dikurangi alpha (α_i) pada iterasi terakhir yang kemudian dikali dengan matriks Hessian. Untuk rumus mencari nilai fungsi regresi $f(x)$ digunakan rumus seperti pada persamaan 2.7, dan contoh perhitungannya seperti berikut:

$$\begin{aligned}
 F(x) &= ((0,041772 - 0) \times 1,25) + ((0,139739 - 0) \times 0,26766) + \dots ((0,040158 - 0) \times \\
 &\quad 0,29689) \\
 &= \mathbf{0.314195915}
 \end{aligned}$$

Hasil perhitungan fungsi regresi ditunjukkan pada Tabel 4.13:

Tabel 4.14 Hasil Hitung Fungsi Regresi $f(x)$

No.	Fungsi Regresi $f(x)$
1	0.314195915
2	0.41811759
3	0.346628333
4	0.41373081
5	0.316965963
6	0.335773771
7	0.301478531
8	0.347285388

4.3.9 Proses Denormalisasi

Setelah mendapatkan nilai fungsi regresi, proses selanjutnya adalah melakukan denormalisasi untuk mendapatkan nilai aktual karena pada langkah awal sudah dilakukan proses normalisasi. Untuk melakukan proses denormalisasi diperlukan nilai maksimal dan minimal dari data sebenarnya. Perhitungan denormalisasi didapat dari nilai regresi yang dikalikan dengan nilai maksimal dan minimal, dan selanjutnya akan ditambah dengan nilai minimal. Untuk rumus proses denormalisasi ditunjukkan pada persamaan 2.9 dan contoh perhitungannya sebagai berikut:

$$\begin{aligned}
 y &= 0,314195915 \times (261 - 91) + 91 \\
 &= \mathbf{144.4133056}
 \end{aligned}$$

Untuk hasil perhitungan denormalisasi yang sudah dibulatkan akan ditunjukkan pada Tabel 4.13, dan data aktualnya diambil dari Tabel 4.1.

Tabel 4.15 Hasil Perhitungan Denormalisasi

No.	Denormalisasi	Data aktual
1	144	152
2	162	261
3	150	233
4	161	255
5	145	136
6	148	157
7	142	169
8	150	152

4.3.10 Proses Evaluasi (RMSE)

Setelah mendapatkan nilai semula dengan proses denormalisasi. Maka proses selanjutnya adalah melakukan proses evaluasi menggunakan *Root Mean Square Error (RMSE)*, dengan cara akar kuadrat dari semua jumlah data prediksi yang sudah di denormalisasi dikurangi oleh data aktual kemudian dibagi dengan jumlah banyaknya data. Untuk rumus menghitung RMSE dapat dilihat pada persamaan 2.11 diatas. Untuk contoh perhitungannya seperti berikut:

$$\sqrt{\frac{(144 - 152)^2 + (162 - 261)^2 + \dots + (150 - 152)^2}{8}}$$

$$= 57.4401437444684$$

4.4 Desain Antarmuka

Pada bab ini menjelaskan tentang desain antarmuka sistem yang akan dibangun. Sistem ini memiliki 2 halaman antarmuka, yang pertama adalah halaman *input*, yang berisikan 8 form pengisian, yaitu Tanggal, Jenis Roti, *Coefisien Learning Rate (cLR)*, *Complexity*, *Epsilon*, *Lambda*, *Sigma*, dan Jumlah iterasi. Lalu di halaman kedua adalah halaman *output*, yang berisikan hasil evaluasi menggunakan *Root Mean Square Error (RMSE)*.

Halaman *input*

PREDIKSI

SUBMIT

Gambar 4.11 Desain Antarmuka Halaman *Input*

Halaman *output*

HASIL RMSE:

Gambar 4.12 Desain Antarmuka Halaman *Output*

BAB 5 IMPLEMENTASI

Pada bab ini akan dijelaskan tentang implentasi sistem dan antarmuka pada prediksi penjualan roti menggunakan metode *Support Vector Regression (SVR)* berdasarkan perancangan yang sudah dilakukan pada bab 4

5.1 Implementasi Sistem

Pada sub bab ini akan dijelaskan mengenai implementasi sistem yang berupa kode program beserta penjelasannya. Kode program yang akan dijelaskan hanya mengambil beberapa sampel langkah perhitungan saja. Sistem yang dibangun menggunakan bahasa pemrograman PHP. Berikut beberapa kode program yang akan dijelaskan.

5.1.1 Implementasi Proses Normaliasi

```
1 class M_normalisasi extends CI_Model {
2
3     function normalisasi($data) {
4         $max = $this->max($data);
5         $min = $this->min($data);
6
7         $normalisasi = array();
8         foreach ($data as $key) {
9             $scnd_array = array();
10            foreach ($key as $row) {
11                $scnd_array[] = ($row-$min)/($max-$min);
12            }
13            $normalisasi[] = $scnd_array;
14        }
15
16        $return['normalisasi']=$normalisasi;
17        $return['max']=$max;
18        $return['min']=$min;
19        return $return;
20    }
21
22    function max($data)
23    {
24        $max = 0;
25        foreach ($data as $key) {
```

```
26         foreach ($key as $row) {
27             if ($row>$max) {
28                 $max = $row;
29             }
30         }
31     }
32     return $max;
33 }
34
35 function min($data)
36 {
37     $min = 10000;
38     foreach ($data as $key) {
39         foreach ($key as $row) {
40             if ($row<$min) {
41                 $min = $row;
42             }
43         }
44     }
45     return $min;
46 }
```

Penjelasan:

1. Baris 4 dan 5 menginisialisasi data maksimal dan minimal yang mengambil dari data penjualan, lalu dijadikan bentuk variabel \$max dan \$min
2. Baris 7 sampai 14 memasukkan rumus dalam menghitung nilai normalisasi, membentuk matriks sesuai dengan matriks data uji dan data latih. \$row untuk satu kolom yang bersangkutan, \$min untuk nilai minimal dan \$max untuk nilai maksimal.
3. Baris 16 sampai 19 menunjukkan nilai normalisasi, nilai maksimal, dan nilai maksimal dikembalikan ke *controller* untuk digunakan dalam proses selanjutnya.
4. Baris 22 sampai 31 memasukkan rumus dalam mencari nilai maksimal dalam matriks data uji dan data latih.
5. Baris 35 sampai 46 memasukkan rumus dalam mencari nilai minimal dalam matriks data uji dan data latih.

5.1.2 Implementasi Proses Menghitung Kernel RBF

```

1 class M_hitungkernel extends CI_Model {
2
3     function hitung($data, $input) {
4         $kernel = array();
5         for ($i=0; $i < count($data); $i++) {
6             for ($j=0; $j < count($data) ; $j++) {
7                 $kernel[$i][$j] = exp((-
8 1*$data[$i][$j])/(2*pow($input['deviasi'], 2)));
9             }
10        }
11        return $kernel;
12    }
13 }

```

Penjelasan:

1. Baris 3 menunjukkan variable yang digunakan dalam menghitung nilai kernel, yaitu \$data untuk data yang sudah dihitung jarak, dan \$input untuk input pada form input sigma yang ada pada *view*.
2. Baris ke 4 menunjukkan bahwa variabel \$kernel akan dibuat sebuah *array*.
3. Baris ke 5 menunjukkan matriks yang dihitung adalah sejumlah i baris sebanyak data pada matriks jarak.
4. Baris ke 6 menunjukkan matriks yang dihitung adalah sejumlah j kolom sebanyak data pada matriks jarak.
5. Baris ke 7 menunjukkan rumus dalam menghitung nilai kernel RBF, 'deviasi' adalah input yang ada pada form di *view*.
6. Baris 11 menunjukkan hasil nilai hitung kernel yang dikembalikan ke *controller* untuk digunakan dalam proses selanjutnya.

5.1.3 Implementasi Proses Menghitung Matriks Hessien

```

1 class M_matrikshessian extends CI_Model {
2
3     function hessian($data, $input){
4
5         $matrikshessian = array();
6         for ($i=0; $i < count($data); $i++) {
7             for ($j=0; $j < count($data) ; $j++) {
8
9

```

```

10         $matrikshessian[$i][$j] = $data[$i][$j]
11         + (pow($input['lambda'], 2));
12
13     }
14 }
15     return $matrikshessian;
16 }
17 }

```

Penjelasan:

1. Baris 3 menunjukkan variabel yang dibutuhkan pada proses perhitungan matriks hessian yaitu `$data` untuk mengambil data pada proses sebelumnya yaitu data matriks kernel, dan `$input` untuk mengambil inputan sesuai dengan id pada form di antarmuka sistem.
2. Baris 6 dan 7 menunjukkan jumlah baris dan kolom yang diambil sehingga jumlah matriks sama dengan matriks hessian.
3. Baris 9 dan 10 menunjukkan rumus dalam menghitung proses matriks hessian. 'lambda' merupakan id untuk form pengisian lambda di halaman *input*.
4. Baris 14 perintah untuk mengembalikan hasil perhitungan matriks hessian ke *Controller* sehingga datanya bisa digunakan untuk menghitung proses selanjutnya.

5.1.4 Implementasi Proses Menghitung Nilai Regresi

```

1  class M_fungsiregresi extends CI_Model {
2
3      function regresi($hessian, $alpha, $a_star) {
4
5          $fungsiregersi = array();
6          for ($i=0; $i < count($hessian); $i++) {
7              $jumlah = 0;
8              for ($j=0; $j < count($hessian) ; $j++) {
9                  $jumlah = $jumlah+(($a_star[$j] -
10                     $alpha[$j])*$hessian[$i][$j]);
11              }
12              $fungsiregersi[]= $jumlah;
13          }
14          return $fungsiregersi;
15      }

```

16	}
	}

Penjelasan:

1. Baris 3 menunjukkan variabel yang dibutuhkan pada proses perhitungan fungsi regresi, yaitu \$hessian untuk data matriks hessian, \$alpha dan \$a_star untuk nilai alpha dan alpha star pada iterasi terakhir.
2. Baris 6 sampai 8 menunjukkan data yang digunakan sejumlah baris dan kolom sesuai dengan matriks hessian.
3. Baris 9 dan 10 menunjukkan rumus yang digunakan dalam proses perhitungan fungsi regresi. Pada awal rumus ditambahkan \$jumlah agar semua data dijumlahkan, bukan hanya satu baris saja.
4. Baris 14 perintah untuk mengembalikan hasil perhitungan fungsi regresi ke Controller sehingga datanya bisa digunakan untuk menghitung proses selanjutnya.

5.1.5 Implementasi Proses Denormalisasi

1	class M_denormalisasi extends CI_Model {
2	
3	function denormalisasi(\$regresi,\$denormalisasi, \$max,
4	\$min) {
5	
6	\$denormalisasi = array();
7	for (\$j=0; \$j < count(\$regresi) ; \$j++) {
8	\$denormalisasi[] = (\$regresi[\$j]*(\$max -
9	\$min))+\$min;
10	}
11	return \$denormalisasi;
12	}
13	}

Penjelasan:

1. Baris 3 dan 4 menunjukkan variabel yang dibutuhkan pada proses denormalisasi. \$regresi untuk mengambil nilai fungsi regresi yang sudah dihasilkan, \$max dan \$min merupakan nilai maksimal dan minimal pada matriks normalisasi.
2. Baris 7 menunjukkan jumlah baris yang sesuai dengan matriks fungsi regresi.
3. Baris 8-9 menunjukkan rumus menghitung denormalisasi.

- Baris 11 mengembalikan nilai denormalisasi yang kemudian digunakan untuk menghitung proses evaluasi menggunakan *Root Mean Square Error* (RMSE).

5.2 Implementasi Antarmuka

Pada sub bab ini akan dijelaskan mengenai implementasi antarmuka sistem, yang terdiri dari 2 halaman yaitu halaman *input* dan *output*. Pada halaman *input* terdapat 8 form pengisian untuk melakukan prediksi, sedangkan halaman *output* menampilkan nilai *lagrange multiplier* (α_i^* dan α_i), fungsi regresi, denormalisasi, serta hasil evaluasi menggunakan *Root Mean Square Error* (RMSE).

5.2.1 Antarmuka Halaman *Input*

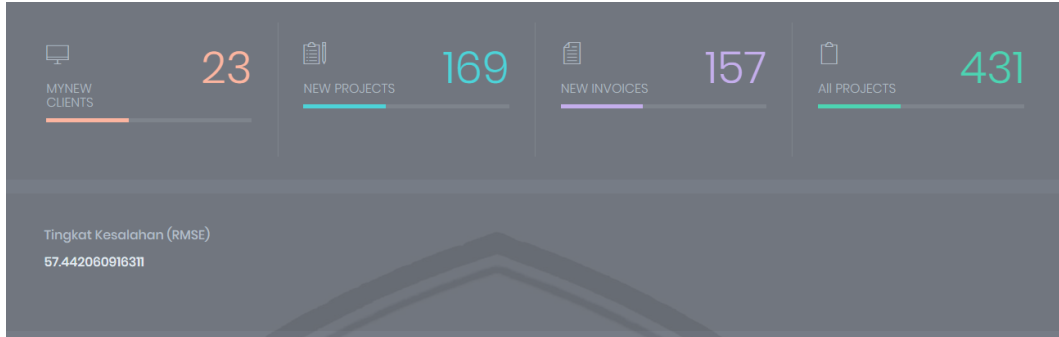
Gambar 5.1 Antarmuka Halaman *Input*

Pada halaman input terdapat 8 form pengisian dan 1 tombol. 8 form pengisian tersebut adalah:

- Tanggal. Form tanggal digunakan untuk memasukkan tanggal sekaligus hari yang akan dilakukan proses prediksi.
- Jenis Roti. Pada form ini berisikan menu dropdown untuk memilih jenis roti yang akan dilakukan proses prediksi. Terdapat 3 pilihan pada form ini yaitu manis, cake, dan tawar.
- Coefisien Learning Rate (cLR)*. Digunakan untuk memasukkan nilai *cLR*.
- Complexity*. Digunakan untuk memasukkan nilai *complexity*.
- Epsilon*. Digunakan untuk memasukkan nilai *epsilon*.
- Lambda*. Digunakan untuk memasukkan nilai *lambda*.
- Sigma. Digunakan untuk memasukkan nilai sigma.
- Jumlah iterasi. Digunakan untuk memasukkan jumlah iterasi dalam melakukan perhitungan *lagrange multiplier* (α_i^* dan α_i).

Serta terdapat satu tombol 'HITUNG' yang berfungsi untuk melakukan proses prediksi.

5.2.2 Antarmuka Halaman *Output*



Gambar 5.2 Antarmuka Halaman *Output*

Pada halaman output akan ditampilkan hasil evaluasi tingkat kesalahan menggunakan *Root Mean Square Error (RMSE)*.



BAB 6 HASIL DAN PEMBAHASAN

Pada bab hasil dan pembahasan ini dilakukan pembahasan hasil dan pengujian dalam melakukan prediksi menggunakan sistem dengan metode *Support Vector Regression (SVR)*. Pengujian yang akan dilakukan meliputi nilai parameter *Lambda* (λ), *Sigma* (σ), *Epsilon* (ϵ), *Coefisien Learning Rate* (*cLR*), *Complexity* (*c*) serta jumlah iterasi. Pada bab pengujian ini akan diambil satu sampel hari yaitu peramalan pada hari Senin tanggal 29 Januari 2018, sedangkan jenis roti yang diuji ada 3, yaitu roti tawar, cake, dan manis.

6.1 Hasil Pengujian Roti Manis

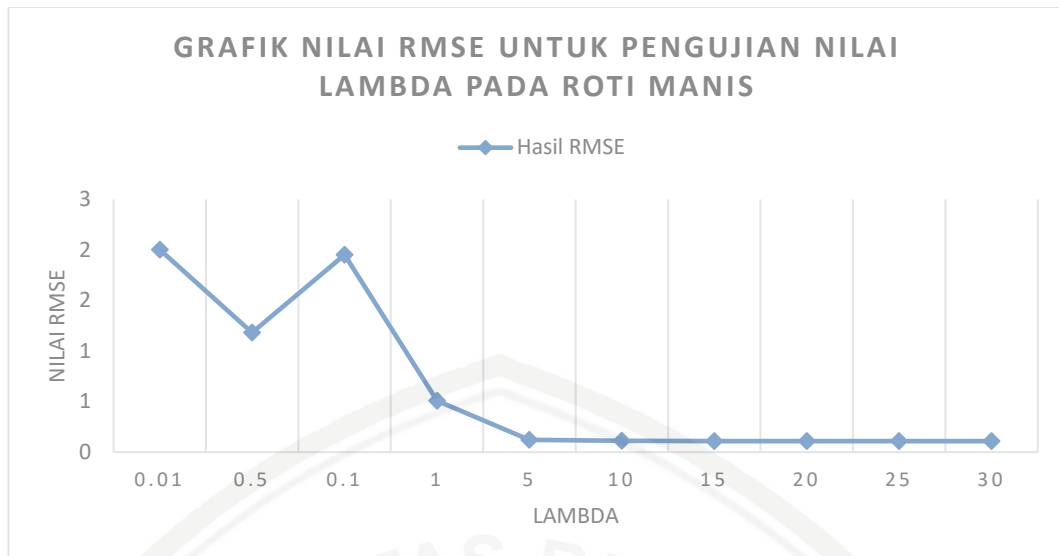
6.1.1 Hasil Pengujian Parameter Nilai *Lambda* (λ)

Untuk pengujian pertama yang dilakukan adalah pengujian parameter nilai *lambda* (λ) terbaik sehingga bisa menghasilkan solusi atau peramalan yang baik dalam kasus permasalahan ini. Ada 10 nilai *lambda* yang akan diuji pada bab ini, dan nilai parameter lain akan ditentukan dahulu. Berikut nilai parameter lain yang akan digunakan untuk mendapatkan nilai *lambda* terbaik:

- Coefisien Learning Rate* (*cLR*) = 0,01
- Complexity* = 5
- Epsilon* = 0,00001
- Sigma* = 0,5
- Jumlah iterasi = 10

Tabel 6.1 Hasil Pengujian Nilai Parameter *Lambda* (λ) untuk Roti Manis

No.	Nilai <i>Lambda</i>	Hasil RMSE
1	0,001	2,0035
2	0,5	1,1818
3	0,1	1,9534
4	1	0,5091
5	5	0,1226
6	10	0,1117
7	15	0,1097
8	20	0,1090
9	25	0,1087
10	30	0,1086



Gambar 6.1 Grafik Nilai RMSE untuk Pengujian Nilai Lambda pada Roti Manis

Parameter *lambda* berfungsi untuk menunjukkan ukuran skalar untuk pemetaan ruang pada kernel SVR (Vijayakumar, & Wu, 1999). Berdasarkan hasil pengujian parameter *lambda* yang dilakukan, Roti Manis memiliki hasil RMSE paling optimal pada parameter *lambda* 5. Seperti ditunjukkan pada gambar grafik 6.1 terlihat nilai RMSE terhadap nilai *lambda* cenderung menurun, tetapi terjadi peningkatan nilai RMSE pada nilai *lambda* 0,1 yang kemudian nilai RMSE kembali menurun dan memberikan penurunan yang signifikan sampai nilai *lambda* 5, karena untuk nilai *lambda* selanjutnya cenderung konvergen atau tidak mengalami perubahan yang signifikan. Hal ini menunjukkan bahwa nilai *lambda* yang terlalu kecil dapat menyebabkan nilai penskalaan ruang pemetaan kernel tidak stabil sehingga nilai *error rate* meningkat.

6.1.2 Hasil Pengujian Parameter Nilai *Sigma* (σ)

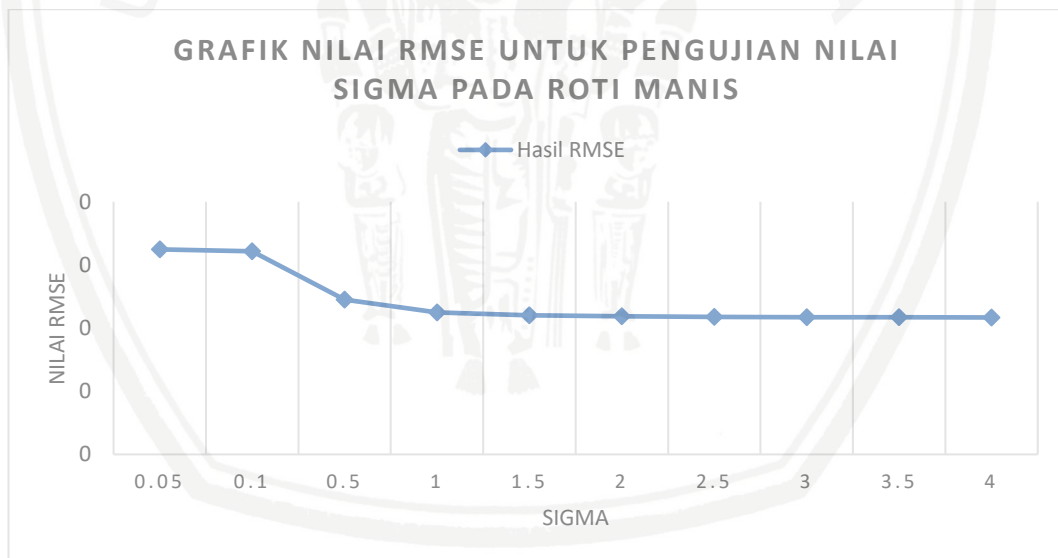
Pengujian selanjutnya adalah menguji parameter nilai *sigma*. Pengujian dilakukan untuk mendapatkan nilai *sigma* terbaik sehingga dapat menghasilkan solusi peramalan terbaik. Ada 10 nilai *sigma* yang akan diuji, dan nilai parameter lain akan ditentukan terlebih dahulu, namun untuk nilai *lambda* akan digunakan nilai 5 yang merupakan hasil terbaik dari pengujian sebelumnya. Berikut nilai parameter lain yang akan digunakan untuk mendapatkan nilai *sigma* terbaik:

- Coefisien Learning Rate* (cLR) = 0,01
- Complexity* = 5
- Epsilon* = 0,00001
- Lambda* = 5

e. Jumlah iterasi = 10

Tabel 6.2 Hasil Pengujian Nilai Parameter *Sigma* (σ) untuk Roti Manis

No.	Nilai <i>Sigma</i>	Hasil RMSE
1	0,05	0,1623
2	0,1	0,1609
3	0,5	0,1226
4	1	0,1124
5	1,5	0,1101
6	2	0,1093
7	2,5	0,1089
8	3	0,1087
9	3,5	0,1085
10	4	0,1084



Gambar 6.2 Grafik Nilai RMSE untuk Pengujian Nilai *Sigma* pada Roti Manis

Berdasarkan pengujian *sigma* yang dilakukan terhadap roti manis, didapatkan hasil RMSE paling optimal yaitu 0,1101 dengan nilai *sigma* = 1,5. Pada grafik 6.2 terlihat pada nilai *sigma* 0,1 grafik mengalami penurunan lalu mulai konvergen pada nilai *sigma* 1,5 sampai sampai nilai *sigma* 4. Hal ini menunjukkan bahwa nilai *sigma* yang terlalu kecil dapat menyebabkan persebaran data yang tidak sesuai,

dan menyebabkan nilai error rate meningkat. Sebagaimana pengertiannya, bahwa nilai σ merupakan konstanta dari fungsi *Kernel RBF* untuk mengatur persebaran data kedalam dimensi fitur yang lebih tinggi. (Furi, Jordi, & Saepudin, 2015).

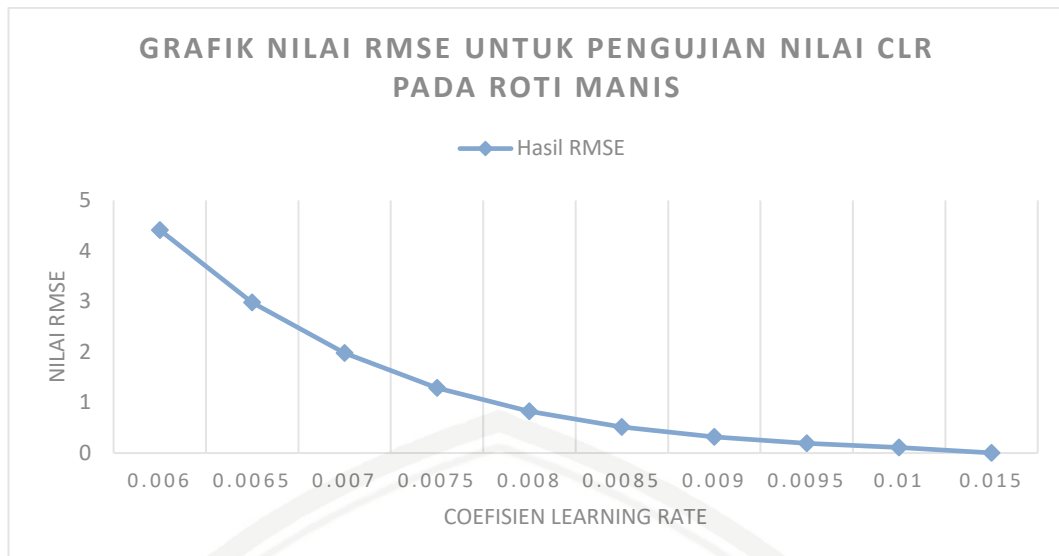
6.1.3 Hasil Pengujian Parameter Nilai *Coeffisien Learning Rate (cLR)*

Selanjutnya dilakukan pengujian parameter nilai *Coeffisien Learning Rate (cLR)*. Pengujian dilakukan untuk mendapatkan nilai *cLR* terbaik sehingga dapat menghasilkan nilai evaluasi yang terbaik. Ada 10 parameter *cLR* yang akan diuji, dan nilai parameter lain akan ditentukan terlebih dahulu. Untuk nilai λ dan σ akan menggunakan nilai terbaik yang sudah didapatkan pada pengujian sebelumnya, yaitu untuk nilai $\lambda = 5$ dan nilai $\sigma = 1$. Berikut nilai parameter lain yang akan digunakan untuk mendapatkan nilai *cLR* terbaik:

- $\lambda = 5$
- $\sigma = 1,5$
- Complexity = 5
- Epsilon = 0,00001
- Jumlah iterasi = 10

Tabel 6.3 Hasil Pengujian Nilai Parameter *cLR* untuk Roti Manis

No.	Nilai <i>cLR</i>	Hasil RMSE
1	0,006	4,3759
2	0,0065	2,9480
3	0,007	1,9544
4	0,0075	1,2734
5	0,008	0,8143
6	0,0085	0,5104
7	0,009	0,3131
8	0,0095	0,1878
9	0,01	0,1101
10	0,015	0,0035



Gambar 6.3 Grafik Nilai RMSE untuk Pengujian Nilai *cLR* pada Roti Manis

Berdasarkan pengujian yang sudah ditunjukkan pada Tabel 6.3, nilai *cLR* terbaik ada pada 0,015 yang mempunyai nilai RMSE paling kecil yaitu 0,0035. Pada grafik 6.3 terlihat bahwa nilai RMSE cenderung mengalami penurunan seiring dengan nilai *cLR* yang semakin besar. Hal ini menunjukkan bahwa nilai *cLR* yang terlalu kecil dapat menyebabkan peningkatan nilai *error rate* dan menghasilkan nilai peramalan yang buruk. Hal ini dikarenakan nilai *cLR* mempengaruhi nilai *gamma*, dan apabila nilai *gamma* keluar dari batas solusi maka nilai *alpha* dan *alpha star* yang didapatkan tidak pas dan menyebabkan nilai evaluasinya sangat besar. Parameter *Coefisien Learning Rate (cLR)* sendiri merupakan konstanta laju pembelajaran (Vijayakumar, & Wu, 1999).

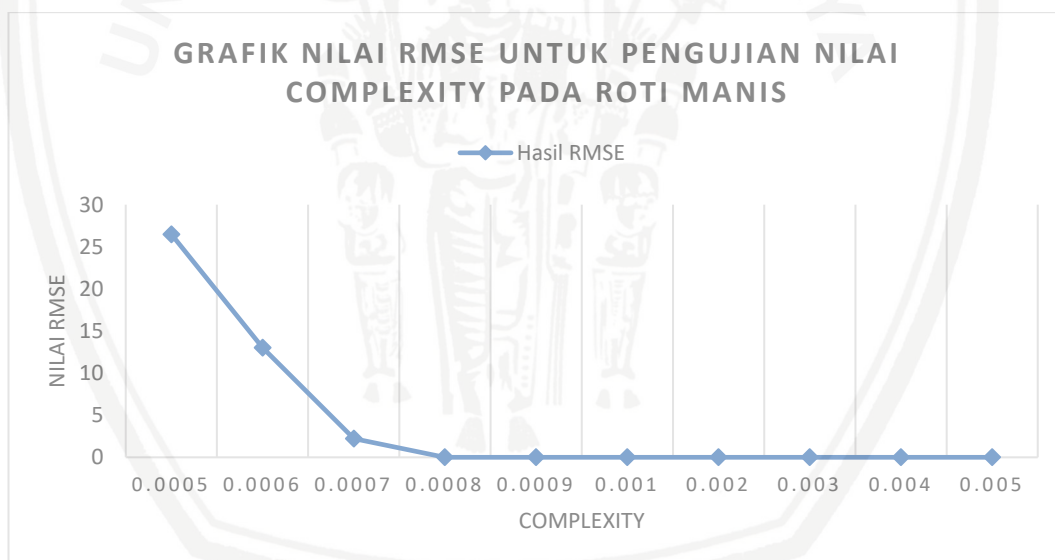
6.1.4 Hasil Pengujian Parameter Nilai *Complexity (C)*

Setelah menguji nilai *cLR* maka selanjutnya dilakukan pengujian nilai *complexity*. Pengujian dilakukan untuk mendapatkan nilai *complexity* terbaik sehingga didapatkan nilai hasil evaluasi terkecil. Untuk pengujian nilai *complexity* digunakan 10 parameter nilai. Untuk nilai *lambda* menggunakan nilai 5, nilai *sigma* 1, dan nilai *cLR* 0,015. Untuk nilai lainnya akan ditentukan seperti berikut:

- a. *Lambda* = 5
- b. *Sigma* = 1,5
- c. *cLR* = 0,015
- d. *Epsilon* = 0,00001
- e. Jumlah iterasi = 10

Tabel 6.4 Hasil Pengujian Nilai Parameter *Complexity* untuk Roti Manis

No.	Nilai <i>Complexity</i>	Hasil RMSE
1	0,0005	26,4875
2	0,0006	13,0248
3	0,0007	2,2285
4	0,0008	0,0035
5	0,0009	0,0035
6	0,001	0,0035
7	0,002	0,0035
8	0,003	0,0035
9	0,004	0,0035
10	0,005	0,0035



Gambar 6.4 Grafik Nilai RMSE Pengujian Nilai *Complexity* pada Roti Manis

Parameter *Complexity* merupakan batas penalti toleransi terhadap kesalahan sebuah peramalan (Furi, Jordi, & Saepudin, 2015). Berdasarkan hasil pengujian parameter nilai C yang sudah dilakukan, ditunjukkan pada Tabel 6.4 ditunjukkan bahwa nilai C paling optimal adalah 0,0008 yang mempunyai nilai evaluasi RMSE sebesar 0,0035. Grafik 6.4 menunjukkan bahwa pada nilai C 0,0005 cenderung mempunyai nilai RMSE yang tinggi, lalu grafik menurun dan terus mengalami

penurunan sampai pada nilai C 0,0008. Pada nilai C 0,0008 grafik terlihat konvergen dan tidak mengalami perubahan lagi. Hal ini sesuai dengan pernyataan Furi, Jordi, & Saepudin (2015) yang menyatakan bahwa semakin besar nilai *Complexity* maka semakin menjadikan model peramalan semakin tidak mentoleransi kesalahan, sehingga memberikan nilai peramalan yang baik.

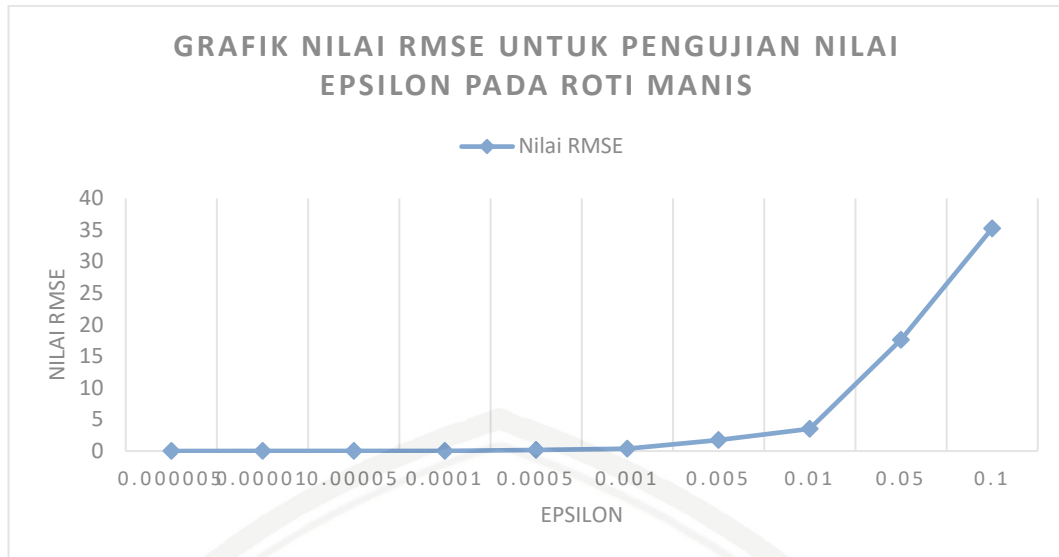
6.1.5 Hasil Pengujian Parameter Nilai *Epsilon*

Selanjutnya dilakukan pengujian terhadap nilai *epsilon*. Pengujian dimaksudkan agar mendapat nilai *epsilon* terbaik sehingga didapatkan nilai evaluasi yang kecil. Untuk pengujian nilai *epsilon* akan digunakan 10 parameter nilai. Untuk jumlah iterasi menggunakan iterasi 10, sedangkan untuk nilai lainnya menggunakan nilai terbaik yang sudah didapatkan dari pengujian sebelumnya. Berikut nilai lain yang digunakan:

- a. $\lambda = 5$
- b. $\sigma = 1,5$
- c. $cLR = 0,015$
- d. $Complexity = 0,0008$
- e. Jumlah iterasi = 10

Tabel 6.5 Hasil Pengujian Nilai Parameter *Epsilon* untuk Roti Manis

No.	Nilai <i>Epsilon</i>	Hasil RMSE
1	0,000005	0,0017
2	0,00001	0,0035
3	0,00005	0,0176
4	0,0001	0,0352
5	0,0005	0,1763
6	0,001	0,3526
7	0,005	1,7631
8	0,01	3,5262
9	0,05	17,6313
10	0,1	35,2625



Gambar 6.5 Grafik Nilai RMSE untuk Pengujian Nilai *Epsilon* pada Roti Manis

Parameter *epsilon* digunakan untuk mengatur batas kesalahan fungsi regresi $f(x)$. Nilai *epsilon* tersebut menyelubungi nilai dari fungsi $f(x)$ sehingga akan membentuk daerah yang disebut daerah *error zone*, dan jika nilai $f(x)$ melebihi *error zone* yang terbentuk maka akan dikenakan penalti sebesar nilai C yang sudah ditentukan (Furi, Jordi, & Saepudin, 2015). Berdasarkan pengujian nilai *epsilon* yang sudah dilakukan, terlihat pada Tabel 6.5 bahwa nilai *epsilon* terbaik didapatkan pada nilai *epsilon* 0,000005 yang memiliki nilai evaluasi sebesar 0,0017. Grafik 6.5 menunjukkan bahwa nilai *epsilon* kecil cenderung konstan, lalu pada nilai *epsilon* = 0,0005 grafik mulai terlihat naik dan terus meningkat seiring dengan nilai *epsilon* yang semakin besar, sehingga menunjukkan nilai *epsilon* yang besar akan melakukan proses pembelajaran yang terlalu cepat sehingga hasil yang didapat tidak maksimal. Nilai *epsilon* yang terlalu besar juga dapat menyebabkan pencarian solusi menjadi keluar batas.

6.1.6 Hasil Pengujian Jumlah Iterasi

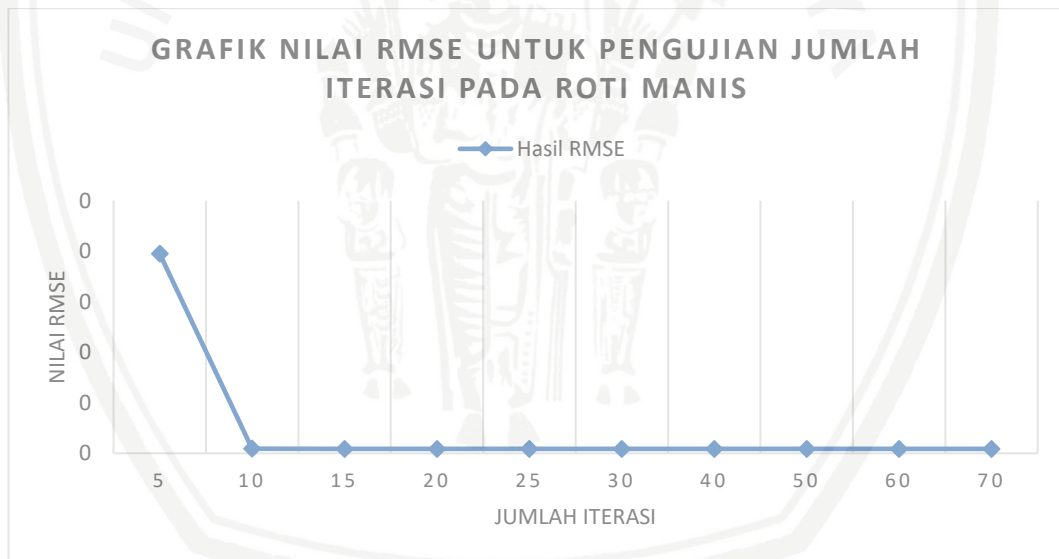
Untuk pengujian yang terakhir adalah melakukan pengujian terhadap jumlah iterasi. Pengujian dilakukan agar didapatkan nilai evaluasi terkecil sehingga proses peramalan semakin baik. Ada 10 parameter jumlah iterasi yang digunakan, sedangkan untuk nilai lainnya menggunakan nilai terbaik yang sudah didapatkan pada pengujian sebelumnya, berikut nilai lain yang digunakan:

- a. $\lambda = 5$
- b. $\sigma = 1,5$
- c. $cLR = 0,015$
- d. $Complexity = 0,0008$

e. $\epsilon = 0,000005$

Tabel 6.6 Hasil Pengujian Iterasi untuk Roti Manis

No.	Jumlah Iterasi	Hasil RMSE
1	5	0,078939534300966
2	10	0,0017916568074186
3	15	0,0017631386324354
4	20	0,0017631275855779
5	25	0,0017631275813001
6	30	0,0017631275812996
7	40	0,0017631275812985
8	50	0,0017631275812969
9	60	0,0017631275813012
10	70	0,0017631275812969



Gambar 6.6 Grafik Nilai RMSE untuk Pengujian Jumlah Iterasi pada Roti Manis

Berdasarkan pengujian jumlah iterasi diatas, dapat disimpulkan bahwa jumlah iterasi sangat berpengaruh besar terhadap proses *sequential learning*. Pada pengujian jumlah iterasi ditunjukkan pada Tabel 6.6 bahwa jumlah iterasi terbaik didapatkan pada jumlah iterasi 50 yang menghasilkan nilai evaluasi sebesar 0,0017631275812969. Terlihat pada grafik 6.6 bahwa grafik mengalami penurunan pada iterasi ke 10 dan mulai konvergen setelahnya. Pengujian ini

menunjukkan bahwa semakin besar jumlah iterasi maka algoritme SVR akan semakin teliti untuk dan observasi terhadap pola data pun semakin meningkat.

6.1.7 Kesimpulan Hasil Pengujian

Pada kesimpulan hasil pengujian ini dijelaskan bahwa pada kasus penjualan roti manis pada Toko Roti Harum Bakery menghasilkan nilai evaluasi menggunakan RMSE sebesar 0,0017 dengan parameter nilai $\lambda = 5$, $\sigma = 1,5$, $cLR = 0,015$, $complexity = 0,0008$, $\epsilon = 0,000005$ dan jumlah iterasi sebanyak 50. Hasil ini terbilang sangat baik karena hasil evaluasi sangat dekat dengan angka 0. Berikut perbandingan nilai hasil prediksi dan nilai aktual pada roti manis menggunakan algoritme SVR dengan parameter diatas:

Tabel 6.7 Perbandingan Nilai Prediksi dan Nilai Aktual Penjualan Roti Manis

Penjualan Roti Manis Bulan Januari 2018	
Nilai Prediksi	Nilai Aktual
129,9982	130
355,9982	356
166,9982	167
234,9982	235
109,9982	110

6.2 Hasil Pengujian Roti Cake

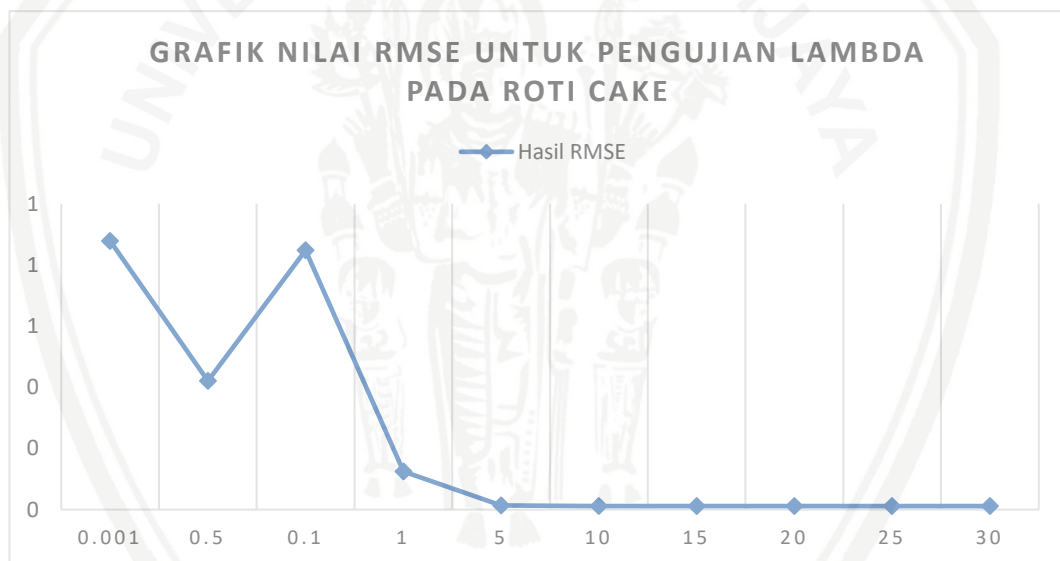
6.2.1 Hasil Pengujian Parameter Nilai λ

Untuk pengujian pertama yang dilakukan adalah pengujian parameter nilai λ terbaik sehingga bisa menghasilkan solusi atau peramalan yang baik dalam kasus permasalahan ini. Ada 10 nilai λ yang akan diuji pada bab ini, dan nilai parameter lain akan ditentukan dahulu. Berikut nilai parameter lain yang akan digunakan untuk mendapatkan nilai λ terbaik:

- Coefisien Learning Rate (cLR) = 0,01
- Complexity = 5
- Epsilon = 0,00001
- Sigma = 0,5
- Jumlah iterasi = 10

Tabel 6.8 Hasil Pengujian Nilai Parameter *Lambda* (λ) untuk Roti Cake

No.	Nilai <i>Lambda</i>	Hasil RMSE
1	0,001	0,8783
2	0,5	0,4204
3	0,1	0,8479
4	1	0,1248
5	5	0,0140
6	10	0,0120
7	15	0,0117
8	20	0,0116
9	25	0,0115
10	30	0,0115



Gambar 6.7 Grafik Nilai RMSE untuk Pengujian *Lambda* pada Roti Cake

Parameter *lambda* berfungsi untuk menunjukkan ukuran skalar untuk pemetaan ruang pada kernel SVR (Vijayakumar, & Wu, 1999). Berdasarkan hasil pengujian parameter *lambda* yang dilakukan, Roti Cake memiliki hasil RMSE paling optimal pada parameter *lambda* 5. Seperti ditunjukkan pada gambar grafik 6.7 terlihat nilai RMSE terhadap nilai *lambda* cenderung menurun, tetapi terjadi peningkatan nilai RMSE pada nilai *lambda* 0,1 yang kemudian nilai RMSE kembali menurun dan memberikan penurunan yang signifikan sampai nilai *lambda* 5, karena untuk nilai *lambda* selanjutnya cenderung konvergen atau tidak mengalami perubahan yang signifikan. Hal ini menunjukkan bahwa nilai *lambda*

yang terlalu kecil dapat menyebabkan nilai penskalaan ruang pemetaan kernel tidak stabil sehingga nilai *error rate* meningkat.

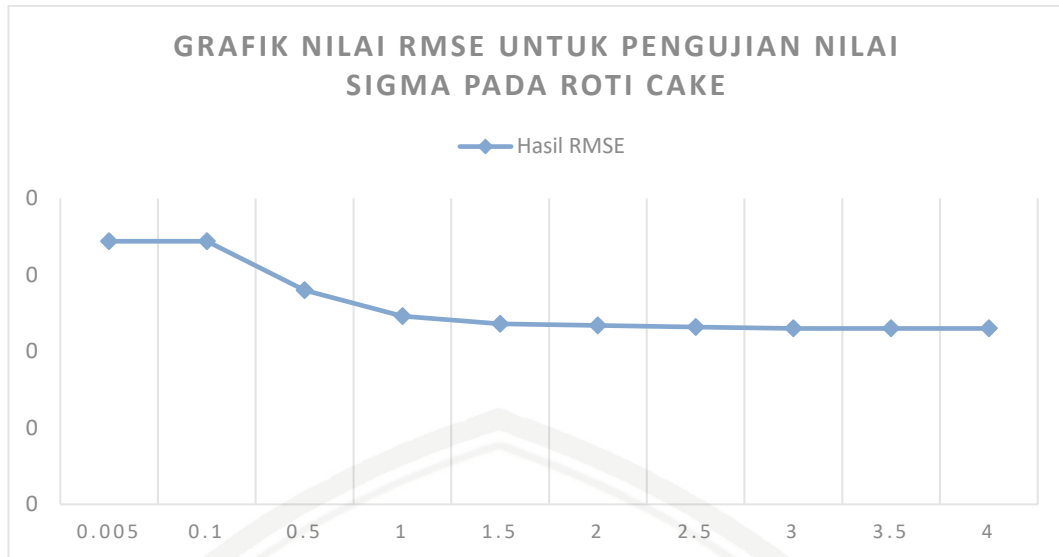
6.2.2 Hasil Pengujian Parameter Nilai *Sigma* (σ)

Pengujian selanjutnya adalah menguji parameter nilai *sigma*. Pengujian dilakukan untuk mendapatkan nilai *sigma* terbaik sehingga dapat menghasilkan solusi peramalan terbaik. Ada 10 nilai *sigma* yang akan diuji, dan nilai parameter lain akan ditentukan terlebih dahulu, namun untuk nilai lambda akan digunakan nilai 5 yang merupakan hasil terbaik dari pengujian sebelumnya. Berikut nilai parameter lain yang akan digunakan untuk mendapatkan nilai *sigma* terbaik:

- Coefisien Learning Rate* (cLR) = 0,01
- Complexity* = 5
- Epsilon* = 0,00001
- Lambda* = 5
- Jumlah iterasi = 10

Tabel 6.9 Hasil Pengujian Nilai Parameter *Sigma* (σ) untuk Roti Cake

No.	Nilai <i>Sigma</i>	Hasil RMSE
1	0,05	0,0172
2	0,1	0,0172
3	0,5	0,0140
4	1	0,0123
5	1,5	0,0118
6	2	0,0117
7	2,5	0,0116
8	3	0,0115
9	3,5	0,0115
10	4	0,0115



Gambar 6.8 Grafik Nilai RMSE untuk Pengujian *Sigma* pada Roti Cake

Berdasarkan pengujian *sigma* yang dilakukan terhadap roti cake, didapatkan hasil RMSE paling optimal yaitu 0,0118 dengan nilai *sigma* = 1,5. Pada grafik 6.8 terlihat pada nilai *sigma* 0,1 grafik mengalami penurunan lalu mulai konvergen pada nilai *sigma* 1,5 sampai sampai nilai *sigma* 4. Hal ini menunjukkan bahwa nilai *sigma* yang terlalu kecil dapat menyebabkan persebaran data yang tidak sesuai, dan menyebabkan nilai error rate meningkat. Sebagaimana pengertiannya, bahwa nilai *sigma* merupakan konstanta dari fungsi *Kernel RBF* untuk mengatur persebaran data kedalam dimensi fitur yang lebih tinggi. (Furi, Jordi, & Saepudin, 2015).

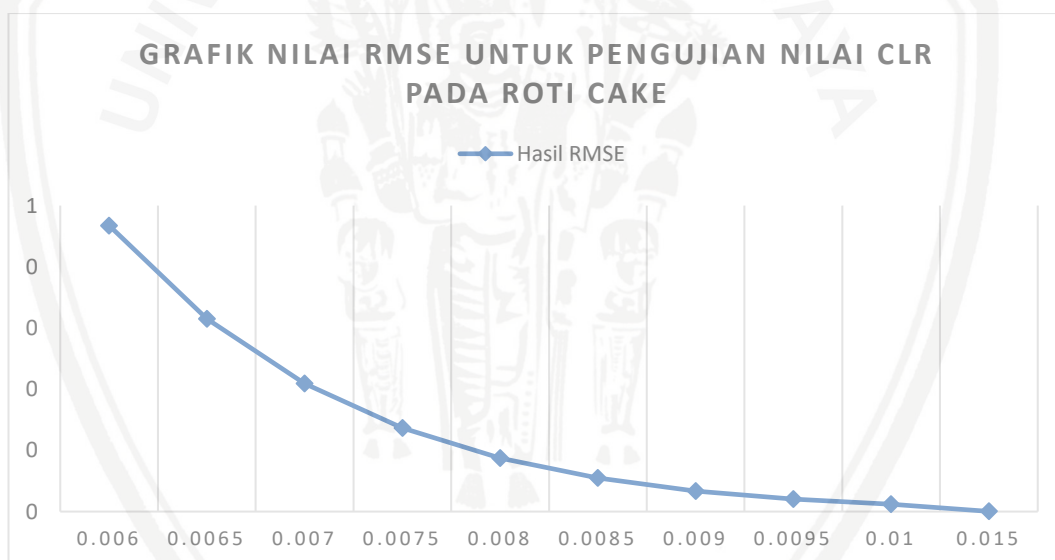
6.2.3 Hasil Pengujian Parameter Nilai *Coefisien Learning Rate (cLR)*

Selanjutnya dilakukan pengujian parameter nilai *Coefisien Learning Rate (cLR)*. Pengujian dilakukan untuk mendapatkan nilai *cLR* terbaik sehingga dapat menghasilkan nilai evaluasi yang terbaik. Ada 10 parameter *cLR* yang akan diuji, dan nilai parameter lain akan ditentukan terlebih dahulu. Untuk nilai *lambda* dan *sigma* akan menggunakan nilai terbaik yang sudah didapatkan pada pengujian sebelumnya, yaitu untuk nilai *lambda* = 5 dan nilai *sigma* = 1,5. Berikut nilai parameter lain yang akan digunakan untuk mendapatkan nilai *cLR* terbaik:

- Lambda* = 5
- Sigma* = 1,5
- Complexity* = 5
- Epsilon* = 0,00001
- Jumlah iterasi = 10

Tabel 6.10 Hasil Pengujian Nilai Parameter cLR untuk Roti Cake

No.	Nilai cLR	Hasil RMSE
1	0,006	0,4674
2	0,0065	0,3151
3	0,007	0,2091
4	0,0075	0,1364
5	0,008	0,0873
6	0,0085	0,0547
7	0,009	0,0336
8	0,0095	0,0202
9	0,01	0,0118
10	0,015	0,0003

Gambar 6.9 Grafik Nilai RMSE untuk Pengujian Nilai cLR pada Roti Cake

Berdasarkan pengujian yang sudah ditunjukkan pada Tabel 6.10, nilai cLR terbaik ada pada 0,015 yang mempunyai nilai RMSE paling kecil yaitu 0,0003. Pada grafik 6.9 terlihat bahwa nilai RMSE cenderung mengalami penurunan seiring dengan nilai cLR yang semakin besar. Hal ini menunjukkan bahwa nilai cLR yang terlalu kecil dapat menyebabkan peningkatan nilai *error rate* dan menghasilkan nilai peramalan yang buruk. Hal ini dikarenakan nilai cLR mempengaruhi nilai *gamma*, dan apabila nilai *gamma* keluar dari batas solusi maka nilai *alpha* dan *alpha star* yang didapatkan tidak pas dan menyebabkan nilai evaluasinya sangat

besar. Parameter *Coefisien Learning Rate (cLR)* sendiri merupakan konstanta laju pembelajaran (Vijayakumar, & Wu, 1999).

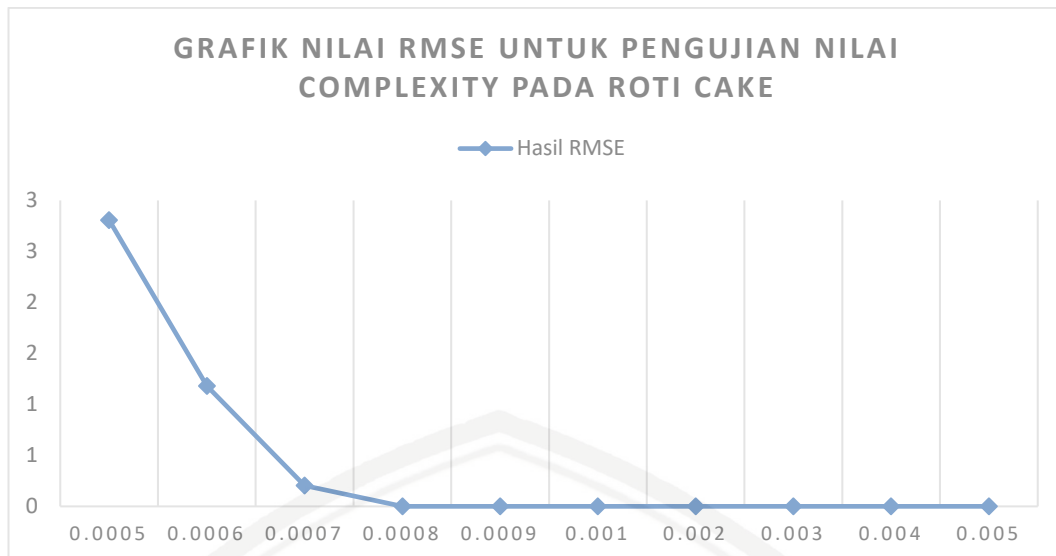
6.2.4 Hasil Pengujian Parameter Nilai *Complexity (C)*

Setelah menguji nilai *cLR* maka selanjutnya dilakukan pengujian nilai *complexity*. Pengujian dilakukan untuk mendapatkan nilai *complexity* terbaik sehingga didapatkan nilai hasil evaluasi terkecil. Untuk pengujian nilai *complexity* digunakan 10 parameter nilai. Untuk nilai *lambda* menggunakan nilai 5, nilai *sigma* 1,5, dan nilai *cLR* 0,015. Untuk nilai lainnya akan ditentukan seperti berikut:

- a. *Lambda* = 5
- b. *Sigma* = 1,5
- c. *cLR* = 0,015
- d. *Epsilon* = 0,00001
- e. Jumlah iterasi = 10

Tabel 6.11 Hasil Pengujian Nilai Parameter *Complexity* untuk Roti Cake

No.	Nilai <i>Complexity</i>	Hasil RMSE
1	0,0005	2.8068
2	0,0006	1,1801
3	0,0007	0,2046
4	0,0008	0,0003
5	0,0009	0,0003
6	0,001	0,0003
7	0,002	0,0003
8	0,003	0,0003
9	0,004	0,0003
10	0,005	0,0003



Gambar 6.10 Grafik Nilai RMSE Pengujian Nilai *Complexity* pada Roti Cake

Parameter *Complexity* merupakan batas penalti toleransi terhadap kesalahan sebuah peramalan (Furi, Jordi, & Saepudin, 2015). Berdasarkan hasil pengujian parameter nilai C yang sudah dilakukan, ditunjukkan pada Tabel 6.11 ditunjukkan bahwa nilai C paling optimal adalah 0,0008 yang mempunyai nilai evaluasi RMSE sebesar 0,0003. Grafik 6.10 menunjukkan bahwa pada nilai C 0,0005 cenderung mempunyai nilai RMSE yang tinggi, lalu grafik menurun dan terus mengalami penurunan sampai pada nilai C 0,0008. Pada nilai C 0,0008 grafik terlihat konvergen dan tidak mengalami perubahan lagi. Hal ini sesuai dengan pernyataan Furi, Jordi, & Saepudin (2015) yang menyatakan bahwa semakin besar nilai *Complexity* maka semakin menjadikan model peramalan semakin tidak mentoleransi kesalahan, sehingga memberikan nilai peramalan yang baik.

6.2.5 Hasil Pengujian Parameter Nilai *Epsilon*

Selanjutnya dilakukan pengujian terhadap nilai *epsilon*. Pengujian dimaksudkan agar mendapat nilai *epsilon* terbaik sehingga didapatkan nilai evaluasi yang kecil. Untuk pengujian nilai *epsilon* akan digunakan 10 parameter nilai. Untuk jumlah iterasi menggunakan iterasi 10, sedangkan untuk nilai lainnya menggunakan nilai terbaik yang sudah didapatkan dari pengujian sebelumnya. Berikut nilai lain yang digunakan:

- $\lambda = 5$
- $\sigma = 1,5$
- $cLR = 0,015$
- $Complexity = 0,0008$
- Jumlah iterasi = 10

Tabel 6.12 Hasil Pengujian Nilai Parameter *Epsilon* untuk Roti Cake

No.	Nilai <i>Epsilon</i>	Hasil RMSE
1	0,000005	0,0001
2	0,00001	0,0003
3	0,00005	0,0019
4	0,0001	0,0038
5	0,0005	0,0193
6	0,001	0,0386
7	0,005	0,1931
8	0,01	0,3863
9	0,05	1,9315
10	0,1	1,8630



Gambar 6.11 Grafik Nilai RMSE untuk Pengujian Nilai *Epsilon* pada Roti cake

Parameter *epsilon* digunakan untuk mengatur batas kesalahan fungsi regresi $f(x)$. Nilai *epsilon* tersebut menyelubungi nilai dari fungsi $f(x)$ sehingga akan membentuk daerah yang disebut daerah *error zone*, dan jika nilai $f(x)$ melebihi *error zone* yang terbentuk maka akan dikenakan penalti sebesar nilai C yang sudah ditentukan (Furi, Jordi, & Saepudin, 2015). Berdasarkan pengujian nilai *epsilon* yang sudah dilakukan, terlihat pada Tabel 6.12 bahwa nilai *epsilon* terbaik didapatkan pada nilai *epsilon* 0,000005 yang memiliki nilai evaluasi sebesar 0,0001. Grafik 6.11 menunjukkan bahwa nilai *epsilon* kecil cenderung konstan, lalu

pada nilai $\epsilon = 0,0005$ grafik mulai terlihat naik dan terus meningkat seiring dengan nilai ϵ yang semakin besar, sehingga menunjukkan nilai ϵ yang besar akan melakukan proses pembelajaran yang terlalu cepat sehingga hasil yang didapat tidak maksimal. Nilai ϵ yang terlalu besar juga dapat menyebabkan pencarian solusi menjadi keluar batas.

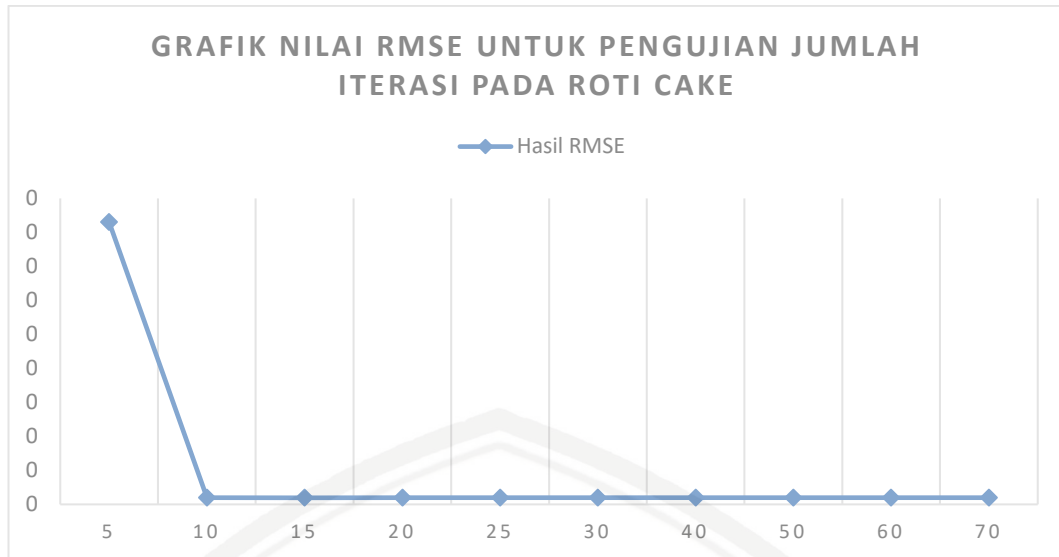
6.2.6 Hasil Pengujian Jumlah Iterasi

Untuk pengujian yang terakhir adalah melakukan pengujian terhadap jumlah iterasi. Pengujian dilakukan agar didapatkan nilai evaluasi terkecil sehingga proses peramalan semakin baik. Ada 10 parameter jumlah iterasi yang digunakan, sedangkan untuk nilai lainnya menggunakan nilai terbaik yang sudah didapatkan pada pengujian sebelumnya, berikut nilai lain yang digunakan:

- $\lambda = 10$
- $\sigma = 0,1$
- $cLR = 0,01$
- $Complexity = 0,1$
- $\epsilon = 0,01$

Tabel 6.13 Hasil Pengujian Iterasi untuk Roti Cake

No.	Jumlah Iterasi	Hasil RMSE
1	5	0,0083016430300742
2	10	0,00019603571704973
3	15	0,00019312572343983
4	20	0,00019315161747642
5	25	0,00019315161705278
6	30	0,00019315161705214
7	40	0,00019315161705241
8	50	0,00019315161705187
9	60	0,00019315161705214
10	70	0,00019315161705241



Gambar 6.12 Grafik Nilai RMSE untuk Pengujian Jumlah Iterasi pada Roti Cake

Berdasarkan pengujian jumlah iterasi diatas, dapat disimpulkan bahwa jumlah iterasi sangat berpengaruh besar terhadap proses *sequential learning*. Pada pengujian jumlah iterasi ditunjukkan pada Tabel 6.13 bahwa jumlah iterasi terbaik didapatkan pada jumlah iterasi 50 yang menghasilkan nilai evaluasi sebesar 0,00019315161705187. Terlihat pada grafik 6.12 bahwa grafik mengalami penurunan pada iterasi ke 10 dan mulai konvergen setelahnya. Pengujian ini menunjukkan bahwa semakin besar jumlah iterasi maka algoritme SVR akan semakin teliti untuk dan observasi terhadap pola data pun semakin meningkat.

6.2.7 Kesimpulan Hasil Pengujian

Pada kesimpulan hasil pengujian ini dijelaskan bahwa pada kasus penjualan roti cake pada Toko Roti Harum Bakery menghasilkan nilai evaluasi menggunakan RMSE sebesar 0,0001 dengan parameter nilai $\lambda = 5$, $\sigma = 1,5$, $cLR = 0,015$, $complexity = 0,0008$, $\epsilon = 0,000005$ dan jumlah iterasi sebanyak 50. Hasil ini terbilang sangat baik karena hasil evaluasi sangat dekat dengan angka 0. Berikut perbandingan nilai hasil prediksi dan nilai aktual pada roti cake menggunakan algoritme SVR dengan parameter diatas:

Tabel 6.14 Perbandingan Nilai Prediksi dan Nilai Aktual Penjualan Roti Cake

Penjualan Roti Cake Bulan Januari 2018	
Nilai Prediksi	Nilai Aktual
20,9998	21
30,9998	31

Nilai Prediksi	Nilai Aktual
26,9998	27
16,9998	17
13,9998	14

6.3 Hasil Pengujian Roti Tawar

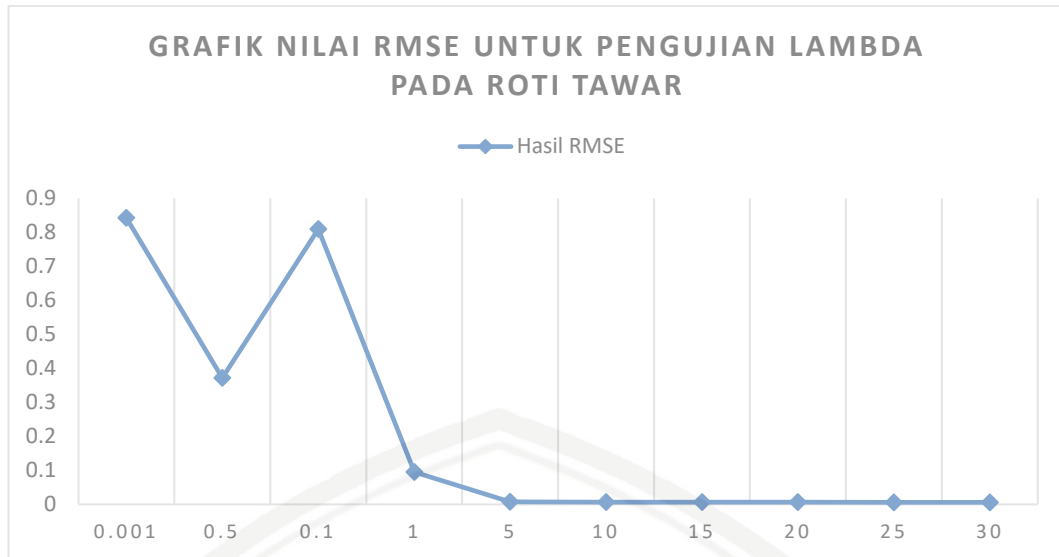
6.3.1 Hasil Pengujian Parameter Nilai *Lambda* (λ)

Untuk pengujian pertama yang dilakukan adalah pengujian parameter nilai *lambda* (λ) terbaik sehingga bisa menghasilkan solusi atau peramalan yang baik dalam kasus permasalahan ini. Ada 10 nilai *lambda* yang akan diuji pada bab ini, dan nilai parameter lain akan ditentukan dahulu. Berikut nilai parameter lain yang akan digunakan untuk mendapatkan nilai *lambda* terbaik:

- Coefisien Learning Rate* (cLR) = 0,01
- Complexity* = 5
- Epsilon* = 0,00001
- Sigma* = 0,5
- Jumlah iterasi = 10

Tabel 6.15 Hasil Pengujian Nilai Parameter *Lambda* (λ) untuk Roti Tawar

No.	Nilai <i>Lambda</i>	Hasil RMSE
1	0,001	0,8423
2	0,5	0,3718
3	0,1	0,8101
4	1	0,0949
5	5	0,0076
6	10	0,0063
7	15	0,0061
8	20	0,0061
9	25	0,0060
10	30	0,0060



Gambar 6.13 Grafik Nilai RMSE untuk Pengujian Lambda pada Roti Tawar

Parameter *lambda* berfungsi untuk menunjukkan ukuran skalar untuk pemetaan ruang pada kernel SVR (Vijayakumar, & Wu, 1999). Berdasarkan hasil pengujian parameter *lambda* yang dilakukan, Roti Tawar memiliki hasil RMSE paling optimal pada parameter *lambda* 5. Seperti ditunjukkan pada gambar grafik 6.13 terlihat nilai RMSE terhadap nilai *lambda* cenderung menurun, tetapi terjadi peningkatan nilai RMSE pada nilai *lambda* 0,1 yang kemudian nilai RMSE kembali menurun dan memberikan penurunan yang signifikan sampai nilai *lambda* 5, karena untuk nilai *lambda* selanjutnya cenderung konvergen atau tidak mengalami perubahan yang signifikan. Hal ini menunjukkan bahwa nilai *lambda* yang terlalu kecil dapat menyebabkan nilai penskalaan ruang pemetaan kernel tidak stabil sehingga nilai *error rate* meningkat.

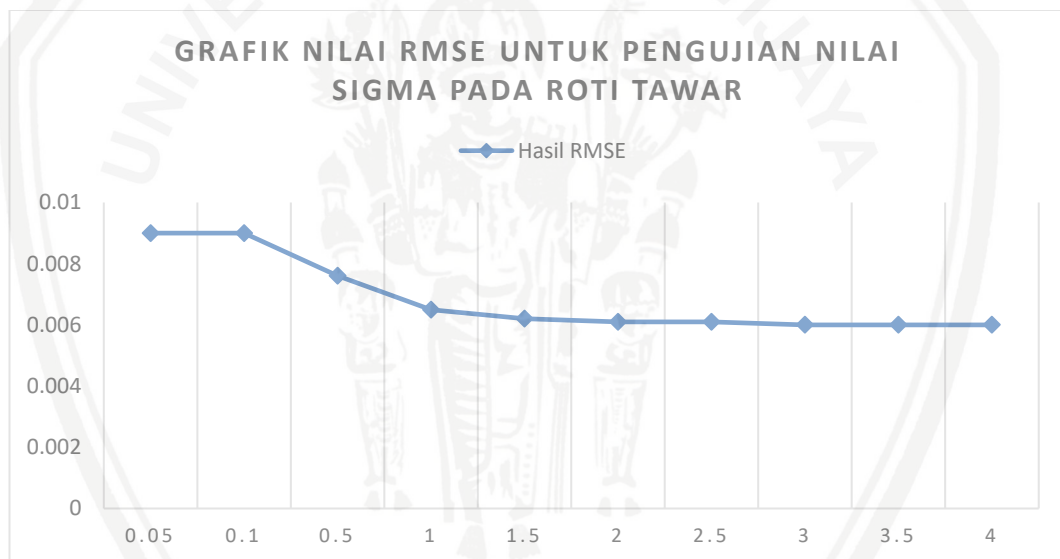
6.3.2 Hasil Pengujian Parameter Nilai *Sigma* (σ)

Pengujian selanjutnya adalah menguji parameter nilai *sigma*. Pengujian dilakukan untuk mendapatkan nilai *sigma* terbaik sehingga dapat menghasilkan solusi peramalan terbaik. Ada 10 nilai *sigma* yang akan diuji, dan nilai parameter lain akan ditentukan terlebih dahulu, namun untuk nilai *lambda* akan digunakan nilai 5 yang merupakan hasil terbaik dari pengujian sebelumnya. Berikut nilai parameter lain yang akan digunakan untuk mendapatkan nilai *sigma* terbaik:

- Coefisien Learning Rate* (cLR) = 0,01
- Complexity* = 5
- Epsilon* = 0,00001
- Lambda* = 5
- Jumlah iterasi = 10

Tabel 6.16 Hasil Pengujian Nilai Parameter σ untuk Roti Tawar

No.	Nilai σ	Hasil RMSE
1	0,05	0,0090
2	0,1	0,0090
3	0,5	0,0076
4	1	0,0065
5	1,5	0,0062
6	2	0,0061
7	2,5	0,0061
8	3	0,0060
9	3,5	0,0060
10	4	0,0060

Gambar 6.14 Grafik Nilai RMSE untuk Pengujian σ pada Roti Tawar

Berdasarkan pengujian σ yang dilakukan terhadap roti tawar, didapatkan hasil RMSE paling optimal yaitu 0,0062 dengan nilai $\sigma = 1,5$. Pada grafik 6.2 terlihat pada nilai σ 0,1 grafik mengalami penurunan lalu mulai konvergen pada nilai σ 1,5 sampai sampai nilai σ 4. Hal ini menunjukkan bahwa nilai σ yang terlalu kecil dapat menyebabkan persebaran data yang tidak sesuai, dan menyebabkan nilai error rate meningkat. Sebagaimana pengertiannya, bahwa nilai σ merupakan konstanta dari fungsi *Kernel RBF* untuk mengatur persebaran data kedalam dimensi fitur yang lebih tinggi. (Furi, Jordi, & Saepudin, 2015).

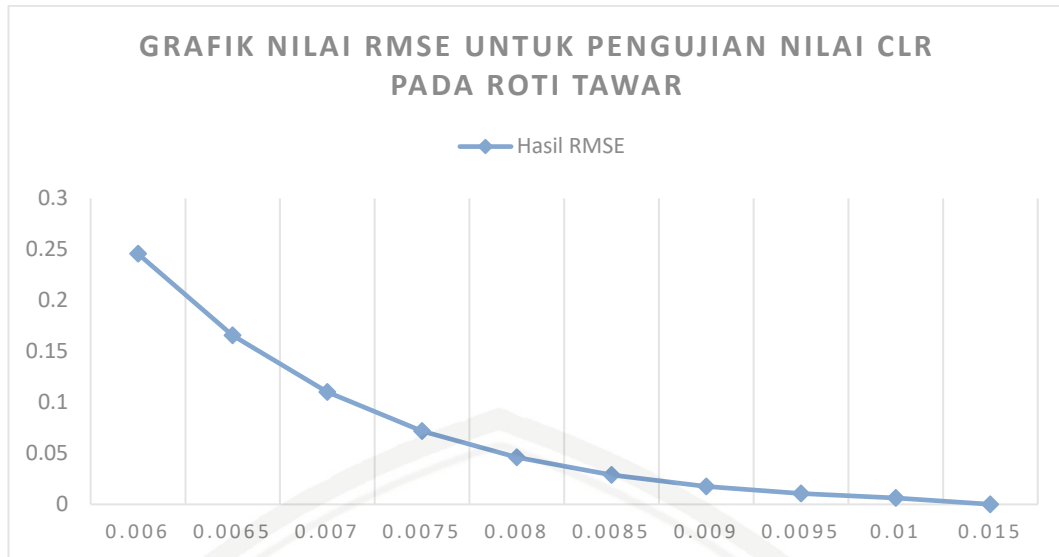
6.3.3 Hasil Pengujian Parameter Nilai *Coefisien Learning Rate (cLR)*

Selanjutnya dilakukan pengujian parameter nilai *Coefisien Learning Rate (cLR)*. Pengujian dilakukan untuk mendapatkan nilai *cLR* terbaik sehingga dapat menghasilkan nilai evaluasi yang terbaik. Ada 10 parameter *cLR* yang akan diuji, dan nilai parameter lain akan ditentukan terlebih dahulu. Untuk nilai *lambda* dan *sigma* akan menggunakan nilai terbaik yang sudah didapatkan pada pengujian sebelumnya, yaitu untuk nilai *lambda* = 5 dan nilai *sigma* = 1,5. Berikut nilai parameter lain yang akan digunakan untuk mendapatkan nilai *cLR* terbaik:

- Lambda* = 5
- Sigma* = 1,5
- Complexity* = 5
- Epsilon* = 0,00001
- Jumlah iterasi = 10

Tabel 6.17 Hasil Pengujian Nilai Parameter *cLR* untuk Roti Tawar

No.	Nilai <i>cLR</i>	Hasil RMSE
1	0,006	0,2457
2	0,0065	0,1657
3	0,007	0,1100
4	0,0075	0,0718
5	0,008	0,0460
6	0,0085	0,0288
7	0,009	0,0177
8	0,0095	0,0106
9	0,01	0,0062
10	0,015	0,0001



Gambar 6.15 Grafik Nilai RMSE untuk Pengujian Nilai *cLR* pada Roti Tawar

Berdasarkan pengujian yang sudah ditunjukkan pada Tabel 6.17, nilai *cLR* terbaik ada pada 0,015 yang mempunyai nilai RMSE paling kecil yaitu 0,0001. Pada grafik 6.15 terlihat bahwa nilai RMSE cenderung mengalami penurunan seiring dengan nilai *cLR* yang semakin besar. Hal ini menunjukkan bahwa nilai *cLR* yang terlalu kecil dapat menyebabkan peningkatan nilai *error rate* dan menghasilkan nilai peramalan yang buruk. Hal ini dikarenakan nilai *cLR* mempengaruhi nilai *gamma*, dan apabila nilai *gamma* keluar dari batas solusi maka nilai *alpha* dan *alpha star* yang didapatkan tidak pas dan menyebabkan nilai evaluasinya sangat besar. Parameter *Coefisien Learning Rate (cLR)* sendiri merupakan konstanta laju pembelajaran (Vijayakumar, & Wu, 1999).

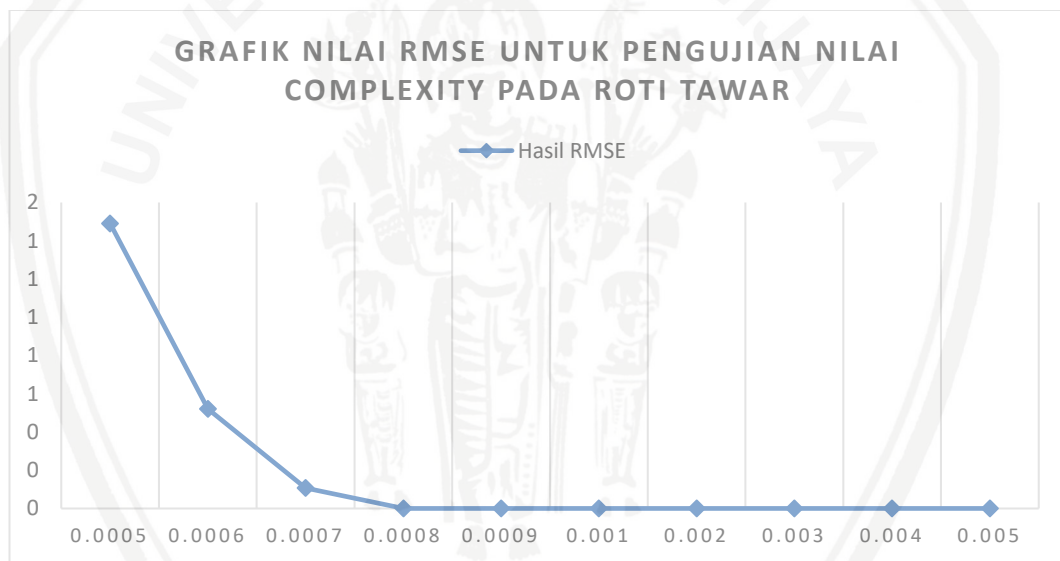
6.3.4 Hasil Pengujian Parameter Nilai *Complexity (C)*

Setelah menguji nilai *cLR* maka selanjutnya dilakukan pengujian nilai *complexity*. Pengujian dilakukan untuk mendapatkan nilai *complexity* terbaik sehingga didapatkan nilai hasil evaluasi terkecil. Untuk pengujian nilai *complexity* digunakan 10 parameter nilai. Untuk nilai *lambda* menggunakan nilai 5, nilai *sigma* 15, dan nilai *cLR* 0,015. Untuk nilai lainnya akan ditentukan seperti berikut:

- a. *Lambda* = 5
- b. *Sigma* = 1,5
- c. *cLR* = 0,015
- d. *Epsilon* = 0,00001
- e. Jumlah iterasi = 10

Tabel 6.18 Hasil Pengujian Nilai Parameter *Complexity* untuk Roti Tawar

No.	Nilai <i>Complexity</i>	Hasil RMSE
1	0,0005	1,4903
2	0,0006	0,5210
3	0,0007	0,1074
4	0,0008	0,0001
5	0,0009	0,0001
6	0,001	0,0001
7	0,002	0,0001
8	0,003	0,0001
9	0,004	0,0001
10	0,005	0,0001



Gambar 6.16 Grafik Nilai RMSE Pengujian Nilai *Complexity* pada Roti Tawar

Parameter *Complexity* merupakan batas penalti toleransi terhadap kesalahan sebuah peramalan (Furi, Jordi, & Saepudin, 2015). Berdasarkan hasil pengujian parameter nilai C yang sudah dilakukan, ditunjukkan pada Tabel 6.18 ditunjukkan bahwa nilai C paling optimal adalah 0,0008 yang mempunyai nilai evaluasi RMSE sebesar 0,0001. Grafik 6.16 menunjukkan bahwa pada nilai C 0,0005 cenderung mempunyai nilai RMSE yang tinggi, lalu grafik menurun dan terus mengalami penurunan sampai pada nilai C 0,0008. Pada nilai C 0,0008 grafik terlihat konvergen dan tidak mengalami perubahan lagi. Hal ini sesuai dengan pernyataan Furi, Jordi, & Saepudin (2015) yang menyatakan bahwa semakin besar nilai

Complexity maka semakin menjadikan model peramalan semakin tidak mentoleransi kesalahan, sehingga memberikan nilai peramalan yang baik.

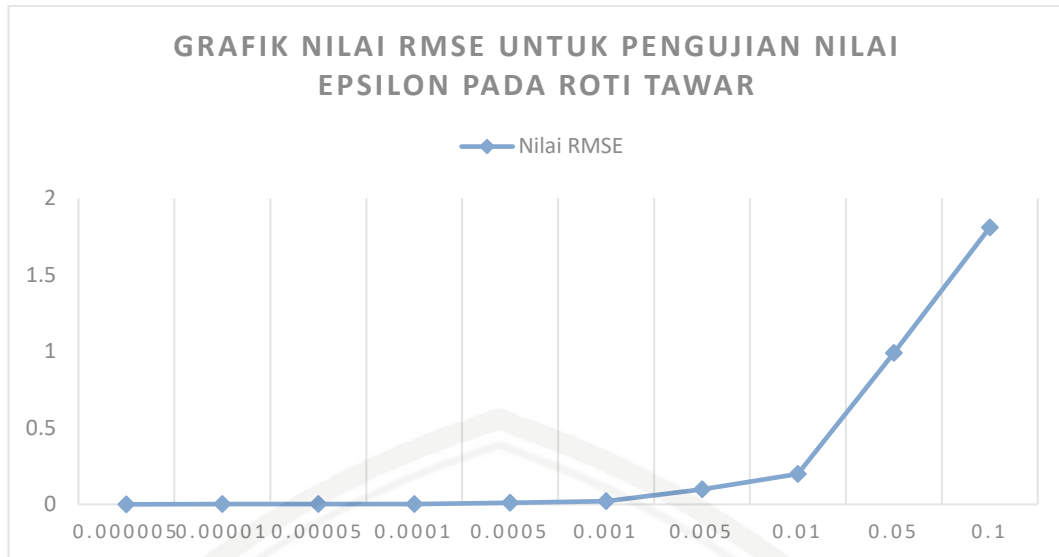
6.3.5 Hasil Pengujian Parameter Nilai *Epsilon*

Selanjutnya dilakukan pengujian terhadap nilai *epsilon*. Pengujian dimaksudkan agar mendapat nilai *epsilon* terbaik sehingga didapatkan nilai evaluasi yang kecil. Untuk pengujian nilai *epsilon* akan digunakan 10 parameter nilai. Untuk jumlah iterasi menggunakan iterasi 10, sedangkan untuk nilai lainnya menggunakan nilai terbaik yang sudah didapatkan dari pengujian sebelumnya. Berikut nilai lain yang digunakan:

- a. $\lambda = 5$
- b. $\sigma = 1,5$
- c. $cLR = 0,015$
- d. $Complexity = 0,0008$
- e. Jumlah iterasi = 10

Tabel 6.19 Hasil Pengujian Nilai Parameter *Epsilon* untuk Roti Tawar

No.	Nilai <i>Epsilon</i>	Hasil RMSE
1	0,000005	0,0001
2	0,00001	0,0002
3	0,00005	0,0009
4	0,0001	0,0019
5	0,0005	0,0099
6	0,001	0,0198
7	0,005	0,0990
8	0,01	0,1981
9	0,05	0,9905
10	0,1	1,9810



Gambar 6.17 Grafik Nilai RMSE untuk Pengujian Nilai *Epsilon* pada Roti Tawar

Parameter *epsilon* digunakan untuk mengatur batas kesalahan fungsi regresi $f(x)$. Nilai *epsilon* tersebut menyelubungi nilai dari fungsi $f(x)$ sehingga akan membentuk daerah yang disebut daerah *error zone*, dan jika nilai $f(x)$ melebihi *error zone* yang terbentuk maka akan dikenakan penalti sebesar nilai C yang sudah ditentukan (Furi, Jordi, & Saepudin, 2015). Berdasarkan pengujian nilai *epsilon* yang sudah dilakukan, terlihat pada Tabel 6.19 bahwa nilai *epsilon* terbaik didapatkan pada nilai *epsilon* 0,000005 yang memiliki nilai evaluasi sebesar 0,0001. Grafik 6.17 menunjukkan bahwa nilai *epsilon* kecil cenderung konstan, lalu pada nilai *epsilon* = 0,0005 grafik mulai terlihat naik dan terus meningkat seiring dengan nilai *epsilon* yang semakin besar, sehingga menunjukkan nilai *epsilon* yang besar akan melakukan proses pembelajaran yang terlalu cepat sehingga hasil yang didapat tidak maksimal. Nilai *epsilon* yang terlalu besar juga dapat menyebabkan pencarian solusi menjadi keluar batas.

6.3.6 Hasil Pengujian Jumlah Iterasi

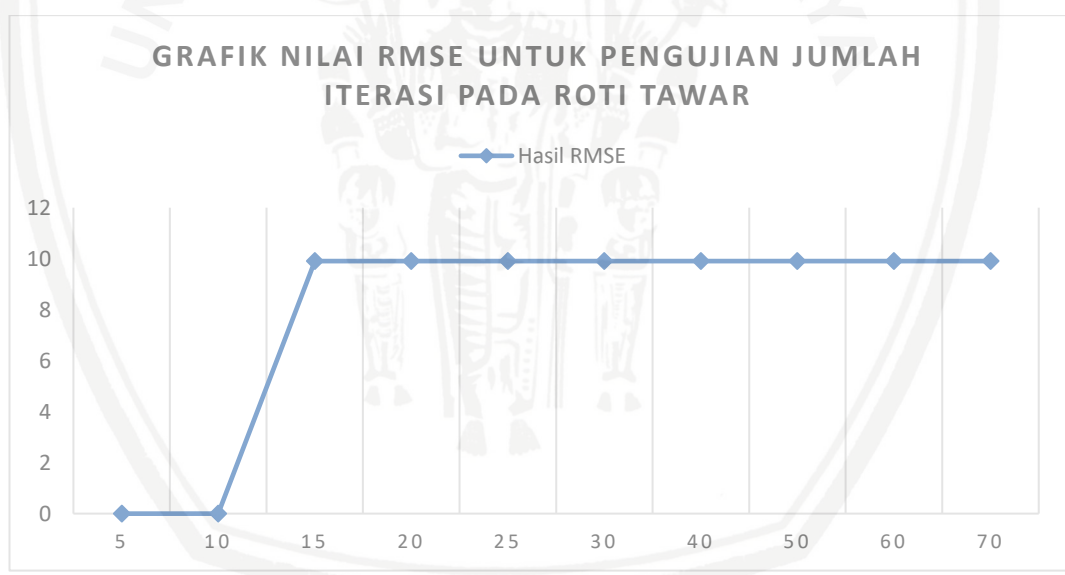
Untuk pengujian yang terakhir adalah melakukan pengujian terhadap jumlah iterasi. Pengujian dilakukan agar didapatkan nilai evaluasi terkecil sehingga proses peramalan semakin baik. Ada 10 parameter jumlah iterasi yang digunakan, sedangkan untuk nilai lainnya menggunakan nilai terbaik yang sudah didapatkan pada pengujian sebelumnya, berikut nilai lain yang digunakan:

- a. $\lambda = 5$
- b. $\sigma = 1,5$
- c. $cLR = 0,015$
- d. $Complexity = 0,0008$

e. $\epsilon = 0,000005$

Tabel 6.20 Hasil Pengujian Iterasi untuk Roti Tawar

No.	Jumlah Iterasi	Hasil RMSE
1	5	0,00441
2	10	0,00010
3	15	9,90527
4	20	9,90521
5	25	9,90521
6	30	9,90521
7	40	9,90521
8	50	9,90521
9	60	9,90521
10	70	9,90521



Gambar 6.18 Grafik Nilai RMSE Pengujian Jumlah Iterasi pada Roti Tawar

Berdasarkan pengujian jumlah iterasi diatas, dapat disimpulkan bahwa jumlah iterasi sangat berpengaruh besar terhadap proses *sequential learning*. Pada pengujian jumlah iterasi ditunjukkan pada Tabel 6.20 bahwa jumlah iterasi terbaik didapatkan pada jumlah iterasi 10 yang menghasilkan nilai evaluasi sebesar 0,00010. Terlihat pada grafik 6.18 bahwa grafik masih konstan di iterasi 5 dan 10, namun pada saat iterasi 15 dan seterusnya, grafik mengalami peningkatan drastis



dan konvergen di nilai RMSE yang tinggi. Pengujian ini menunjukkan bahwa pada kasus penjualan roti tawar ini nilai iterasi yang terlalu tinggi menyebabkan observasi pada pola data menjadi tidak stabil dan menyebabkan nilai *error rate* meningkat.

6.3.7 Kesimpulan Hasil Pengujian

Pada kesimpulan hasil pengujian ini dijelaskan bahwa pada kasus penjualan roti tawar pada Toko Roti Harum Bakery menghasilkan nilai evaluasi menggunakan RMSE sebesar 0,0001 dengan parameter nilai $\lambda = 5$, $\sigma = 1,5$, $cLR = 0,015$, $complexity = 0,0008$, $\epsilon = 0,000005$ dan jumlah iterasi sebanyak 10. Hasil ini terbilang sangat baik karena hasil evaluasi sangat dekat dengan angka 0. Berikut perbandingan nilai hasil prediksi dan nilai aktual pada roti tawar menggunakan algoritme SVR dengan parameter diatas:

Tabel 6.21 Perbandingan Nilai Prediksi dan Nilai Aktual Penjualan Roti Tawar

Penjualan Roti Tawar Bulan Januari 2018	
Nilai Prediksi	Nilai Aktual
10,99989831	11
5,99989990	6
8,99989862	9
12,99989802	13
10,99989834	11

BAB 7 PENUTUP

7.1 Kesimpulan

Dari penelitian yang telah dilakukan menggunakan metode *Support Vector Regression* untuk peramalan penjualan roti pada toko roti Harum Bakery, dapat ditarik kesimpulan sebagai berikut:

1. Berdasarkan pengujian yang telah dilakukan pada bab sebelumnya, yaitu peramalan penjualan pada toko roti Harum Bakery yang dilakukan terhadap 3 jenis roti yaitu roti manis, roti cake, dan roti tawar, didapatkan hasil parameter terbaik adalah $\lambda = 5$, $\sigma = 1,5$, $cLR = 0,015$, $complexity = 0,0008$ dan $\epsilon = 0,000005$. Parameter tersebut berlaku untuk semua jenis roti yang diuji, namun untuk jumlah iterasi terdapat perbedaan pada jenis roti tertentu. Untuk jenis roti manis dan roti cake memiliki nilai evaluasi RMSE terbaik pada iterasi ke 50, namun untuk roti tawar, hasil evaluasi terbaik didapatkan pada iterasi ke 10. Untuk hasil evaluasi peramalan penjualan roti manis menggunakan *Root Mean Square Error* (RMSE) dengan parameter terbaik didapatkan hasil evaluasi sebesar 0,001763, untuk roti cake sebesar 0,0001931, dan roti tawar sebesar 0,00010059.
2. Setelah dilakukan pengujian terhadap 3 jenis roti menggunakan metode *Support Vector Regression* dengan parameter terbaik, dapat disimpulkan bahwa akurasi yang dihasilkan sangat baik dan metode SVR dapat dinyatakan cocok untuk kasus peramalan dengan data seperti pada penelitian ini, karena didapatkan nilai evaluasi RMSE sangat dekat dengan nilai 0.

7.2 Saran

Terdapat beberapa saran yang bisa dilakukan untuk penelitian selanjutnya dengan topik yang serupa, yaitu penambahan pada pengujian. Pada pengujian dapat ditambahkan pengujian normalisasi, sehingga dapat disimpulkan apakah data lebih baik dinormalisasi atau tidak. Lalu dapat juga ditambahkan pengujian fitur, apakah data dengan jenis *time series* mendapatkan hasil peramalan yang berbeda dengan fitur yang berbeda.

DAFTAR PUSTAKA

- Che, J., Wang, J., 2014. Short Term Load Forecasting Using Support Vector Regression Combination Model. *Applied Energy* 132, pp. 602 - 609.
- Conway, D., & White, J. M. (2012). *Machine Learning for Hackers*. (J. Steele, Ed.).
- Furi, R. P., Jondri & Saepudin, D., 2015. Peramalan Financial Time Series Menggunakan *Independent Component Analysis* dan *Support Vector Regression* (Studi Kasus: IHSB dan JII). S1. Telkom University.
- Heizer, Jay dan Barry Render., 2009. *Manajemen Operasi Buku 1 Edisi 9*. Jakarta: Salemba Empat.
- Hosseini, M., Javaherian, A., & Movahed, B. (2014). Determination of permeability index using Stoneley slowness analysis, NMR models, and formation evaluations: a case study from a gas reservoir, south of Iran. *Journal of Applied Geophysics*, 109, 80–87.
- Makridakis, Spyros dkk., 1991. *Metode dan Aplikasi Peramalan*. Edisi Kedua. Jakarta: Erlangga.
- Parto, S. & Sahu, K., 2015. *Normalization: A Preprocessing Stage*. Burla, Odisha, India: Departmen of CSE & IT.
- Patel, V, R. & Mehta, R. G., 2011. *Impact of Outlier Removal and normalization Approach in Modified K-means Clustering Algorithm*. *IJCSI International Journal of Computer Science*.
- Raharyani, M. P., 2017. Implementasi Algoritme Support Vector Regression pada Peramalan Jumlah pengunjung Pariwisata. S1. Universitas Brawijaya.
- Rangkuti, Freddy., 2005. *Analisis SWOT : Teknik Membedah Kasus Bisnis*. Jakarta: PT. Gramedia.
- Rifqi, M. R., 2018. Support Vector Regression Untuk Peramalan Permintaan Darah: Studi Kasus Unit Transfusi Darah Cabang – PMI Kota Malang. S1. Universitas Brawijaya.
- Surtiningsih, L., 2017. Peramalan Jumlah Kunjungan Wisatawan Mancanegara ke Bali Menggunakan Support Vector Regression dengan Algoritme Genetika. S1. Universitas Brawijaya.

Susanto, K., 2018. Normalisasi SVR dengan Ant Colony Optimization untuk peramalan tingkat produksi susu segar (Studi Kasus pada koperasi susu Sae Pujon, Malang). S1. Universitas Brawijaya.

Vijayakumar, S. Wu, Si, 1999. Sequential Support Vector Classifiers and Regression. Genoa: International Conference on Soft Computing.

