PREDIKSI VOLUME IMPOR BERAS NASIONAL MENGGUNAKAN METODE *SUPPORT VECTOR REGRESSION* (SVR)

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:
Cindy Inka Sari
NIM: 155150200111023

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

PENGESAHAN

PREDIKSI IMPOR BERAS NASIONAL MENGGUNAKAN METODE *SUPPORT VECTOR REGRESSION* (SVR)

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh :
Cindy Inka Sari
NIM: 155150200111023

Skripsi ini telah diuji dan dinyatakan lulus pada
22 April 2019
Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I	Dosen Pembimbing II
Budi Darma Setiawan, S.Kom, M.Cs	Ir.Sutrisno, M.T
NIP: 19841015 201404 1 002	NIP: 19570325 198701 1 001

Mengetahui
Ketua Jurusan Teknik Informatika

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

PERNYATAAN ORISINALITAS

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

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

Malang, 9 Mei 2019

Cindy Inka Sari

NIM: 155150200111023

PRAKATA

Puji syukur kehadirat Allah SWT karena atas berkah, rahmat dan kaunia-Nya penulis dapat menyelesaikan skripsi dengan judul "Prediksi Volume Impor Beras Nasional Menggunakan Metode *Suport Vector Regression* (SVR)". Skripsi yang ditulis merupakan tugas akhir dalam memenuhi persyaratan memperoleh gelar sarjana Komputer pada Fakultas Ilmu Komputer (FILKOM) Universitas Brawijaya. Selama pengerjaan skripsi ini penulis mendapatkan banyak bantuan, dukungan dan doa dari berbagai macam pihak. Pada keseempatan ini penulis bermaksud menyampaikan ucapan terimakasih kepada:

1. Bapak Budi Darma Setiawan, S.Kom, M.Sc selaku dosen pembimbing I yang telah memberikan bimbingan, pengarahan, dan saran kepada penulis selama masa pengerjaan skripsi.
2. Bapak Ir. Sutrisno, M.T selaku dosen pembimbing II yang telah memberikan bimbingan, dan ilmu kepada penulis selama masa pengerjaan skripsi.
3. Bapak Wayan Firdaus Mahmudy, S.Si., M.T., Ph.D. selaku Dekan Fakultas Ilmu Komputer Universitas Brawijaya.
4. Bapak Tri Astoto Kurniawan, S.T., M.T., Ph.D. selaku Ketua Jurusan Teknik Informatika, Fakultas Ilmu Komputer Universitas Brawijaya.
5. Bapak Agus Wahyu Widodo, S.T., M.Cs. selaku Ketua Program Studi Teknik Informatika, Fakultas Ilmu Komputer Universitas Brawijaya.
6. Bapak Achmad Arwan, S.Kom, M.Kom selaku dosen penasihat akademik yang telah memberikan saran dan nasihat selama menempuh masa studi.
7. Seluruh Dosen Fakultas Ilmu Komputer Universitas Brawijaya yang telah memberikan ilmu dan pengetahuannya kepada penulis selama masa studi hingga skripsi ini selesai.
8. Seluruh Civitas Akademika Fakultas Ilmu Komputer Universitas Brawijaya yang telah memberikan bantuan kepada penulis selama masa studi hingga skripsi ini selesai.
9. Ibu Sundiyah, ibu penulis yang senantiasa mendukung, memberikan motivasi dan doa serta pengorbanannya baik dari segi moril, materi kepada penulis sehingga penulis dapat menyelesaikan skripsi ini.
10. Rafi Raynaldo adik penulis yang memberikan dukungan dan selalu siap membantu penulis dalam menyelesaikan skripsi ini.
11. Mukhammad Shodiq Alwi yang senantiasa memberikan motivasi, nasehat, kritik dan saran serta selalu siap membantu dalam kondisi apapun.
12. Hetty Mukamillah dan Olivia Very Noorlinda teman kos yang selalu mendengarkan curahan hati, senantiasa memberikan saran dan motivasi kepada penulis.
13. Keluarga Alumni TIF-B 2015 khususnya Habridio Kurniawan Putra yang telah membantu, memberikan kritik, dan saran kepada penulis.
14. Keluarga LPM DISPLAY yang telah banyak membantu, menjadi tempat bertukar pikiran dan belajar dalam segala hal.

15. Teman-teman Teknik Informatika angkatan 2015 atas segala bantuan yang telah diberikan.
16. Serta semua pihak lainnya yang tidak dapat disebutkan satu persatu oleh penulis.

Penulis menyadari masih terdapat kekurangan dari skripsi ini, karena itu penulis mengharapkan kritik dan saran guna dapat menyempurnakan skripsi ini. Harapannya semoga skripsi ini dapat memberikan manfaat bagi pembaca, peneliti lainnya dan pihak lainnya.

Malang, 9 Mei 2019

Penulis

Email:

cindyinkas@gmail.com

ABSTRAK

Cindy Inka Sari, Prediksi Volume Impor Beras Nasional menggunakan Metode *Support Vector Regression* (SVR).

Pembimbing: Budi Darma Setiawan, S.Kom, M.Cs dan Ir. Sutrisno, M.T.

Pemenuhan kebutuhan beras sebagai bahan pangan utama di Indonesia selama ini mengandalkan produksi dari dalam negeri dan impor beras. Impor beras di Indonesia yang dilakukan secara terus-menerus seiring bertambahnya jumlah masyarakat membuat adanya ketergantungan impor beras dari negara lain. Prediksi dibutuhkan untuk mengontrol volume impor beras nasional karena volume impor yang tidak terkontrol akan menimbulkan kerugian dan dampak negatif. *Support Vector Regression* (SVR) dipilih dalam melakukan prediksi jumlah volume impor beras nasional karena lebih unggul dibandingkan beberapa metode lain. Data yang digunakan dalam prediksi adalah data konsumsi, produksi, volume impor beras 1 tahun sebelumnya sebagai variabel bebas dan data volume impor beras nasional dalam kurun waktu 1971 – 2016 sebagai variabel terikat. Pengujian dilakukan dengan menggunakan 9 data uji didapatkan parameter terbaik *Sigma (σ)* = 0.07, *Lambda (λ)* = 0.4, *Constanta Learning Rate* (cLr) = 0.01, Kompleksitas *(C)* = 10, *Epsilon (ε)* = 0.0004, jumlah Iterasi = 2000 dan jumlah data latih = 37. Hasil evaluasi diukur menggunakan nilai *Mean Absolute Percentage Error* (MAPE). Nilai MAPE terbaik yang dihasilkan termasuk dalam kategori cukup sebesar 32.2748.

Kata kunci: prediksi, regresi, impor beras nasional, *support vector regression*, MAPE.

ABSTRACT

Cindy Inka Sari, ***Prediction of National Rice Import using Support Vector Regression (SVR) Method.***

***Supervisors*: Budi Darma Setiawan, S.Kom, M.Cs dan Ir. Sutrisno, M.T.**

In Indonesia domestic rice production and rice imports are needed in order to attain national rice consumption. As the number of people increases and imported rice are consumed continuously, therefore Indonesia depends on rice imports from other countries*. Prediction is needed to control the volume of national rice imports because excessive imports will cause negative losses and impacts. Support Vector Regression (SVR) is used to predict the volume of national rice imports. The data used in the prediction are data on consumption, production, volume of rice imports 1 year earlier as bebas variables and data on the volume of national rice imports in the period 1971 - 2016 as terikat variables. Tests carried out using 9 test data obtained the best parameters Sigma (σ)* = 0.07, *Lambda (λ) =* 0.4, *Constanta Learning Rate* (cLr) = 0.01, Kompleksitas *(C)* = 10, *Epsilon (ε) = 0.0004, the number of Iterations = 2000 and the number of training data = 37. The evaluation results were measured using Mean Absolute Presentage Error (MAPE). The best MAPE value produced is included in the sufficient category of 32.2748.*

Keywords: *prediction, regression, national rice imports, support vector regression,* MAPE.

DAFTAR ISI

PENGESAHAN ii
PERNYATAAN ORISINALITAS iii
PRAKATA iv
ABSTRAK vi
ABSTRACT vii
DAFTAR ISI viii
DAFTAR TABEL xi
DAFTAR GAMBAR xii
DAFTAR KODE PROGRAM xiii
DAFTAR LAMPIRAN xiv
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 Prediksi 8
2.3 Impor 8
2.3.1 Impor Beras Nasional 9
2.4 *Support Vector Regression* (SVR) 9
2.4.1 Fungsi Kernel 11
2.5 Normalisasi dan Denormalisasi Data 11
2.6 *Mean Absolute Percentage Error* (MAPE) 12
BAB 3 METODOLOGI 13
3.1 Tipe Penelitian 13
3.2 Strategi Penelitian 13
3.3 Pengumpulan Data 13

3.4 Implementasi 13
3.5 Pengujian 14
3.6 Penarikan Kesimpulan 18
BAB 4 ALGORITME 19
4.1 Formulasi Permasalahan 19
4.2 Diagram Alir Algoritma Support Vector Regression 19
4.2.1 Normalisasi Min-Max 20
4.2.2 *Training* 21
4.2.3 *Testing* 29
4.2.4 Denormalisasi 34
4.2.5 Perhitungan MAPE 35
4.3 Manualisasi 35
4.3.1 Inisialisasi Parameter 36
4.3.2 Normalisasi Data 36
4.3.3 Training 37
4.3.4 *Testing* 40
4.3.5 Denormalisasi 42
4.3.6 Perhitungan MAPE 42
4.4 Perancangan User Interface 43
BAB 5 IMPLEMENTASI 45
5.1 Implementasi Membaca File 45
5.2 Implementasi Normalisasi Data 46
5.3 Implementasi Pelatihan 47
5.3.1 Implementasi Perhitungan Jarak Data Latih 47
5.3.2 Implementasi Perhitungan Matriks Hessian Data Latih 48
5.3.3 Implementasi *Sequential Learning* 49
5.3.4 Implementasi Pengujian 52
5.3.5 Implementasi Perhitungan Matriks Hessian Data Uji 53
5.3.6 Implementasi Perhitungan Denormalisasi 54
5.3.7 Implementasi Perhitungan MAPE 54
5.4 Implementasi Antarmuka 55
BAB 6 PENGUJIAN DAN ANALISIS 57

6.1 Pengujian Nilai Parameter SVR 57
6.1.1 Pengujian Nilai Parameter *Sigma* (σ) 57
6.1.2 Pengujian Nilai Parameter *Lambda (λ)* 58
6.1.3 Pengujian Nilai cLR (*Constanta Learning Rate*) 59
6.1.4 Pengujian Nilai Parameter Kompleksitas (*C*) 61
6.1.5 Pengujian Nilai Parameter *Epsilon (ε)* 62
6.2 Pengujian Jumlah Iterasi 64
6.3 Pengujian Variasi Jumlah Data Latih 65
6.4 Analisis Pembahasan 66
6.4.1 Analisis Korelasi Variabel Konsumsi dengan Variabel Volume Impor Beras 68
6.4.2 Analisis Korelasi Variabel Produksi dengan Variabel Volume Impor Beras 68
6.4.3 Analisis Korelasi Variabel Volume Impor 1 Tahun Sebelumnya dengan Variabel Volume Impor Beras 69
BAB 7 PENUTUP 70
7.1 Kesimpulan 70
7.2 Saran 70
DAFTAR REFERENSI 71
LAMPIRAN A DATA PENELITIAN 73
LAMPIRAN B PERBANDINGAN HASIL PREDIKSI DAN DATA AKTUAL IMPOR BERAS NASIONAL 75

DAFTAR TABEL

Tabel 2.1 Kajian Pustaka 6
Tabel 2.2 Kriteria Nilai MAPE 12
Tabel 3.1 Batasan Parameter Penelitian Sebelumnya 15
Tabel 3.2 Perancangan Pengujian Nilai *Parameter Sigma (σ)* 15
Tabel 3.3 Perancangan Pengujian Nilai Parameter *Lambda (λ)* 16
Tabel 3.4 Perancangan Pengujian Nilai Parameter *Constanta Learning Rate* (cLR) 16
Tabel 3.5 Perancangan Pengujian Nilai Parameter Kompleksitas (*C*) bagian 1 16
Tabel 3.6 Perancangan Pengujian Nilai Parameter Kompleksitas (*C*) bagian 2 17
Tabel 3.7 Parencangan Pengujian Nilai Parameter *Epsilon (ε)* 17
Tabel 3.8 Perancangan Pengujian Jumlah Iterasi 18
Tabel 3.9 Perancangan Pengujian Variasi Data Latih 18
Tabel 4.1 Data Latih 36
Tabel 4.2 Data Uji 36
Tabel 4.3 Inisialisasi Parameter 36
Tabel 4.4 Hasil Normalisasi Data Latih 37
Tabel 4.5 Hasil Normalisasi Data Uji 37
Tabel 4.6 Hasil Perhitungan Jarak Data Latih Bagian 1 37
Tabel 4.7 Hasil Perhitungan Jarak Data Latih Bagian 2 38
Tabel 4.8 Hasil Perhitungan Matriks Hessian Data Latih 38
Tabel 4.9 Hasil Perhitungan *Sequential Learning* Iterasi 1 39
Tabel 4.10 Hasil Perhitungan *Sequential Learning* iterasi 15 39
Tabel 4.11 Nilai *f(x)* data latih 40
Tabel 4.12 Hasil Perhitungan Jarak data latih dengan data uji 41
Tabel 4.13 Hasil Perhitungan Matriks Hessian Data Uji 41
Tabel 4.14 Hasil Perhitungan nilai *f(x)* data uji 41
Tabel 4.15 Hasil Perhitungan Denormalisasi Dan Data Aktual Data Latih Dan Data Uji 42
Tabel 4.16 Hasil Perhitungan Nilai MAPE Data Latih dan Data Uji 42
Tabel 6.1 Hasil Pengujian Rentang Nilai Parameter *Sigma (σ)* 57
Tabel 6.2 Hasil Pengujian Nilai Parameter *Lambda (λ)* Bagian 1 58
Tabel 6.3 Hasil Pengujian Nilai Parameter *Lambda (λ)* Bagian 2 59
Tabel 6.4 Hasil Pengujian Nilai Parameter cLR 60
Tabel 6.5 Hasil Pengujian Nilai Parameter Kompleksitas (*C*) bagian 1 61
Tabel 6.6 Hasil Pengujian Nilai Parameter Kompleksitas (*C*) bagian 2 62
Tabel 6.7 Hasil Pengujian Nilai Parameter *Epsilon (ε)* 63
Tabel 6.8 Hasil Pengujian Jumlah Iterasi 64
Tabel 6.9 Hasil Pengujian Variasi Jumlah Data Latih 65
Tabel 6.10 Perbandingan Hasil Prediksi dengan Data Aktual 66
Tabel 6.11 Perbandingan Hasil Prediksi dengan Data Aktual bagian 2 67
Tabel 6.12 Intepretasi terhadap Koefisien Korelasi (r) 67

DAFTAR GAMBAR

Gambar 3.1 Implementasi Sistem 14
Gambar 4.1 Diagram Alir Algoritma *Support Vector Regression* (SVR) 19
Gambar 4.2 Diagram alir Normalisasi *Min-Max* 20
Gambar 4.3 Diagram Alir Proses *Training* 21
Gambar 4.4 Diagram Alir Perhitungan Jarak Latih 22
Gambar 4.5 Diagram Alir Perhitungan Matriks Hessian Data Latih 24
Gambar 4.6 Diagram Alir *Sequential Learning* 25
Gambar 4.7 Diagram Alir Perhitungan Nilai Error 26
Gambar 4.8 Diagram Alir Proses Perhitungan Nilai $\delta\alpha_i^*$ dan $\delta\alpha_i$ 27
Gambar 4.9 Diagram Alir Proses Perhitungan nilai *Lagrange Multiplier* 28
Gambar 4.10 Diagram Alir Perhitungan Nilai f(x) Data Latih 29
Gambar 4.11 Diagram Alir Proses *Testing* 30
Gambar 4.12 Diagram Alir Proses Perhitungan Jarak Data Uji dan Data Latih 31
Gambar 4.13 Diagram Alir Proses Perhitungan Matriks Hessian Data Uji 32
Gambar 4.14 Diagram Alir Proses Perhitungan f(x) Data Uji 33
Gambar 4.15 Diagram Alir Denormalisasi 34
Gambar 4.16 Diagram alir perhitungan Nilai MAPE 35
Gambar 4.17 Perancangan *User interface* 43
Gambar 5.1 Implementasi Antarmuka 55
Gambar 6.1 Grafik Hasil Pengujian Nilai Parameter Sigma 58
Gambar 6.2 Grafik Hasil Pengujian Nilai Parameter Lambda 59
Gambar 6.3 Grafik Hasil Pengujian Nilai Parameter cLR 60
Gambar 6.4 Grafik Hasil Pengujian Nilai Parameter Kompleksitas (C) 62
Gambar 6.5 Grafik Hasil Pengujian Rentang Nilai Epsilon 63
Gambar 6.6 Grafik Hasil Pengujian Jumlah Iterasi 64
Gambar 6.7 Grafik Hasil Pengujian Variasi Data Latih 66
Gambar 6.8 Hasil Analisis Korelasi Variabel Konsumsi dengan Variabel Volume Impor Beras 68
Gambar 6.9 Hasil Analisis Korelasi Variabel Produksi dengan Variabel Volume Impor Beras 68
Gambar 6.11 Hasil Analisis Korelasi Variabel Volume Impor 1 Tahun Sebelumnya dengan Variabel Volume Impor Beras 69

DAFTAR KODE PROGRAM

Kode Program 5.1 Membaca File 46
Kode Program 5.2 Normalisasi Data 46
Kode Program 5.3 Pencarian Nilai Minimum 47
Kode Program 5.4 Pencarian Nilai Maksimum 47
Kode Program 5.5 Formula Menghitung Jarak Data Latih 48
Kode Program 5.6 Matriks Jarak Data Latih 48
Kode Program 5.7 Perhitungan Matriks Hessian Data Latih 49
Kode Program 5.8 *Sequential Learning* 50
Kode Program 5.9 Perhitungan Nilai Error 50
Kode Program 5.10 Perhitungan Nilai $\delta\alpha^*$ dan $\delta\alpha$ 51
Kode Program 5.11 Perhitungan Nilai *Lagrange Multiplier* 51
Kode Program 5.12 Perhitungan Nilai f(x) Data Latih 52
Kode Program 5.13 Formula Menghitung Jarak Data Uji dan Data Latih 52
Kode Program 5.14 Matriks Jarak Data Uji 53
Kode Program 5.15 Perhitungan Matriks Hessian Data Uji 53
Kode Program 5.16 Perhitungan Denormalisasi Data Latih 54
Kode Program 5.17 Perhitungan Denormalisasi Data Uji 54
Kode Program 5.18 Perhitungan MAPE Data Latih 54
Kode Program 5.19 Perhitungan MAPE Data Uji 55

DAFTAR LAMPIRAN

LAMPIRAN A DATA PENELITIAN 73

LAMPIRAN B PERBANDINGAN HASIL PREDIKSI DAN DATA AKTUAL IMPOR BERAS NASIONAL..... 75

BAB 1 PENDAHULUAN

Bab ini menjelaskan latar belakang penelitian, rumusan masalah yang akan dibahas dalam penelitian, tujuan dan manfaat yang ingin dicapai penulis, Batasan masalah yang ditentukan serta sistematika pembahasan

1.1 Latar Belakang

Beras merupakan kebutuhan pangan utama masyarakat Indonesia. Dalam pemenuhan kebutuhan beras nasional, tidak hanya mengandalkan produksi beras dalam negeri namun juga mengandalkan produksi beras dari luar negeri dengan kata lain impor beras. Pemenuhan kebutuhan beras nasional Indonesia kenyataannya masih bergantung pada impor beras dari negara lain (Christianto, 2013). Hal-hal yang mendorong dilakukannya impor beras yaitu adanya peningkatan permintaan masyarakat akan beras seiring bertambahnya penduduk yang ada di Indonesia, perubahan iklim secara ekstrem yang terjadi karena bergesernya musim hujan dan musim kemarau yang menyebabkan penurunan produksi beras (Khotimah, 2018), dan berkurangnya lahan-lahan pertanian di Indonesia karena dijadikan lahan untuk non-pertanian seperti kawasan untuk industrI, perdagangan dan perumahan juga sarana publik (Febrianty, 2013).

Impor beras nasional dipengaruhi oleh beberapa faktor yaitu jumlah produksi dan jumlah konsumsi beras. Jumlah produksi yang dihasilkan berpengaruh secara signifikan dalam jangka waktu yang panjang maupun pendek terhadap impor beras (Kurniyawan, 2013). Sedangkan jumlah konsumsi beras berpengaruh positif dan signifikan dalam jangka panjang maupun pendek terhadap impor beras (Christianto, 2013). Hal ini terjadi karena ketika konsumsi masyarakat tinggi dan produksi yang dihasilkan belum mampu memenuhi kebutuhan, maka impor beras akan dilakukan untuk memenuhi kekurangannya.

Prediksi dibutuhkan untuk mengontrol volume impor beras nasional karena volume impor yang terlalu banyak akan menimbulkan kerugian dan dampak negatif. Salah satu dampak negatif yang dapat dirasakan adalah surplus beras yang berlebihan. Surplus beras yang berlebihan dapat menimbulkan harga turun di pasaran dan inflasi. Sebaliknya jika jumlah impor beras yang dilakukan kurang sedangkan konsumsi masyarakat tengah tinggi, dapat menyebabkan kelangkaan beras, harga beras melambung karena stok yang menipis serta menimbulkan pemborosan pengeluaran negara untuk impor beras padahal pengeluaran negara dapat dialokasikan ke sektor lain yang lebih penting (Christianto, 2013).

Beberapa metode yang dapat digunakan untuk prediksi model regresi antara lain Support Vector Regression (SVR), Regresi Linier Berganda (RLB) dan ARIMA. Dibandingkan dengan metode RLB, SVR lebih unggul dalam pemanfatan data nonlinier melalui fungsi kernel. Fungsi kernel dapat digunakan untuk memetakan vektor input ke ruang fitur dengan dimensi yang tinggi. RLB mampu mengatasi hal tersebut dengan model regresi namun fungsi transformasi untuk data seringkali tidak ditemukan. ARIMA yang digunakan untuk prediksi tidak bisa digunakan

untuk memodelkan data tipe nonlinier karena belum mampu mengakomodasi data outlier. SVR dapat digunakan untuk data nonlinier dengan fungsi kernelnya (Raharyani, Putri and Setiawan, 2018). SVR lebih unggul dalam mengatasi masalah *overfiting* dibandingkan dengan model regresi biasa dan jaringan syaraf tiruan (Maharesi, 2013).

SVR telah banyak digunakan dalam melakukan prediksi. Hasil yang didapatkan mempunyai akurasi yang baik. Salah satunya adalah penelitian yang dilakukan oleh Mustakim, Buono dan Hermadi ditahun 2015 menggunakan SVR untuk memprediksi produktivitas kelapa sawit di Riau. Dalam penelitiannya menggunakan 3 kernel yaitu kernel linier, kernel polynomial dan kernel RBF (*Radial Basis Function*). Hasilnya kernel RBF menghasilkan hasil yang terbaik dengan MSE sebesar 6% dengan koefisien determinasi sebesar 95% (Mustakim, Buono & Hermadi, 2015).

Penelitian sebelumnya tentang prediksi volume beras nasional telah dilakukan oleh Nendiana ditahun 2017. Dimana prediksi yang dilakukan meggunakan metode *multifactor high order fuzzy time series* dengan *antecedent factor* yaitu produksi dan konsumsi beras. Penelitian yang dilakukan menghasilkan evaluasi yang baik dengan nilai NRMSE sebesar 0,298 (Putri, Santoso & Adinugroho, 2017).

Berdasarkan uraian latar belakang di atas peneliti mengangkat judul Prediksi Volume Impor Beras Nasional menggunakan *Support Vector Regression* (SVR). Penelitian ini mengimplementasikan metode SVR dengan kernel *Radial Basis Function* (RBF) yang ditujukan untuk mengetahui pengaruh nilai parameter terhadap hasil prediksi dan mengetahui hasil evaluasi metode yang diukur dengan MAPE (*Mean Absolute Percentage Error*).

1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah ditulis, maka didapatkan rumusan masalah:

1. Bagaimana pengaruh nilai parameter *Support Vector Regression* (SVR) yang digunakan dalam menentukan hasil prediksi impor beras nasional?
2. Bagaimana hasil evaluasi yang diperoleh dari sistem dalam penerapan metode *Support Vector Regression* (SVR) untuk prediksi volume impor beras nasional yang diukur dengan MAPE (*Mean Absolute Percentage Error*)?

1.3 Tujuan

Tujuan yang ingin dicapai dalam penelitian ini adalah :

1. Mengetahui pengaruh nilai parameter *Support Vector Regression* (SVR) yang digunakan dalam menentukan hasil prediksi impor beras nasional
2. Mengetahui hasil evaluasi dari sistem prediksi volume impor beras nasional menggunakan metode *Support Vector Regression* (SVR) yang diukur dengan MAPE (*Mean Absolute Percentage Error*)

1.4 Manfaat

Manfaat yang dapat diperoleh dari penelitian ini adalah:

1. Sebagai bahan kajian ulang dan masukan untuk pemerintah dalam merumuskan kebijakan dan pengambilan keputusan mengenai impor beras nasional.

1.5 Batasan Masalah

Batasan masalah dari penelitian ini adalah variable bebas yang digunakan berjumlah 3 yaitu produksi dan konsumsi beras dalam negeri dan impor tahun sebelumnya. Variable terikat yag digunakan adalah jumlah impor beras dalam negeri.

1.6 Sistematika Pembahasan

Sistematika pembahasan ini merupakan gambaran dan uraian dari penelitian secara garis besar yang terdiri dari beberapa yaitu:

BAB 1 PENDAHULUAN

Bab ini menjelaskan tentang latar belakang masalah dari penelitian, rumusan masalah ynag akan dibahas, tujuan dan manfaat yang ingin dicapai, Batasan masalah yang telah ditentukan dan sistematika pembahasan tentang prediksi volume impor beras nasional menggunakan metode *Support Vector Regression* (SVR).

BAB 2 LANDASAN KEPUSTAKAAN

Bab ini menjelaskan tentang kajian pustaka yang digunakan karena terkait dengan penelitian ynag telah dilakukan sebelumnya dan dasar teori yang berkaitan dengan penelitian ini.

BAB 3 METODOLOGI

Bab ini menjelaskan langkah langkah yang dilakukan dalam melakukan penelitian meliputi studi literatur, pengumpulan data, analisis kebutuhan, perancangan sistem, implementasi sistem, pengujian, analisis dan kesimpulan

BAB 4 ALGORITME

Bab ini berisi penjelasan gambaran dari sistem secara umum dan cara kerja sistem meliputi perancangan proses, manualisasi, perancangan antarmuka dan perancangan pengujian.

BAB 5 IMPLEMENTASI

Bab ini menjelaskan hasil implementasi system berupa kode program dan antarmuka yang telah diimplementasikan sesuai dengan perancangan yang ada.

BAB 6 PENGUJIAN DAN ANALISIS

Bab ini menjelaskan tentang pengujian sistem hasil dari implementasi yang telah dilakukan kemudian hasil pengujian akan dianalisis.

BAB 7 PENUTUP

Bab ini menjelaskan kesimpulan dari implementasi, pengujian sistem yang dilakukan dan saran agar dalam penelitian selanjutnya dapat dilakukan penelitian serupa yang lebih baik.

BAB 2 LANDASAN KEPUSTAKAAN

2.1 Kajian Pustaka

Dalam melakukan penelitian, penulis menggunakan beberapa kajian pustaka dari penelitian yang telah dilakukann sebelumnya. 4 kajian pustaka tentang metode SVR dan 1 kajian pustaka menggunakan objek yang sama.

Kajian pertama adalah penelitian yang ditulis oleh Nendiana Putri yang memprediksi volume impor beras nasional dengan metode *multi factors high order fuzzy time series*. Dalam penelitiannya diguanakan volume beras sebagai faktor utama dan faktor produksi dan konsumsi beras dijadikan sebagai *antecedent factor*. Hasil evaluasi menggunakan nilai *normalized root mean squared error* (NRMSE) sebesar 0,298 dimana nilai yang mendekati 0 berarti prediksi yang dilakukan mendapatkan hasil yang baik (Putri, Santoso & Adinugroho, 2017).

Kajian kedua adalah penelitian di tahun 2017 yang dilakukan oleh Rezzy eko Caraka dkk dengan judul Peramalan *Crude Palm Oil* (CPO) Menggunakan *Support Vector Regression* Kernel Radian Basis. Dalam penelitian ini menggunakan fungsi kernel yang digunakan adalah radial basis. Hasil evaluasi menghasilkan nilai dan akurasi yang baik dengan nilai R^2 sebesar 98,71738% untuk training dan 83,45659% untuk data testing sedangkan nilai MAPE yang dihasilkan sebesar 0.9191906 dan 0.8761705 untuk data training dan testing (Caraka, Yasin & Basyiruddin, 2017).

Kajian ketiga adalah penelitian dengan *Support Vector Regression* yang digunakan untuk memprediksi produktifitas kelapa sawit di propinsi Riau yang ditulis oleh Mustakim dkk di tahun 2015. Data yang digunakan adalah data produksi kelapa sawit 8 tahun terakhir. Pada penelitian ini dilakukan uji coba terhadap kernel linier, polynomial dan kernel RBF. Hasilnya, kernel polynomial dan linier tidak direkomendasikan oleh penulis karena prediksi yang dihasilkan menjahui data aktual dari observasi yang dilakukan. Kernel RBF menghasilkan prediksi yang terbaik diantara ke 3 kernel lainnya. Model terbaik SVR pada kernel RBF yang didapat dalam penelitian yaitu nilai MSE sebesar 6% koefisien determinasi (R^2) sebesar 95% (Mustakim, Buono & Hermadi, 2015).

Kajian keempat adalah penelian dengan judul Penggunaan *Support Vector Regression* (SVR) pada prediksi return Saham Syariah BEI. Yang ditulis oleh Retno Maharesi ditahun 2013. Data yang digunakan adalah nilai proporsi rerata tahunan harga saham pada 2 tahun yang digunakan sebagai variable terikat, sedangkan data *Book Value* (BV), tingkat pembelian ekuitas dan proporsi deviden yang dibayarkan ke investor public menjadi data untuk variable bebas. Penelitian ini membandingkan hasil dari SVR dengan Regresi Linier Berganda dengan pengukuran MSE di kedua metode. Hasilnya metode SVR menghasilkan model prediksi yang lebih baik dibandingkan Regresi Linier Berganda. Penelitiaini juga membandingkan kernel RBF, polinomial dan linier. SVR dengan kernel RBF menghasilkan hasil yang terbaik dengan MSE sebesar 0.0916 (Maharesi, 2013).

Kajian kelima adalah penelitian yang dilakukan oleh yousefi et al ditahun 2015 dengan judul *Support vector regression methodology for prediction of output energy in rice production.* Penelitian ini dilakukan perbandingan antara adaptive neuro-fuzzy inference system (ANFIS) dan SVR dengan fungsi kernel polynomial dan RBF. Hasil yang didapatkan SVR dengan kernel RBF mempunyai hasil yang lebih baik dibanding yang lainnya dengan koefisien determinasi sebesar 98 untuk data training dan 94 untuk data testing. Sedangkan evaluasinya menggunakan nilai RMSE dengan hasil 2.3 untuk data training dan 8 untuk data testing (Yousefi *et al.*, 2015).

Tabel 2.1 Kajian Pustaka

No.	Judul	Topik	Metode	Hasil
1.	Prediksi Volume Impor Beras Nasional dengan Metode *Multi-Factors High-Order Fuzzy Time Series*	Prediksi volume impor beras nasional dengan pertimbangan faktor produksi dan konsumsi beras	*multi-factors high-order time series* dan *fuzzy c-means* untuk pengelompokan	Nilai *Normalized Root Mean Squared Error* (NRMSE) minimum yang didapat sebesar 0,298 yang berarti prediksi dapat dilakukan dengan baik
2.	Peramalan *Crude Palm Oil* (CPO) Menggunakan *Support Vector Regression* Kernel Radial Basis	Peramalan harga minyak mentah kelapa sawit / *crude palm oil*	*Support Vector Regression* dengan fungsi kernel radian basis	Dihasilkan nilai yang menunjukan validitas dan akurasi yang baik dengan nilai R^2 untuk data training dan testing sebesar 98,71738% dan 83,45659% dengan MAPE sebesar 0.91
3.	*Support Vector Regression* Untuk Prediksi Produktivitas Kelapa Sawit	Prediksi Produktivitas Kelapa Sawit Di Provinsi Riau	*Support Vector Regression* dengan fungsi kernel linier, polynomial dan kernel RBF	Kernel RBF menghasilkan prediksi yang terbaik diantara ke 3 kernel lainnya. Model terbaik SVR pada kernel RBF

	Di Provinsi Riau			yaitu nilai MSE 6% koefisien determinasi 95%
4.	Penggunaan *Support Vector Regression* (Svr) Pada Prediksi Return Saham Syariah Bei	Prediksi Return Saham Syariah Bei dengan 1 variabel terikat dan 3 variabel bebas	*Support Vector Regression* dengan kernel linier, polynomial dan RBF dibandingkan dengan Regresi Linier Berganda	metode SVR menghasilkan model prediksi yang lebih baik dibandingkan Regresi Linier Berganda. SVR dengan kernel RBF menghasilkan hasil yang terbaik dengan MSE sebesar 0.0916
5.	*Support vector regression methodology for prediction of output energy in rice production*	Prediksi output energi produksi padi	*Support Vector Regresssion* dengan fungsi kernel polynomial dan RBF, ANFIS *(adaptive neuro-fuzzy inference system)*	Hasil terbaik adalah SVR dengan kernel RBF. Koefisien determinasi sebesar 98 untuk data training dan 94 untuk data testing. RMSE dengan hasil 2.3 untuk data training dan 8 untuk data testing

Berdasarkan uraian yang telah dipaparkan di atas perbedaan penelitian ini dengan penelitian sebelumnya terletak pada penggunaan variabel. Pana penelitian ini penulis menggunakan 3 variabel bebas yaitu produksi, konsumsi dan impor beras 1 tahun sebelumnya serta variabel impor beras sebagai variabel terikatnya. Adapun metode yang digunakan dalam penelitian berdasarkan pemaparan di atas dipilih metode SVR dengan kernel RBF sebagai solusi dalam melakukan prediksi guna menyelesaikan permasalahan terkait impor beras nasional.

2.2 Prediksi

Menurut KBBI (Kamus Besar Bahasa Indonesia), prediksi berarti ramalan atau prakiraan. Salah satu upaya dengan menggunakan data masa lalu untuk diestimasi nilainya dimasa depan disebut prediksi/peramalan. Tujuan adanya prediksi ini adalah untuk memperkirakan kejadian ataupun peristiwa di masa depan (Yuniastari & Wirawan, 2016). Pengertian lain dari prediksi adalah suatu ilmu maupun seni digunakan untuk meramalkan suatu kejadian dengan menggunakan data masa lalu yang dikemas dalam model matematis (Kusuma, 2015). Berdasarkan beberapa paparan diatas dapat dikatakan bahwa prediksi merupakan sebuah langkah atau upaya yang merupakan bagian dari seni maupun ilmu untuk memperkirakan masa depan dengan data dan informasi dari masa lalu yang dapat dikemas dalam model matematis.

Prediksi dapat dibedakan menjadi berdasarkan jangka waktu dan berdasarkan metode. Berdasarkan jangka waktu adalah prediksi dalam jangka panjang, dalam jangka pendek dan dalam jangka menengah. Sedangkan prediksi berdasarkan metode dapat dibedakan menjadi metode kuantitatif dan metode kualitatif. Prediksi kuantitatif dilakukan dengan menggunakan metode matematis dan statistic karena menggunakan angka. Sedangka prediksi kualitatif melibatkan pendapat para ahli di bidangnya (Dewi & Himawati, 2015).

2.3 Impor

Menurut KBBI, impor dapat diartikan dengan proses memasukkan barang dari luar negeri ke dalam negeri. Dalam laporan akhir kajian penyusunan target ekspor impor Indonesia 2015-2019 yang ditulis oleh pusat kebijakan perdagangan luar negeri, Badan Pengkajian dan Pengembangan Kebijakan Perdagangan Kementrian Perdagangan dijelaskan impor merupakan suatu kegiatan pembelian barang yang dilakukan oleh suatu negara kepada negara lain penghasil produk yang akan dibeli. Impor terjadi jika terdapat kelebihan dalam permintaan internasional. Permintaan impor terhadap suatu barang dapat dilakukan bila permintaan atau kebutuhan suatu barang melebihi produksinya. Kebijakan impor merupakan salah satu instrument strategis pemerintah guna menjaga kepentingan ekonomi maupun kepentingan social secara lebih luas.

Dalam Kajian yang dilakukan oleh Badan Pengkajian dan Pengembangan Perdagangan Kementerian Perdagangan Indonesia ditahun 2016 tentang Peran Kebijakan Impor Dalam Rangka Mendukung Industri Manufaktur, dijelaskan bahwa impor disuatu negara di pengaruhi oleh beberapa faktor diantaranya:

1. Barang barang yang terdapat di dalam negeri tidak dapat dipenuhi oleh produsen domestik atau jumlahnya terbatas sedangkan permintaan kebutuhan domestik tinggi.
2. Produk impor mempunyai harga yang lebih murah daripada produk domestik. Sehingga faktor harga dan keseimbangan harga nasional maupun internasional sangat menentukan permintaan impor.

3. Impor lebih menguntungkan karena proses produksi yang dilakukan dalam negeri ditujukan untuk ekspor dengan harga ekspor yang lebih tinggi sehingga dapat mengganti biaya yang dikeluarkan untuk impor
4. Nilai impor bergantung pada nilai tingkat pendapatan nasional. Bila pendapatan nasional semakin tinggi, semakin rendah menghasilkan barang maka impor pun semakin tinggi.

Selain faktor-faktor yang telah disebutkan, masih banyak faktor yang mempengaruhi impor di suatu negara.

2.3.1 Impor Beras Nasional

Impor beras nasional merupakan salah satu kebijakan pemerintah untuk mendatangkan beras dari luar negeri karena kurangnya jumlah beras yang ada untuk memenuhi permintaan dan konsumsi masyarakat. Impor beras di Indonesia telah diatur dalam Peraturan Menteri Perdagangan R.I Nomor 1 tahun 2018 tentang ketentuan ekspor dan impor beras. Impor beras nasional hanya dapat dilakukan oleh Perusahan Umum BULOG dan mendapat persetujuan resmi dari menteri sebagaimana diatur dalam pasal 16 dan 17 Peraturan Menteri Perdagangan R.I Nomor 1 tahun 2018.

Sebuah negara mempunyai alasan untuk melakukan impor karena adanya kegagalan pemenuhan kebutuhan dalam negeri. Kebutuhan beras yang tidak mampu disediakan oleh suatu negara dapat terjadi akibat tidak efisiennya produksi beras yang ada di negara tersebut (Ricart, 2016). Produksi beras yang dihasilkan oleh Indonesia mempengaruhi impor beras yang dilakukan oleh Indonesia. Jumlah impor yang lebih besar dari produksi beras nasional menimbulkan ketergantungan impor karena jumlah penduduk yang selalu meningkat mengimbangi peningkatan konsumsi masyarakat akan beras (Khotimah, 2018).

2.4 *Support Vector Regression* (SVR)

SVR merupakan salah satu metode untuk menyelesaikan masalah regresi dimana metode ini merupakan pengembangan dari Algoritme *Support Machine* (SVM). SVM merupakan salah satu algoritma klasifikasi dengan mencari hyperplane terbaik. Hyperplane dalam SVM berguna untuk memisahkan 2 kelas data dalam klasifikasi. Ditahun 1999 Vijayakumar dan Wu mengenalkan algoritma sekuensial untuk SVR. Dibandingkan dengan SVR konvensional, algoritma sekuensial dari SVR ini memakan waktu komputasi yang cepat dan solusi yang lebih optimal (Muhamad, et al., 2017). Algoritma Sekuensial SVR (Vijayakumar & Wu, 1999):

1) Inisialisasi parameter, a_i = 0, $a_i^* = 0$

   Hitung $[R_{ij}] = K(x_i, x_j) + \lambda^2$, untuk $i, j = 1, \dots, l$ (2.1)

   Dengan

   a_i^* , a_i = Lagrange Multipliers

Rij = Matriks Hessian (i = baris, j = kolom)

$K(x, x_i)$ = Fungsi kernel

λ = variabel scalar

l = jumlah data

2) Proses pelatihan (sequential learning)

a) Perhitungan nilai error dengan persamaan:

$E_i = y_i - \sum_{j=1}^{l}(a_j^* - a_j)R_{ij}$, untuk *i*, *j* = 1, ..., l (2.2)

Dengan

E_i = nilai error ke i

y_i = Nilai aktual

a_i^* , a_i = Lagrange Multipliers

R_{ij} = Matriks hessian

l = jumlah data

b) Menghitung $\delta\alpha_i{}^*$ dan $\delta\alpha_i$ dengan persamaan:

$\delta a_i^* = \min\{\max[\gamma(E_i - \varepsilon), -a_i^*], C - a_i^*\}$ (2.3)

$\delta a_i = \min\{\max[\gamma(-E_i - \varepsilon), -a_i], C - a_i\}$ (2.4)

dengan,

δa_i^*, δa_i = perubahan nilai a_i^* dan a_i

γ = Learning Rate

E_i = nilai error ke-i

ε = nilai kerugian

C = kompleksitas

a_i^* , a_i = Lagrange Multipliers

c) Perhitungan Lagrange Multipliers dengan persamaan:

$a_i^* = a_i^* + \delta a_i^*$ (2.5)

$a_i = a_i + \delta a_i$ (2.6)

Dengan

a_i^* , a_i = Lagrange Multipliers

δa_i^*, δa_i = perubahan nilai a_i^* dan a_i

3) Ulangi langkah pelatihan pada point ke 2 hingga iterasi maksimum atau telah mencapai konvergensi dengan kriteria berhenti:

Max ($|\delta a_i^*|$) < ε dan max ($|\delta a_i|$) < ε

4) Support vector didapat ketika (a_i^* - a_i) ≠ 0

5) Hasil dari regresi dapat dihitung dengan fungsi regresi:

$$f(x) = \sum_{i=1}^{l}(a_i^* - a_i)(K(x, x_i) + \lambda^2), \text{ untuk } i, j = 1, \dots, l \quad (2.7)$$

Dimana:

$f(x)$ = Hasil Regresi

a_i^*, a_i = Lagrange Multipliers

$K(x, x_i)$ = Fungsi kernel

λ = variabel scalar

l = jumlah data

2.4.1 Fungsi Kernel

Fungsi kernel merupakan fungsi yang digunakan untuk memetakan ke ruang fitur dimensi yang lebih tinggi data input nonlinier yang digunakan untuk meningkatkan kemampuan sistem melalui proses pembelajaran. Fungsi kernel memungkinkan *dot product* dilakukan dalam ruang fitur dimensi tinggi menggunakan input data ruang dimensi rendah (Wu, Ho & Lee, 2004). SVR mempunyai beberapa fungsi kernel nonlinier salah satunya adalah fungsi kernel *Radial Basis Function* (RBF). Fungsi kernel RBF dapat dihitung dengan persamaan:

$$K(x, x_i) = \exp\left(\frac{-||x-y||^2}{2\sigma^2}\right) \quad (2.8)$$

Dengan:

$K(x, x_i)$ = Fungsi kernel

x, y = titik data

σ = standar deviasi (sigma)

2.5 Normalisasi dan Denormalisasi Data

Normalisasi data dilakukan agar data berada pada kisaran standar pada jarak tertentu. Tujuannya untuk mendapatkan bobot yang sama dari semua data yang ada. Dengan normalisasi bobot dari masing masing data akan sama atau tidak ada yang lebih dominan. Nilai rentang yang bervariasi harus diskalakan ke dalam batas tertentu agar dimensi data rata tidak ada yang terlalu besar dan terlalu kecil. Karena data yang punya dimensi berbeda akan berpengaruh terhadap hasil (Mustakim, Buono & Hermadi, 2015).

Teknik normalisasi yang digunakan dalam penelitian ini adalah normalisasi Min-Max. Teknik normalisasi data pada metode normalisasi Min-Max menggunakan transformasi linier. Normalisasi ini memetakan nilai A ke dalam kisaran (0,1). MinA adalah nilai minimum untuk atribut A. MaxA adalah nilai maximum untuk atribut A (Patel & Mehta, 2011). Persamaan normalisasi Min-Max dapat dituliskan sebagai berikut:

$$v' = \frac{v - MinA}{MaxA - MinA} \quad (2.10)$$

Data yang dinormalisasi akan dikembalikan menjadi data aktual dengan denormalisasi. Persamaan untuk menghitung denormalisasi dapat dituliskan sebagai berikut:

$$denormalisasi = \big(\mathrm{f(x)} * (\max - \min)\big) + \min \quad (2.11)$$

f(x) merupakan nilai regresi yang akan didapatkan di akhir perhitungan dengan metode SVR. Max merupakan nilai max dari data yang digunakan dan min adalah nilai minimal data yang digunakan

2.6 *Mean Absolute Percentage Error* (MAPE)

MAPE merupakan ukuran untuk kesalahan relative. MAPE menyatakan presentase kesalahan hasil dari peramalan (Yuniastari & Wirawan, 2016). Digunakan untuk menghitung selisih data dengan data prediksi dalam bentuk presentase yang kemudian dicari nilai tengahnya. Persamaan untuk menghitung MAPE seperti berikut:

$$\text{MAPE} = \left(\frac{100\%}{n}\right) \sum_{t}^{n} \frac{|Xt - Ft|}{Xt} \quad (2.12)$$

Dengan:

Xt = nilai data aktual

Ft = nilai prediksi

n = jumlah data

Kriterian nilai MAPE ditunjukan pada tabel berikut (Chang, Wang & Liu, 2007):

Tabel 2.2 Kriteria Nilai MAPE

No	Nilai MAPE	Jenis Kriteria
1	<10%	Sangat Baik
2	10%-20%	Baik
3	20%-50%	Cukup
4	>50%	Buruk

BAB 3 METODOLOGI

Pada bab ini akan dijelaskan metodelogi dalam melakukan penelitian. Metodelogi penelitian yang dijelaskan dalam bab ini meliputi tipe penelitian yang digunakan, strategi penelitian, pengumpulan data, langkah dalam melakukan implementasi, pengujian dan penarikan kesimpulan.

3.1 Tipe Penelitian

Penelitian yang dilakukan merupakan penelitian yang bersifat non-implementatif karena fokus terhadap studi hubungan fenomena yang sedang diteliti yang dilakukan untuk menghasilkan sebuah analisis ilmiah. Adapun pendekatan penelitian yang dilakukan adalah analitik yang bertujuan untuk menjelaskan hubungan atau relasi antara komponen dalam penelitian yang dilakukan. Keluaran yang dihasilkan adalah hasil analisis.

3.2 Strategi Penelitian

Strategi penelitian dilakukan dengan studi pustaka, pengumpulan data, implementasi, pengujian dan analisis serta penarikan kesimpulan. Studi pustaka dilakukan dalam rangka pemahaman dari teori teori yang berkaitan dengan penelitian dimana studi pustaka akan menjadi pendukung dalam melakukan penelitian baik perancangan maupun implementasi sistem. Teori yang digunakan berasal dari buku, jurnal, beberapa dokumentasi yang ada di internet dan lainnya. Pengumpulan data dilakukan untuk mencari data sesuai dengan kebutuhan penelitian dimana data yang digunakan adalah data produksi, konsumsi, volume impor beras 1 tahun sebelumnya dan volume impor beras nasional. Implementasi yang dilakukan menggunakan meetode SVR yang dibuat berdasarkan perancangan sistem. Pengujian dan analisis dilakukan setelah implementasi dilakukan dimana dalam penelitian ini pengujian dan analisis yang digunakan adalah pengujian nilai parameter SVR, pengujian iterasi, variasi data data latih, dan analisis pembahasan secara keseluruhan. Selanjutnya hasil yang didapat dan dianalisis digunakan untuk menjawab rumusan masalah yang telah ditulis dan dituangkan dalam kesimpulan akhir dari penelitian.

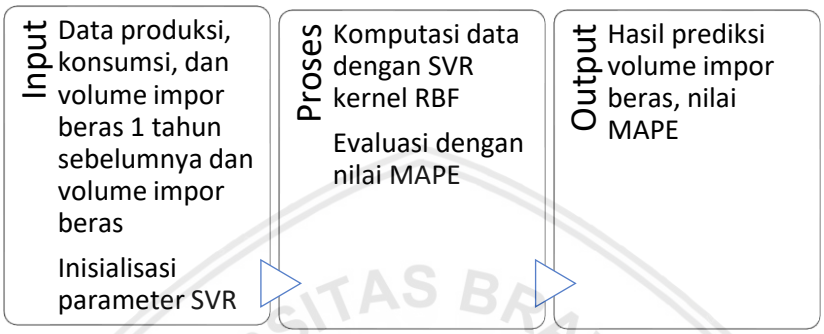
3.3 Pengumpulan Data

Data yang digunakan oleh penelitian ini merupakan data sekunder yang didapat dari laporan tahunan outlook padi yang ditulis oleh pusat data dan informasi Kementerian Pertanian terbitan tahun 2016. Adapun data yang digunakan adalah data produksi beras, data konsumsi beras dan data volume impor beras dari tahun 1970-2016. Adapun data yang dikumpulkan telah tertulis pada lampiran.

3.4 Implementasi

Implementasi dilakukan guna mentransformasikan hasil analisis dan perancangan yang dilakukan menjadi bentuk nyata. Sistem prediksi yang

diimplementasikan dengan metode SVR selanjutkan akan dilakukan pengujian dan evaluasi terhadap hasil. Metode pelatihan yang digunakan adalah *sequential learning* dengan fungsi kernel RBF. Sebelum dilakukan komputasi untuk menghitung prediksi, data yang ada terlebih dulu akan di normalisasi dengan normalisasi min-max. Setelah melalui tahapan perhitungan prediksi, data akan dihitung nilai MAPE nya. Hasil keluaran sistem berupa nilai prediksi dan nilai MAPE. Selanjutnya implementasi sistem pada penelitian dapat digambarkan dalam diagram berikut:



Gambar 3.1 Implementasi Sistem

- Input berupa data produksi, konsumsi, volume impor beras 1 tahun sebelumnya dan volume impor beras. Parameter SVR yang menjadi input sistem adalah σ (*sigma*), ε (*epsilon*), λ (*lambda*), *C* (kompleksitas), cLR *(constanta learning rate)* dan jumlah iterasi. Penentuan inputan parameter dilakukan hingga menghasilkan hasil prediksi yang optimal.
- Proses dalam sistem dilakukan perhitungan menggunakan SVR dengan *sequential learning* dan kernel RBF. Hasil dari proses perhitungan akan dihitung evaluasinya dengan MAPE.
- Hasil keluaran sistem berupa nilai hasil regresi, prediksi volume impor beras, nilai actual volume impor beras dan nilai MAPE

Sistem ini diimplementasikan dengan metode SVR dalam bentuk pemrograman java dengan menggunakan Netbeans IDE. Perangkat keras yang di gunakan mempunyai spesifikasi 6GB RAM, *processor* intel core i3 1,7 GHz dengan sistem operasi Windows 10 64 bit, dokumentasi dalam mendukung penulisan penelitian menggunakan Microsoft Word 2019 dan Microsoft Excel 2019. Antarmuka yang diimplementasikan disesuaikan dengan analisa kebutuhan dan perancangan yang telah dibuat.

3.5 Pengujian

Pengujian dilakukan untuk menilai keberhasilan sistem yang telah dibangun. Pengujian yang dilakukan dalam sistem ini adalah pengujian nilai parameter SVR, pengujian jumlah iterasi dan pengujian variasi data latih.

- Pengujian nilai parameter yang dilakukan untuk masing masing parameter SVR. Pengujian rentang nilai parameter meliputi pengujian nilai *lambda*,

nilai *epsilon*, cLR, kompleksitas dan nilai *sigma*. Pengujian parameter SVR dilakukan untuk mengetahui pengaruh parameter terhadap nilai MAPE.

Pada penelitian kali ini pengujian nilai parameter dilakukan mengacu pada batasan parameter penelitian penelitian yang telah dilakukan sebelumnya. Tabel batasan parameter pada penelitian sebelumnya ditunjukkan Tabel 3.1.

Tabel 3.1 Batasan Parameter Penelitian Sebelumnya

Parameter	Penelitian Oleh	Batas Bawah	Batas Atas
Sigma	Chen, Hong, Jun, & Jiulong, 2013	0.01	10
Lambda	Vijayakumar & Wu, 1999	0	5
Epsilon	Raharyani, Putri & Setiawan, 2018	1×10^{-15}	0.01
	Rifqi, Setiawan & Bacthiar, 2018	0.000001	0.1
cLr	Vijayakumar & Wu, 1999	0.0001	0.6
C	Li et al., 2005	0.8	1
	Chen et al., 2013	0.1	100

Berdasarkan tabel batasan parameter penelitian sebelumnya diatas, maka tabel yang digunakan untuk pengujian nilai parameter SVR dalam penelitian ini ditunjukkan oleh Tabel 3.2, Tabel 3.3, Tabel 3.4, Tabel 3.5, Tabel 3.6 dan Tabel 3.7

Tabel 3.2 Perancangan Pengujian Nilai *Parameter Sigma (σ)*

No.	Nilai *Sigma*	Nilai MAPE
1	0.01	
2	0.02	
3	0.03	
4	0.04	
5	0.05	
6	0.06	
7	0.07	
8	0.08	
9	0.09	

Tabel 3.3 Perancangan Pengujian Nilai Parameter *Lambda (λ)*

No.	Nilai *Lambda*	Nilai MAPE
1	0.1	
2	0.2	
3	0.3	
4	0.4	
5	0.5	
6	0.6	
7	0.7	
8	0.8	
9	0.9	

Tabel 3.4 Perancangan Pengujian Nilai Parameter *Constanta Learning Rate* (cLR)

No.	Nilai cLR	Nilai MAPE
1	0.005	
2	0.006	
3	0.007	
4	0.008	
5	0.009	
6	0.01	
7	0.02	
8	0.03	
9	0.04	
10	0.05	

Tabel 3.5 Perancangan Pengujian Nilai Parameter Kompleksitas (*C*) bagian 1

No.	Nilai *C*	Nilai MAPE
1	0.1	
2	0.2	
3	0.3	

Tabel 3.6 Perancangan Pengujian Nilai Parameter Kompleksitas (*C*) bagian 2

No.	Nilai *C*	Nilai MAPE
4	0.4	
5	0.5	
6	0.6	
7	0.7	
8	0.8	
9	0.9	
10	1	
11	2	
12	3	
13	4	
14	5	
15	6	
16	7	
17	8	
18	9	
19	10	

Tabel 3.7 Parencangan Pengujian Nilai Parameter *Epsilon (ε)*

No.	Nilai *Epsilon*	Nilai MAPE
1	0.00005	
2	0.00006	
3	0.00007	
4	0.00008	
5	0.00009	
6	0.0001	
7	0.0002	
8	0.0003	
9	0.0004	
10	0.0005	

- Pengujian jumlah iterasi yang dilakukan untuk mengetahui pengaruh jumlah iterasi terhadap nilai MAPE yang dihasilkan. Tabel perancangan pengujian jumlah iterasi ditunjukkan oleh Tabel 3.8.

Tabel 3.8 Perancangan Pengujian Jumlah Iterasi

No.	Jumlah iterasi	Nilai MAPE
1	100	
2	500	
3	1000	
4	2000	
5	2500	
6	5000	

- Pengujian variasi jumlah data latih dilakukan untuk mengetahui pengaruh data latih terhadap nilai MAPE yang didapatkan. Tabel 3.8 adalah tabel perancangan pengujian jumlah variasi data latih.

Tabel 3.9 Perancangan Pengujian Variasi Data Latih

No.	Jumlah Data Latih	Nilai MAPE
1	9	
2	13	
3	18	
4	23	
5	27	
6	32	
7	37	

3.6 Penarikan Kesimpulan

Pengambilan kesimpulan dilakukan setelah semua tahapan selesai. Kesimpulan didapat dari hasil pengujian yang dilakukan. Kesimpulan yang diambil memberikan jawaban dari rumusan masalah yang disebutkan sebelumnya. Saran diharapkan dapat menjadi masukan untuk memperbaiki kesalahan dan kekurangan yang ada dalam penelitian ini kemudian dapat digunakan sebagai pertimbangan untuk pengembangan penelitian selanjutnya.

BAB 4 ALGORITME

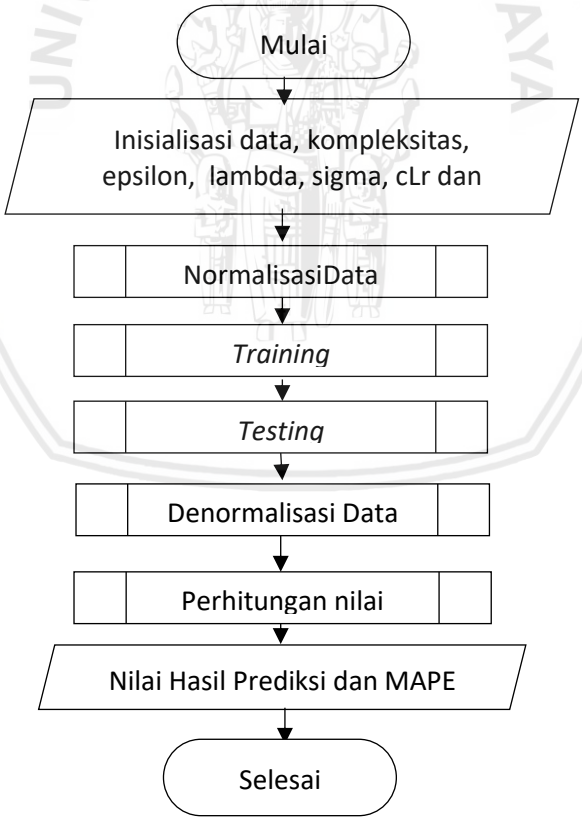
Bab ini akan membahas perancangan sistem yang akan dibuat meliputi formulasi permasalahan, diagram alir algoritma *Support Vector Regression*, manualisasi dari algoritma dan perancangan antarmuka yang digunakan.

4.1 Formulasi Permasalahan

Berdasarkan analisa kebutuhan yang telah dituliskan pada bab 3, permasalahan yang diselesaikan dengan *Support Vector Regression* untuk memprediksi volume impor beras nasional ditentukan oleh *input* masing masing parameter dan *input* data yang digunakan. Data yang digunakan untuk *input* adalah data produksi, konsumsi, volume impor beras 1 tahun sebelumnya dan volume impor beras dari tahun 1971 hingga 2016 yang nantinya akan dilakukan evaluasi dengan nilai MAPE.

4.2 Diagram Alir Algoritma Support Vector Regression

Diagram alir dari SVR yang menggambarkan tahapan yang dilakukan dalam algoritma ditunjukkan oleh Gambar 4.1.



Gambar 4.1 Diagram Alir Algoritma *Support Vector Regression* (SVR)

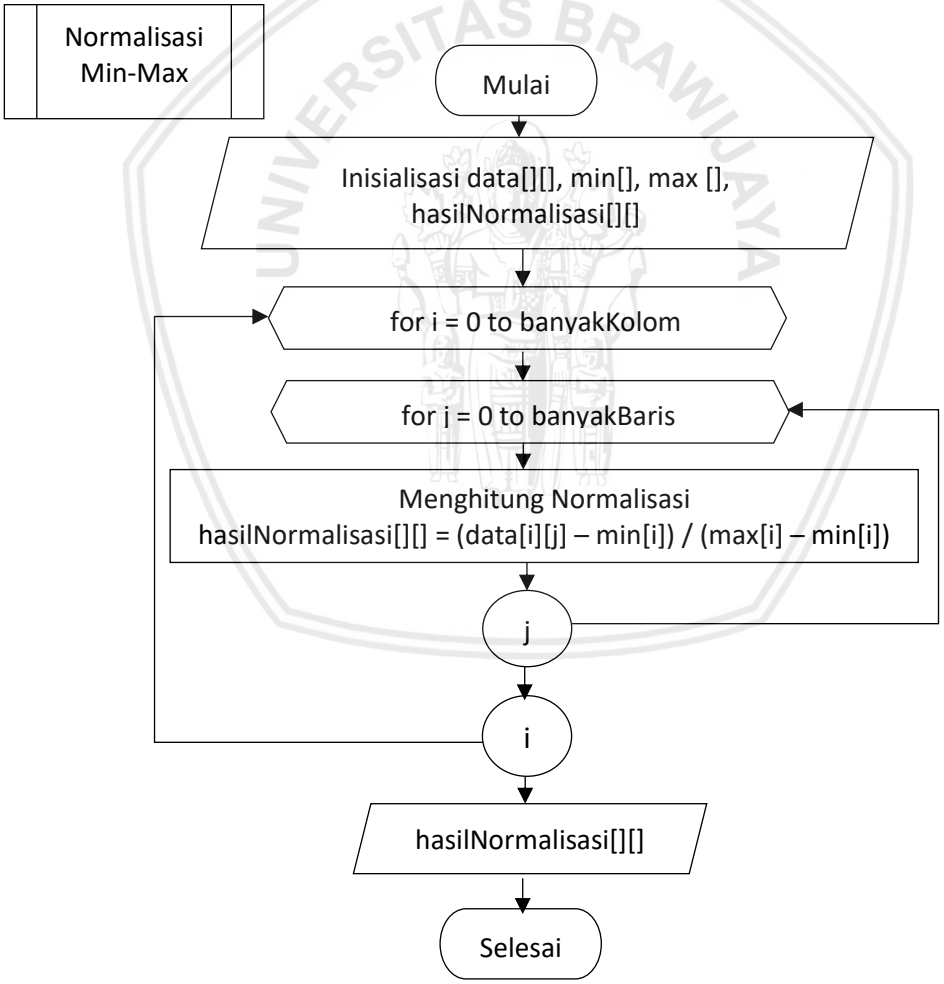
Tahapan perhitungan berdasarkan diagram alir diatas adalah:

1) Sistem menerima masukan berupa data yang akan diolah dan parameter yang digunakan.

2) Sistem melakukan proses normalisasi data dengan menggunakan normalisasi min-max
3) Melakukan proses *training* dengan metode SVR
4) Melakukan proses *testing*
5) Melakukan denormalisasi data
6) Perhitungan nilai MAPE
7) *Output* sistem yang dihasilkan berupa hasil prediksi dan nilai MAPE

4.2.1 Normalisasi Min-Max

Normalisasi data dilakukan agar data dengan range yang berbeda diskalakan nilai atributnya sehingga berada di range yang sama. Normalisasi data yang digunakan dalam penelitian ini adalah normalisasi min-max. Diagram alir dari proses normalisasi min-max ditunjukkan oleh Gambar 4.2.



Gambar 4.2 Diagram alir Normalisasi *Min-Max*

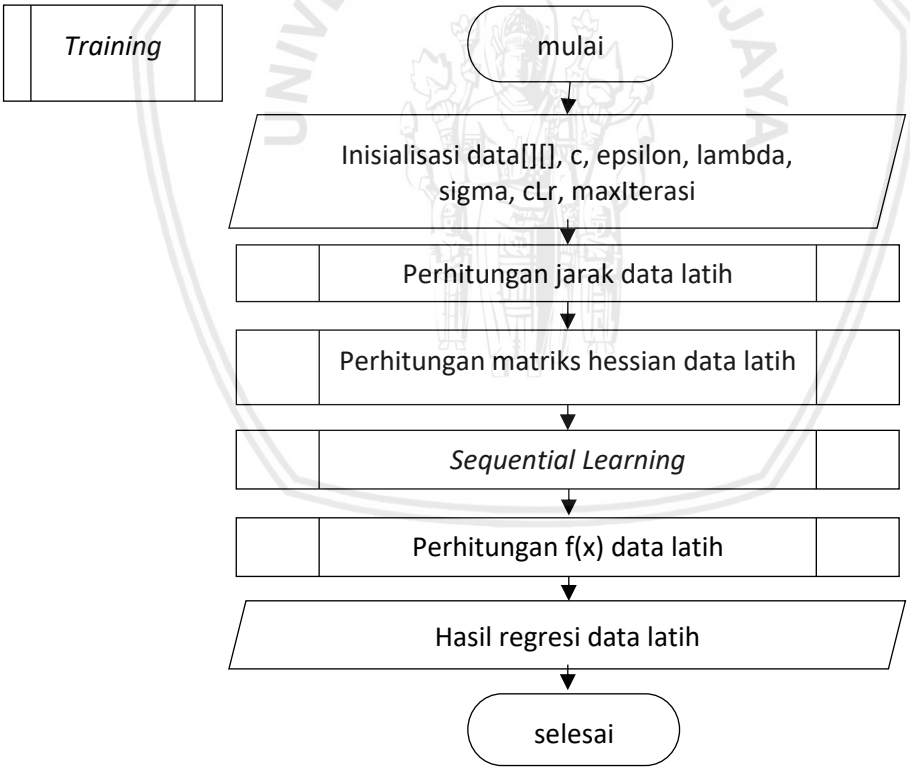
Tahapan normalisasi min-max berdasarkan diagram alir diatas adalah:

1) Inisialisasi data, nilai min dan nilai max, dan array hasilNormalisasi untuk menyimpan hasil dari proses normalisasi

2) Melakukan perulangan ketika i bernilai 0 sampai bernilai banyakKolom
3) Melakukan perulangan ketika j bernilai 0 sampai bernilai banyakBaris
4) Proses perhitungan normalisasi dengan membagi antara hasil pengurangan dari data dengan nilai min dan hasil pengurangan nilai max dengan nilai min (range) dan disimpan kedalam array hasilNormalisasi
5) Bila telah memenuhi batas j (banyakBaris) maka proses perulangan selesai
6) Bila telah memenuhi batas i (banyakKolom) maka proses perulangan selesai
7) Hasil berupa data yang telah ternormalisasi

4.2.2 *Training*

Training pada perancangan yang dimaksud adalah proses yang dilakukan untuk mendapatkan nilai regresi pada data latih. Secara garis besar proses *training* SVR dimulai dengan inisialisasi variabel variabel yang di perlukan, perhitungan matriks hessian, dan *sequential learning*. Diagram alir proses *training* ditunjukkan oleh Gambar 4.3.



Gambar 4.3 Diagram Alir Proses *Training*

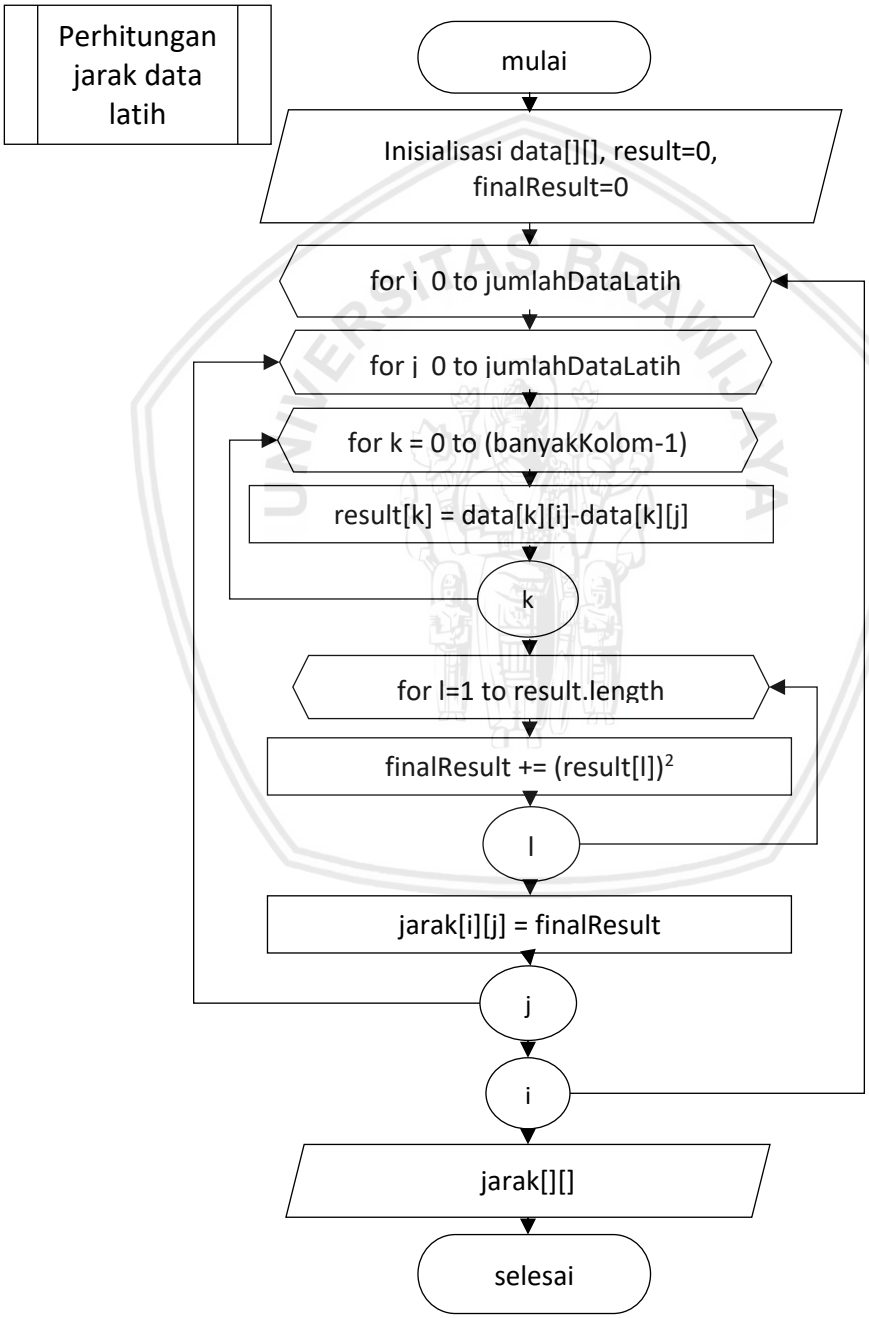
Tahapan *training* berdasarkan diagram alir diatas adalah:

1) Inisalisasi data, kompleksitas, epsilon, lambda, sigma, cLr, maxIterasi
2) Melakukan perhitungan jarak data latih
3) Melakukan perhitungan matriks hessian data latih

4) Perhitungan *sequential learning* dengan fungsi kernel
5) Melakukan perhitungan nilai f(x) yang merupakan nilai hasil regresi
6) Hasil *output* berupa nilai regresi dari data latih

4.2.2.1 Perhitungan Jarak Data Latih

Perhitungan jarak data latih digunakan untuk mengetahui posisi atau jarak data latih. Perhitungan data latih digambarkan pada diagram alir yang ditunjukkan oleh Gambar 4.4



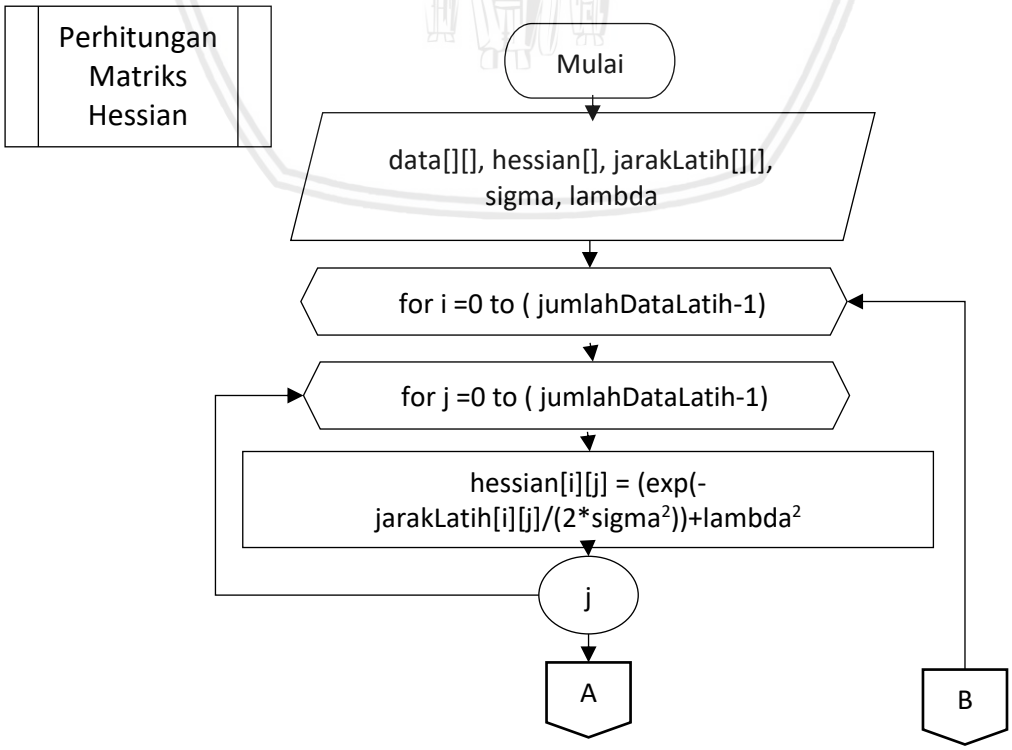
Gambar 4.4 Diagram Alir Perhitungan Jarak Latih

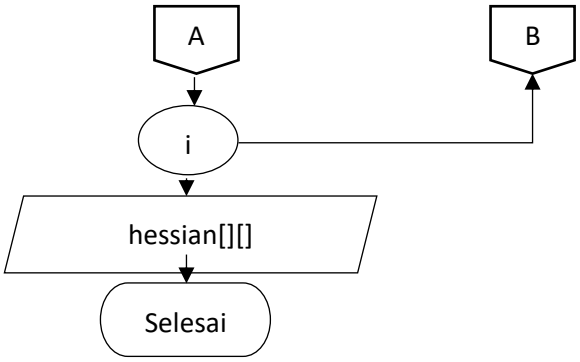
Tahapan perhitungan jarak latih berdasarkan diagram alir adalah:

1) Inisialisasi data, result untuk menyimpan hasil pengurangan antara jarak data latih baris ke-i dan ke-j
2) Melakukan perulangan mulai dari i=0 hingga i bernilai jumlahDataLatih
3) Melakukan perulangan mulai dari j=0 hingga j bernilai jumlahDataLatih
4) Melakukan perulangan mulai dari k=0 hingga k bernilai banyakKolom-1
5) Pengurangan data latih baris ke-i dan ke-j, hasilnya disimpan di array result
6) Bila telah mencapai batas maksimum perulangan k perulangan selesai
7) Melakukan perulangan mulai dari l=0 hingga l bernilai panjang array result
8) Perhitungan kuadrat dari result dan hasilnya disimpan dalam finalResult
9) Bila telah mencapai batas maksimum perulangan l perulangan selesai
10) Menjadikan hasil jarak data latih kedalam bentuk matriks dan disimpan dalam array jarak
11) Bila telah mencapai batas maksimum perulangan j perulangan selesai
12) Bila telah mencapai batas maksimum perulangan i perulangan selesai
13) Hasil yang didapatkan adalah berupa nilai jarak data latih

4.2.2.2 Perhitungan Matriks Hessian

Perhitungan matriks hessian digunakan untuk memetakan data menggunakan fungsi kernel. Kernel yang digunakan adalah kernel RBF. Proses perhitungan matriks hessian digambarkan pada Gambar 4.5





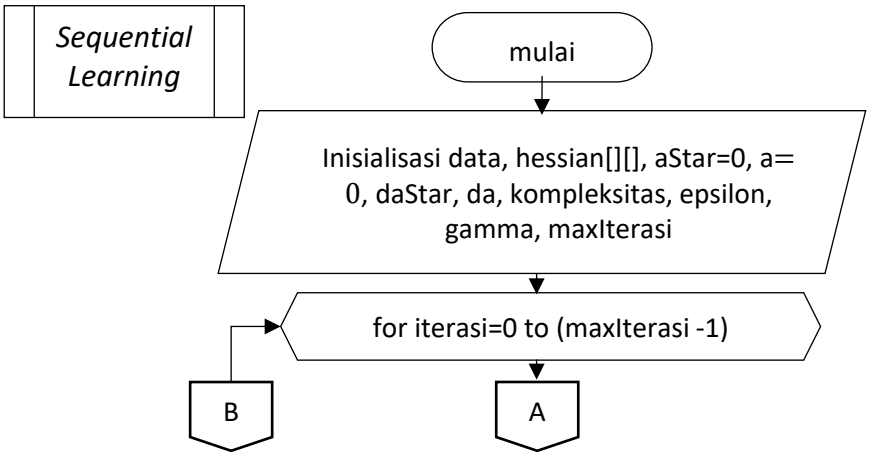
Gambar 4.5 Diagram Alir Perhitungan Matriks Hessian Data Latih

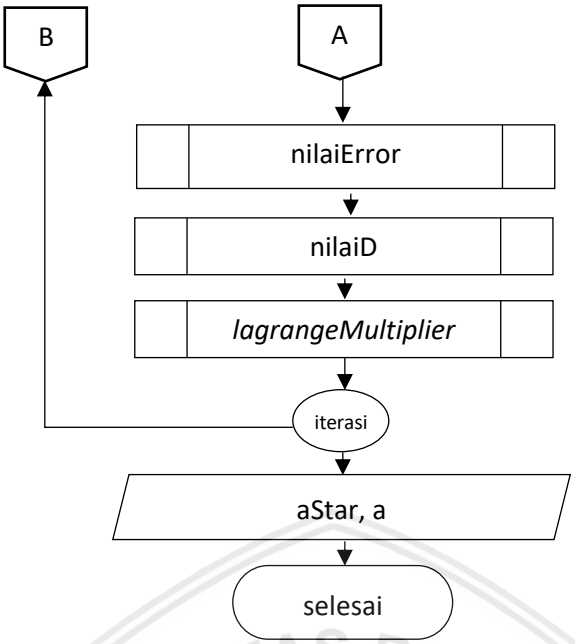
Tahapan proses perhitungan matriks hessian berdasarkan diagram alir diatas adalah:

1. Inisialisasi data, array hessian untuk menyimpan hasil perhitungan, jarak dataLatih, nilai sigma dan nilai lambda
2. Melakukan perulangan ketika i bernilai 0 sampai i bernilai jumlahDataLatih
3. Melakukan perulangan ketika j bernilai 0 sampai j bernilai jumlahDataLatih
4. Perhitungan matriks hessian dengan fungsi kernel RBF ditambahkan dengan nilai lambda kuadrat
5. Bila telah memenuhi batas j maka proses perulangan selesai
6. Bila telah memenuhi batas i maka proses perulangan selesai
7. Hasil berupa nilai matriks hessian data latih

4.2.2.3 Perhitungan *sequential learning*

Sequential Learning digunakan untuk melakukan perhitungan regresi dari data latih. Dalam melakukan *sequential learning* diperlukan Batasan berupa nilai iterasi. Proses *sequential* learning hanya digunakan untuk data latih. Data uji tidak melewati proses *sequential learning.* Proses s*equential Learning* ditunjukkan oleh Gambar 4.6





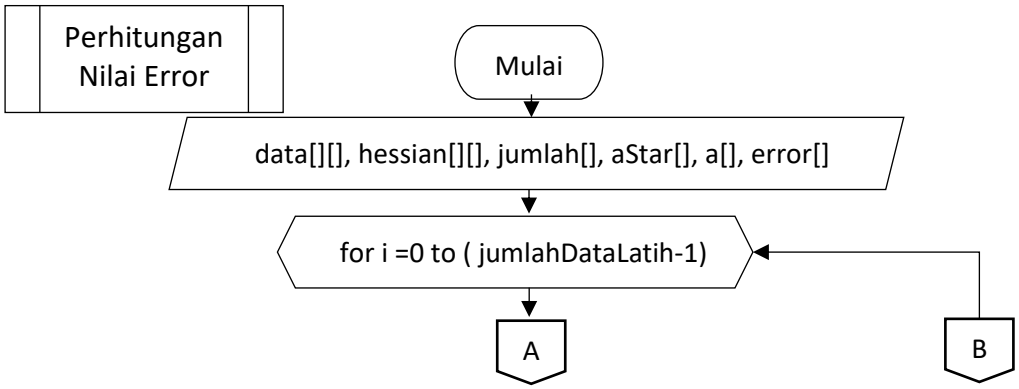
Gambar 4.6 Diagram Alir *Sequential Learning*

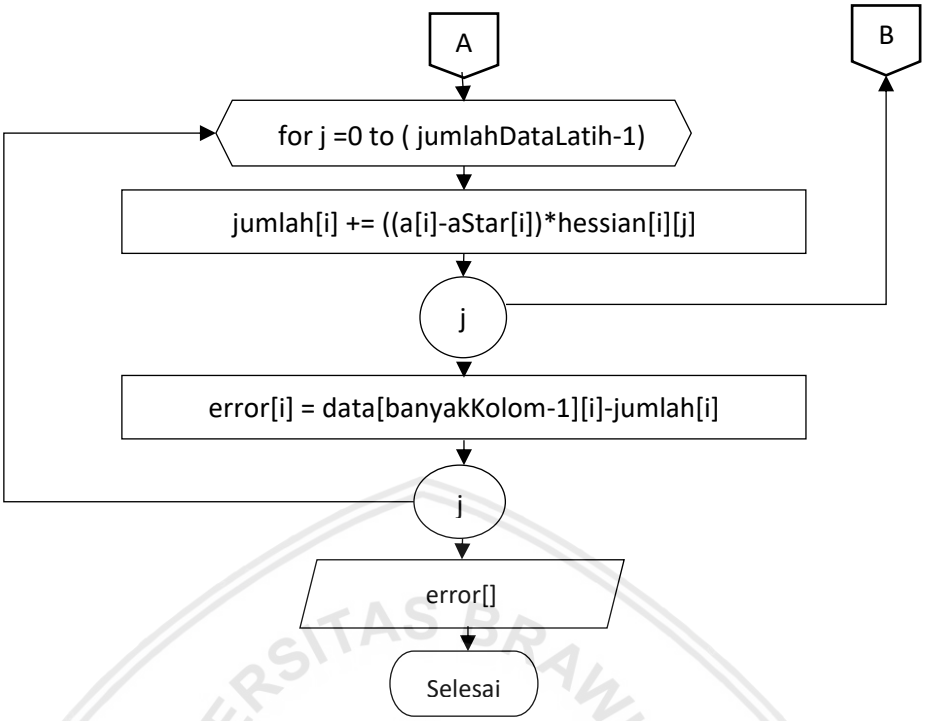
Tahapan proses *Sequential Learning* berdasarkan diagram alir diatas adalah:

1) Inisialisasi data, array hessian, aStar=0, a= 0, daStar, da, kompleksitas, epsilon, gamma, maxIterasi
2) Melakukan perulangan mulai dari iterasi=0 hingga iterasi maksimal
3) Melakukan perhitungan nilai error
4) Melakukan perhitungan nilai D
5) Melakukan perhitungan nilai lagrange multiplier
6) Bila telah memenuhi batas iterasi maksimum maka perulangan selesai
7) Keluaran yang dihasilkan adalah nilai lagrange multiplier

4.2.2.3.1 Perhitungan Nilai Error

Perhitungan nilai error merupakan bagian dari proses *sequential learning* yang dilakukan pada setiap data latih. Proses perhitungan nilai error ditunjukkan oleh diagram alir pada Gambar 4.7.





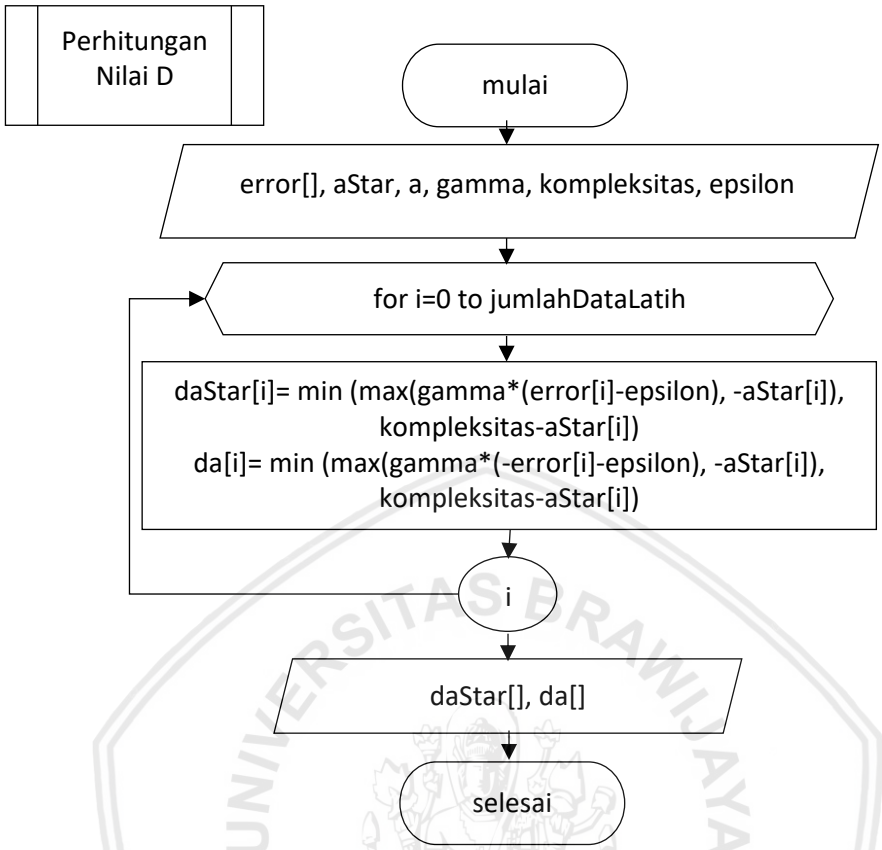
Gambar 4.7 Diagram Alir Perhitungan Nilai Error

Tahapan proses perhitungan nilai error berdasarkan diagram alir diatas adalah:

1. Inisialisasi data, matriks hessian, jumlah, aStar, a dan array error untuk menyimpan hasil perhitungan nilai error
2. Melakukan perulangan mulai dari i=0 hingga jumlahDataLatih-1
3. Melakukan perulangan mulai dari j=0 hingga jumlahDataLatih-1
4. Mengisi variabel array dengan hasil penjumlahan antara nilai aStar dikurangi a dan dikalikan dengan matriks hessian
5. Bila telah memenuhi batas maksimum nilai j maka perulangan selesai
6. Mengisi array error dengan hasil pengurangan antara data dengan jumlah
7. Bila memenuhi batas maksimum i maka perulangan selesai
8. Hasil keluaran berupa nilai error

4.2.2.3.2 Perhitungan Nilai $\delta\alpha_i^*$ dan $\delta\alpha_i$

Perhitungan nilai d merupakan bagian dari proses *sequential learning* yang dilakukan untuk mendapatkan nilai $\delta\alpha_i^*$ dan $\delta\alpha_i$. Proses perhitungan menggunakan nilai error yang telah didapatkan sebelumnya dan hasil keluarannya akan digunakan untuk proses *sequential learning* selanjutnya yaitu mencari nilai *lagrange multiplier*. Proses perhitungan nilai $\delta\alpha_i^*$ dan $\delta\alpha_l$ ditunjukkan oleh diagram alir pada Gambar 4.8



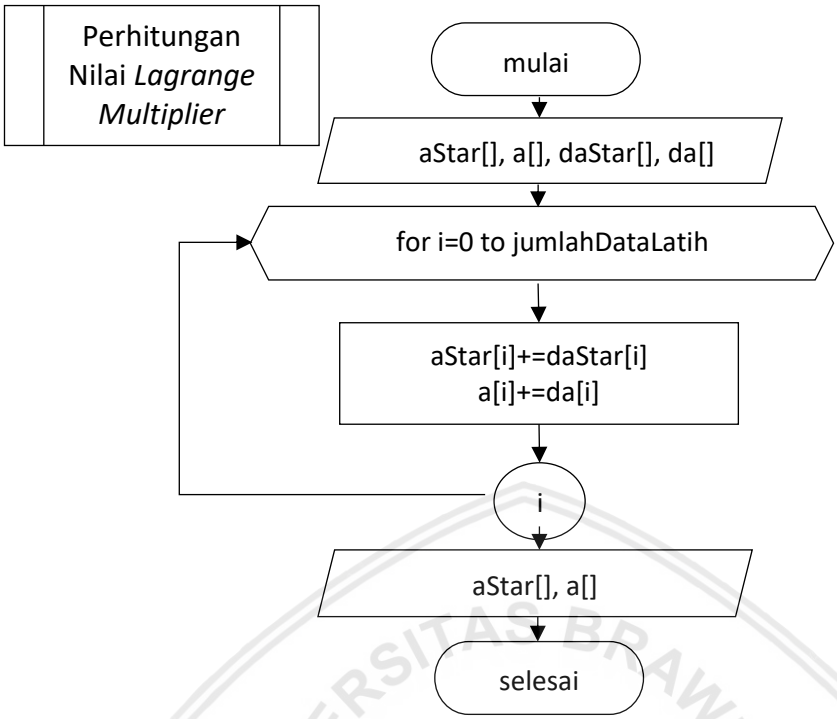
Gambar 4.8 Diagram Alir Proses Perhitungan Nilai $\delta\alpha_i^*$ dan $\delta\alpha_i$

Tahapan proses perhitungan nilai $\delta\alpha_i^*$ dan $\delta\alpha_i$ sesuai dengan diagram alir diatas adalah:

1. Inisialisasi nilai error, aStar, a, nilai gamma, kompleksitas, epsilon
2. Melakukan perulangan mulai dari 1=0 hingga jumlahData-1
3. Melakukan perhitungan nilai $\delta\alpha_i^*$ dan $\delta\alpha_i$ sesuai dengan rumus yang ada
4. Bila perulangan telah mencapai batas maksimum nilai I maka perulangan selesai
5. Hasil keluaran berupa nilai daStar dan da

4.2.2.3.3 Perhitungan Nilai *Lagrange Multiplier*

Perhitungan nilai *lagrange multiplier* merupakan nilai yang dihasilkan dari keseluruhan proses *sequential learning* yang nantinya digunakan untuk menghitung nilai hasil regresi. Poses perhitungan nilai *lagrange multiplier* ditunjukan oleh diagram alir pada Gambar 4.9



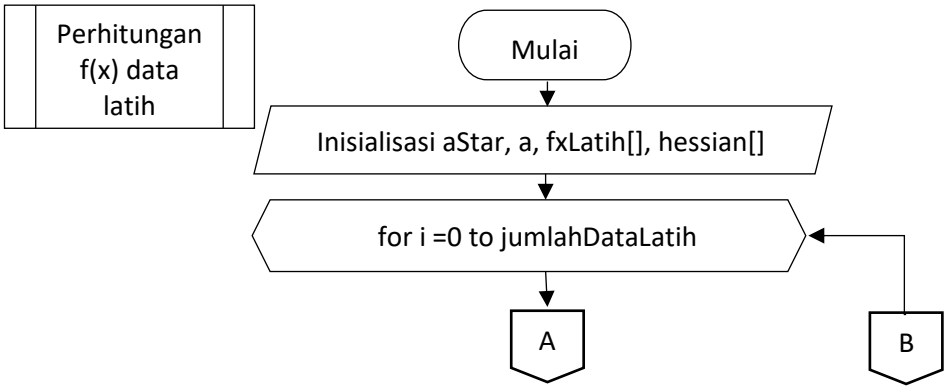
Gambar 4.9 Diagram Alir Proses Perhitungan nilai *Lagrange Multiplier*

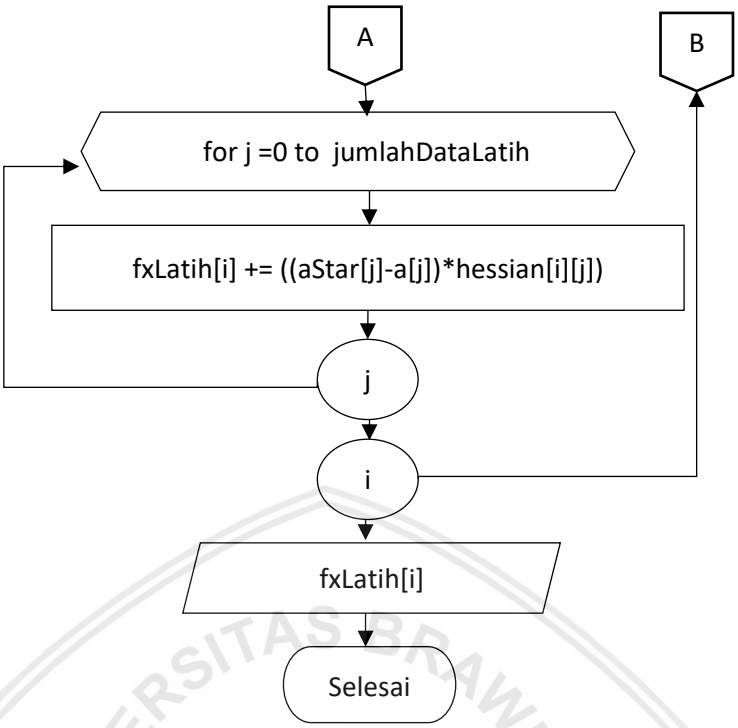
Tahapan proses perhitungan nilai *lagrange multiplier* berdasarkan diagram alir adalah:

1. Inisialisasi aStar, a, $\delta\alpha_i^*$ dan $\delta\alpha_i$
2. Melakukan perulangan dimulai dari i=0 hingga jumlahDataLatih
3. Perhitungan nilai aStar dengan penjumlahan nilai daStar. Perhitunga nilai a dengan penjumlahan nilai daStar.
4. Bila memenuhi batas perulangan nilai i maka perulangan selesai
5. Hasil keluaran berupa nilai a dan aStar

4.2.2.4 Perhitungan f(x) Data Latih

Perhitungan nilai f(x) data latih digunakan untuk menghitung nilai hasil regresi dari data latih. Proses perhitungan f(x) data latih ditunjukkan oleh diagram alir pada Gambar 4.10.





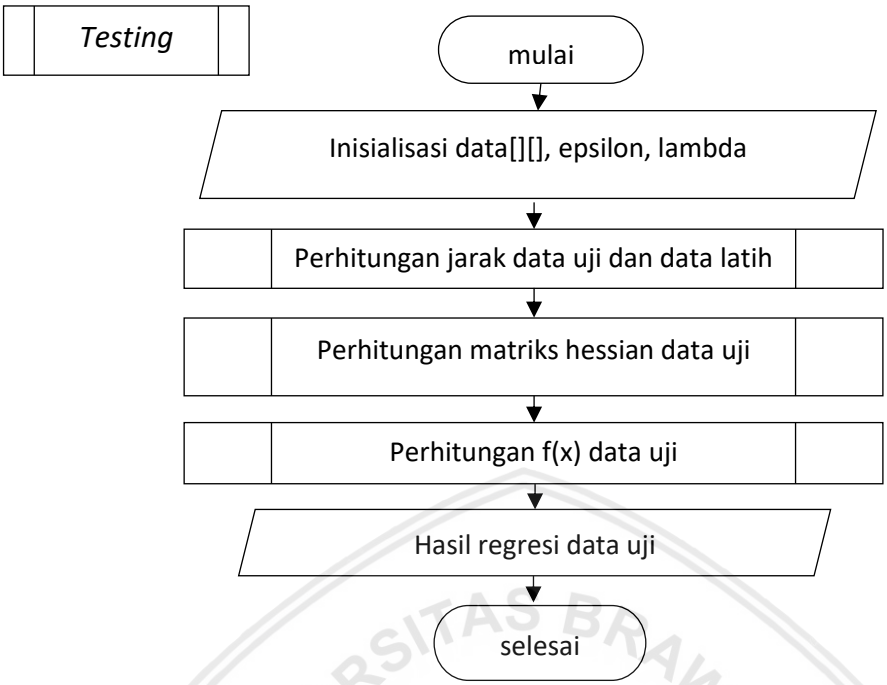
Gambar 4.10 Diagram Alir Perhitungan Nilai f(x) Data Latih

Tahapan perhitungan nilai f(x) data latih sesuai dengan diagram alir adalah:

1. Inisialisasi aStar, a fxLatih untuk menyimpan hasil perhitungan dan array hessian
2. Melakukan perulangan mulai dari i=0 hingga jumlahDataLatih-1
3. Melakukan perulangan mulai dari j=0 hingga jumlahDataLatih-1
4. Perhitungan nilai f(x) data latih yang disimpan dalam array fxLatih dengan menjumlahkan hasil perkalian antara hessian dengan hasil pengurangan aStar dan a
5. Bila telah memenuhi batas maksimum nilai j maka perulangan selesai
6. Bila memenuhi batas maksimum i maka perulangan selesai
7. Hasil perhitungan adalah nilai fxLatih

4.2.3 *Testing*

Testing digunakan untuk mendapatkan nilai f(x) data uji yang merupakan hasil dari prediksi. Proses pengujian dihitung menggunakan matriks hessian data uji yang didapatkan melalui perhitungan jarak data uji dengan data latih. Sedangkan untuk nilai aStar dan a merupakan nilai hasil dari *sequential learning* yang berasal dari proses pelatihan. Proses *testing* ditunjukkan oleh diagram alir pada Gambar 4.11



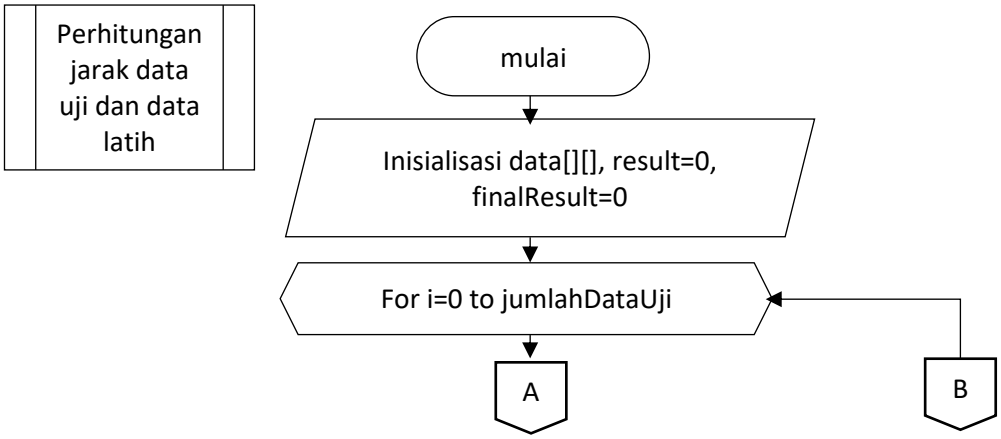
Gambar 4.11 Diagram Alir Proses *Testing*

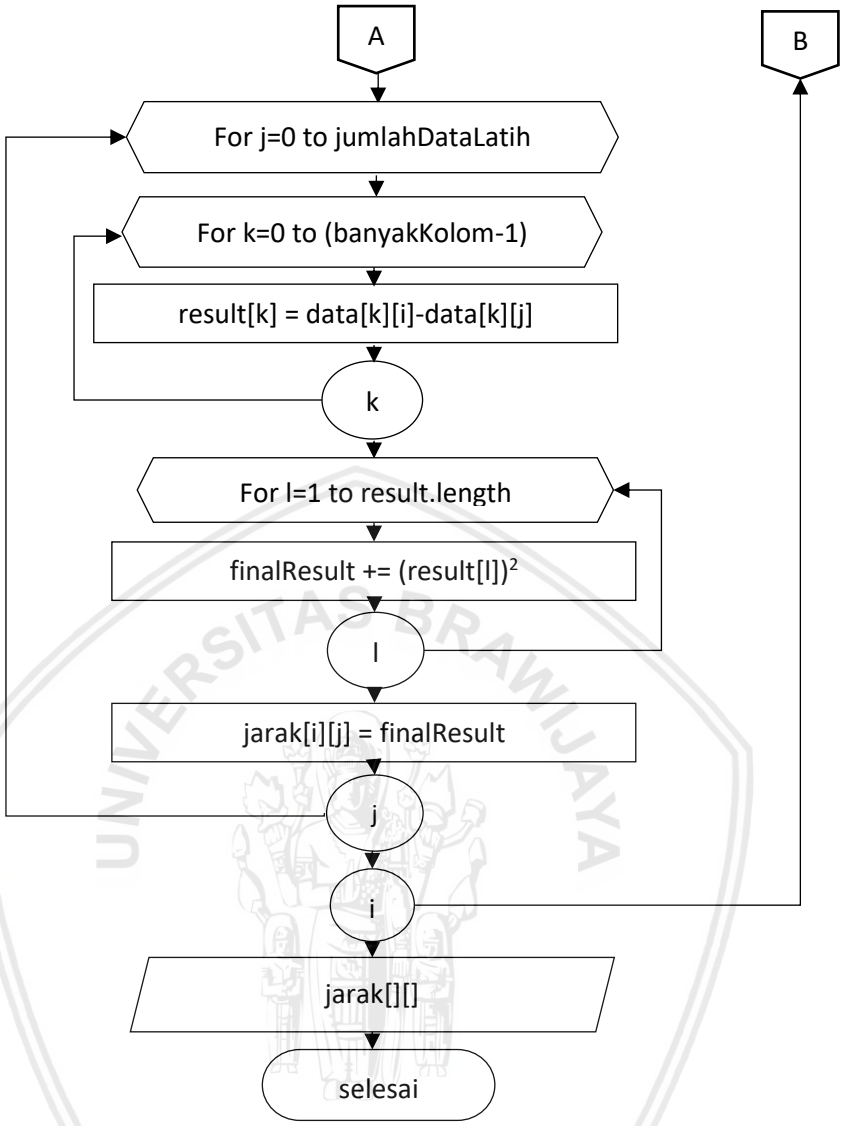
Tahapan Pelatihan berdasarkan diagram alir adalah:

1) Inisalisasi data, epsilon, lambda
2) Melakukan perhitungan jarak antara data uji dengan data latih
3) Melakukan perhitungan matriks hessian data uji
4) Melakukan perhitungan nilai f(x) yang merupakan nilai hasil regresi
5) Hasil pelatihan berupa nilai regresi dari data uji

4.2.3.1 Perhitungan Jarak Data Uji dan Data Latih

Perhitungan jarak data uji dan data latih dilakukan untuk mencari jarak antara data uji dengan data latih yang hasilnya akan digunakan untuk proses perhitungan matriks hessian. Proses perhitungan jarak data uji dan latih ditunjukkan oleh diagram alir pada Gambar 4.12.





Gambar 4.12 Diagram Alir Proses Perhitungan Jarak Data Uji dan Data Latih

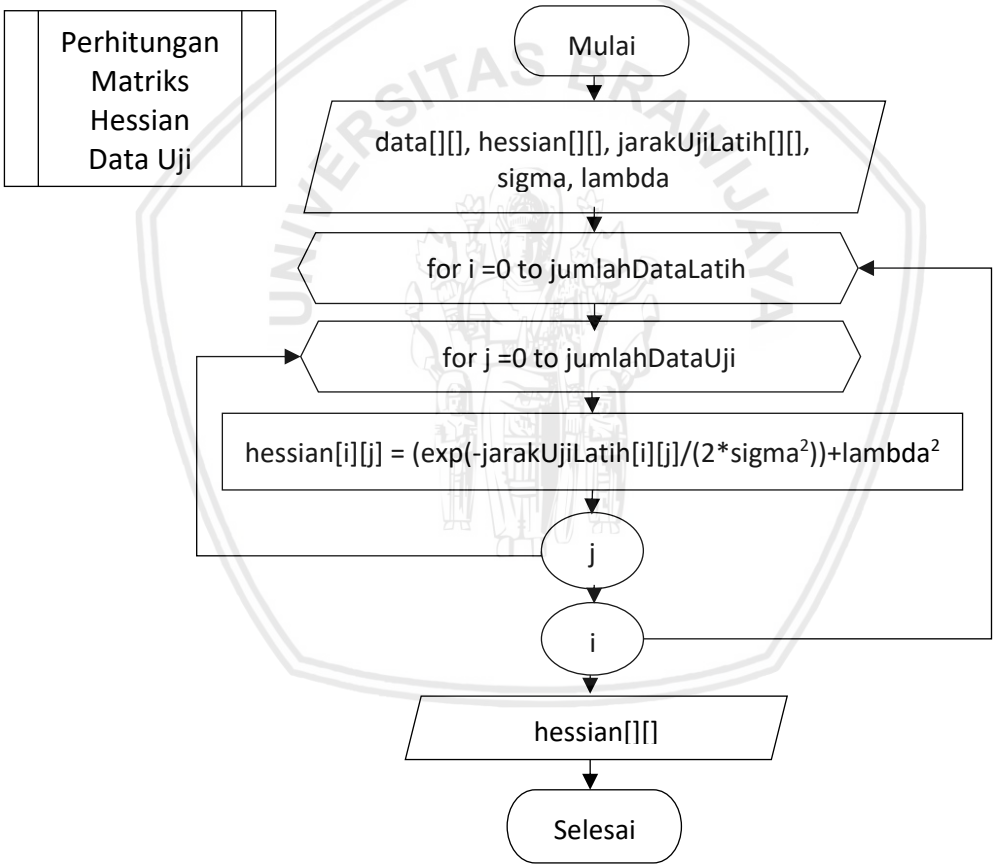
Tahapan proses perhitungan sesuai dengan diagram alir adalah:

1) Inisialisasi data, result untuk menyimpan hasil pengurangan antara jarak data uji dan latih baris ke-i dan ke-j
2) Melakukan perulangan mulai dari i=0 hingga i bernilai jumlahDataUji
3) Melakukan perulangan mulai dari j=0 hingga j bernilai jumlahDataLatih
4) Melakukan perulangan mulai dari k=0 hingga k bernilai banyakKolom-1
5) Pengurangan data latih dengan data uji baris ke-i dan ke-j, hasilnya disimpan di array result
6) Bila telah mencapai batas maksimum perulangan k perulangan selesai
7) Melakukan perulangan mulai dari l=0 hingga l bernilai panjang array result
8) Perhitungan kuadrat dari result dan hasilnya disimpan dalam finalResult

9) Bila telah mencapai batas maksimum perulangan l perulangan selesai
10) Menjadikan hasil jarak data latih kedalam bentuk matriks dan disimpan dalam array jarak
11) Bila telah mencapai batas maksimum perulangan j perulangan selesai
12) Bila telah mencapai batas maksimum perulangan i perulangan selesai
13) Hasil yang didapatkan adalah berupa nilai jarak data latih

4.2.3.2 Perhitungan Matriks Hessian Data Uji

Perhitungan matriks hessian data uji digunakan untuk memetakan data uji menggunakan fungsi kernel. Kernel yang digunakan adalah kernel RBF. Perhitungan matriks hessian data uji ditunjukkan oleh diagram alir pada Gambar 4.13



Gambar 4.13 Diagram Alir Proses Perhitungan Matriks Hessian Data Uji

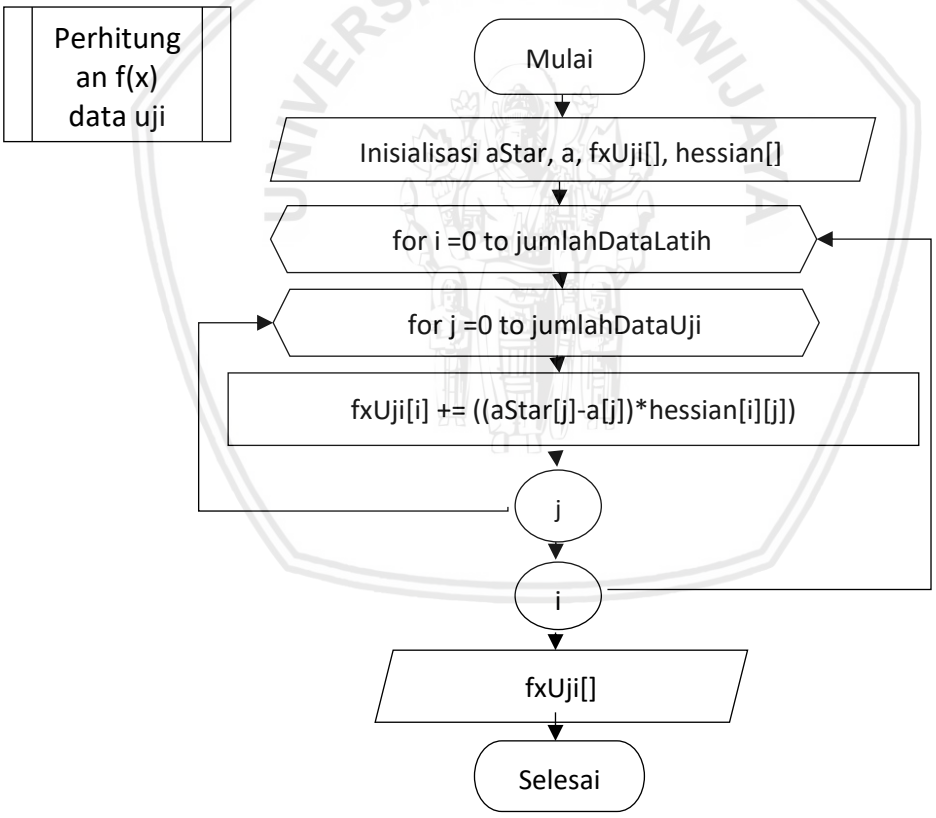
Tahapan proses perhitungan matriks hessian data uji berdasarkan diagram alir diatas adalah:

1. Inisialisasi data, array hessian untuk menyimpan hasil perhitungan, jarakUjiLatih, nilai sigma dan nilai lambda
2. Melakukan perulangan ketika i bernilai 0 sampai i bernilai jumlahDataLatih-1

3. Melakukan perulangan ketika j bernilai 0 sampai j bernilai jumlahDataLatih-1
4. Perhitungan matriks hessian dengan fungsi kernel RBF ditambahkan dengan nilai lambda kuadrat
5. Bila telah memenuhi batas j (jumlahDataLatih-1) maka proses perulangan selesai
6. Bila telah memenuhi batas i (jumlahDataUji-1) maka proses perulangan selesai
7. Hasil berupa nilai matriks hessian data Uji

4.2.3.3 Perhitungan Nilai f(x) Data Uji

Perhitungan nilai f(x) data latih digunakan untuk menghitung nilai hasil regresi dari data latih. Proses perhitungan f(x) data latih ditunjukkan oleh diagram alir pada Gambar 4.14



Gambar 4.14 Diagram Alir Proses Perhitungan f(x) Data Uji

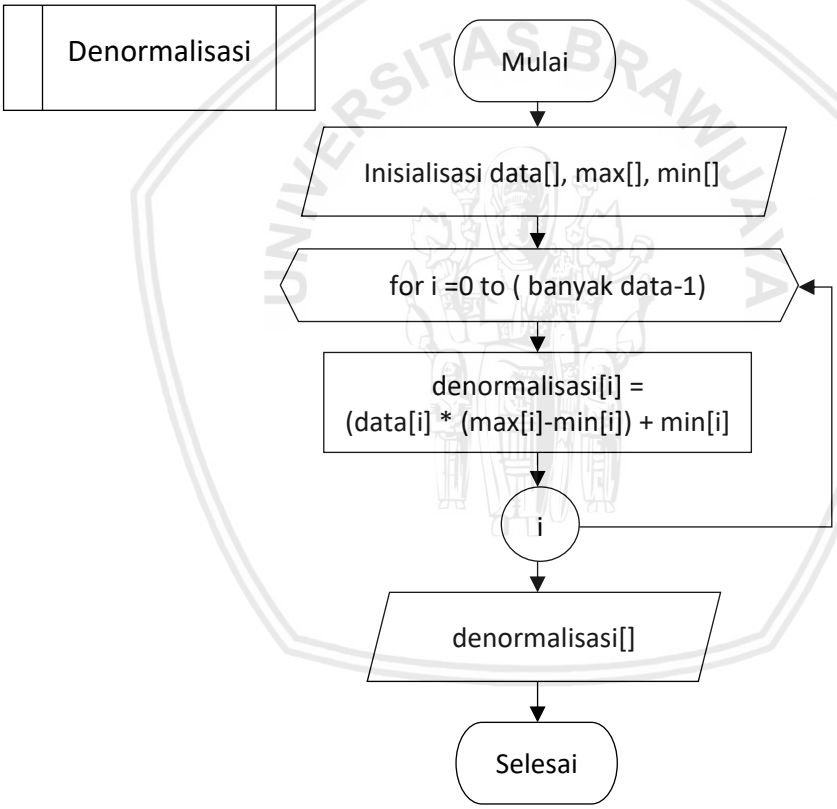
Tahapan perhitungan nilai f(x) data uji sesuai dengan diagram alir adalah:

1. Inisialisasi aStar, a fxUji untuk menyimpan hasil perhitungan dan array hessian
2. Melakukan perulangan mulai dari i=0 hingga jumlahDataLatih-1
3. Melakukan perulangan mulai dari j=0 hingga jumlahDataUji-1

4. Perhitungan nilai f(x) data uji yang disimpan dalam array fxUji dengan menjumlahkan hasil perkalian antara hessian dengan hasil pengurangan aStar dan a
5. Bila telah memenuhi batas maksimum nilai j maka perulangan selesai
6. Bila memenuhi batas maksimum i maka perulangan selesai
7. Hasil perhitungan adalah nilai fxUji

4.2.4 Denormalisasi

Proses denormalisasi digunakan untuk mengembalikan data yang sebelumnya dinormalisasi. Proses denormalisasi dapat digambarkan pada diagram alir Gambar 4.15



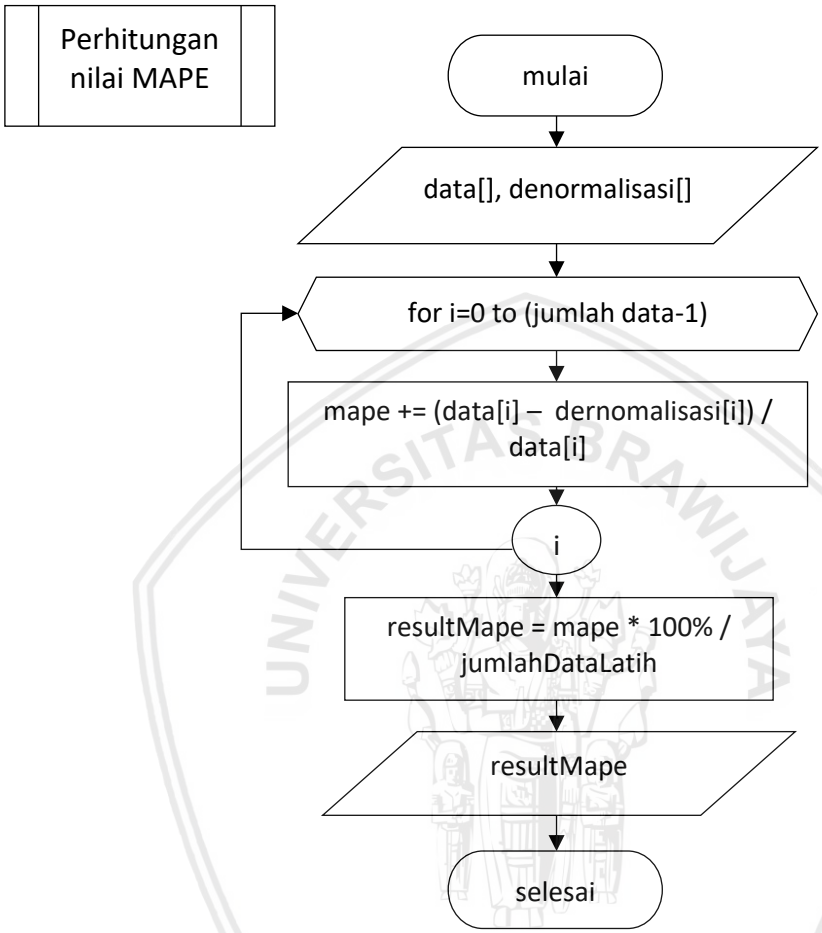
Gambar 4.15 Diagram Alir Denormalisasi

Tahapan denormalisasi data berdasarkan diagram alir adalah:

1) Inisialisasi data, nilai max, dan min
2) Proses perhitungan denormalisasi dengan mengalikan data hasil prediksi f(x) dan range (nilai maksimal – nilai minimal) kemudian hasilnya ditambahkan dengan nilai minimal.

4.2.5 Perhitungan MAPE

Perhitungan nilai MAPE digunakan untuk evaluasi tingkat nilai error dengan presentase absolut yang didapat dari hasil prediksi dengan data aktual. Perhitungan nilai MAPE ditunjukkan oleh diagram alir pada Gambar 4.15



Gambar 4.16 Diagram alir perhitungan Nilai MAPE

Tahapan perhitungan nilai MAPE berdasarkann diagram alir adalah:

1) Insialisasi data actual dan hasil prediksi yang telah di denomalisasi
2) Melakukan perhitungan nilai mape dengan mengurangi nilai actual dengan nilai denormalisasi dibagi dengan nilai actual
3) Hasil MAPE dikalikan dengann 100% dan dibagi dengan jumlah data
4) Hasil keluaran berupa nilai MAPE

4.3 Manualisasi

Pada manualisasi untuk studi kasus penelitian ini digunakan data latih dari data produksi, konsumsi dan volume impor beras dan target dari tahun 2008 hingga 2013 dimana target merupakan volume impor beras yang akan diprediksi ditunjukkan pada Tabel 4.1.

Tabel 4.1 Data Latih

Tahun	Konsumsi Beras	Produksi Beras	Impor Beras	Target
2008	37100	38306.96	289.274	250.276
2009	38000	40360.22	250.276	687.583
2010	38550	40716.87	687.583	2744.261
2011	31427.34	38244.2	2744.261	1927.563
2012	47842.56	40163.03	1927.563	472.665
2013	50277.03	41455.98	472.665	815.285

Sedangkan untuk data uji yang digunakan adalah data konsumsi, produksi, impor beras dan target tahun 2014 dan 2015 yang ditunjukkan pada Tabel 4.2

Tabel 4.2 Data Uji

Tahun	Konsumsi Beras	Produksi Beras	Impor Beras	Target
2014	52570.59	39451.84	815.285	816.63
2015	33350	43610	816.63	1073.72

4.3.1 Inisialisasi Parameter

Tahap awal yang dilakukan sebelum melakukan perhitungan SVR adalah melakukan inisialisasi parameter SVR yang dibutuhkan. Dalam studi kasus ini, parameter SVR yang digunakan ditunjukkan dalam Tabel 4.3.

Tabel 4.3 Inisialisasi Parameter

cLR	Kompleksitas (C)	Epsilon (ε)	Gamma (γ)	Lambda (λ)	Sigma (σ)
0.01	50	0.0001	0.03227	0.005	0.005

4.3.2 Normalisasi Data

Normalisasi data yang digunakan adalah menggunakan normalisasi min-max. berikut adalah contoh perhitungan normalisasi min-max untuk data latih pertama dari konsumsi beras.

X = 50277.03

X_{min} = 13507.14

X_{max} = 52570.59

$$x' = \frac{(50277.03 - 13507.14)}{(52570.59 - 13507.14)} = 0.941286$$

X merupakan nilai dari data yang akan di normalisasi, X_{min} adalah nilai minimum dari kolom data yang akan di normalisasi dan X_{max}. Hasil normalisasi data latih dan data uji ditunjukkan pada Tabel 4.4 dan Tabel 4.5

Tabel 4.4 Hasil Normalisasi Data Latih

Tahun	Konsumsi Beras	Produksi Beras	Impor beras	Target
2008	0.603963	0.827354	0.060577	0.052349
2009	0.627002	0.8942	0.052349	0.144614
2010	0.641082	0.905811	0.144614	0.578539
2011	0.458746	0.825311	0.578539	0.406229
2012	0.878965	0.88778	0.406229	0.099269
2013	0.941286	0.929874	0.099269	0.171557

Tabel 4.5 Hasil Normalisasi Data Uji

Tahun	Konsumsi Beras	Produksi Beras	Impor beras	Target
2014	1	0.864627	0.171557	0.17184
2015	0.507965	1	0.17184	0.226082

4.3.3 Training

Training dilakukan untuk memperoleh nilai *lagrange multiplier* yang nantinya akan digunakan untuk menghitung regresi. Sebelumnya data latih akan dihitung jaraknya, kemudian dilakukan perhitungan nilai matriks hessian dan dilanjutkan dengan proses *sequential learning*. Hasil dari proses *sequential learning* terakhir berupa nilai *lagrange multiplier* akan digunakan untuk perhitungan nilai *f(x)* data uji.

4.3.3.1 Perhitungan Jarak Data Latih

Perhitungan jarak latih dilakukan pada masing masing data latih. Berikut merupakan contoh perhitungan jarak data latih pertama dengan data latih ke-2. Adapun data yang digunakan adalah data yang telah dinormalisasi. Hasil perhitungan seluruh data latih ditunjukkan oleh Tabel 4.6 dan Tabel 4.7.

$$\|x_1 - x_2\|^2 = (0.603963 - 0.627002)^2 + (0.827354 - 0.8942)^2 + (0.060577 - 0.052349)^2 = 0.005067$$

Tabel 4.6 Hasil Perhitungan Jarak Data Latih Bagian 1

Data ke-	1	2	3	4	5	6
1	0	0.005067	0.014596	0.289377	0.198753	0.125795
2	0.005067	0	0.008846	0.309932	0.188758	0.102249

Tabel 4.7 Hasil Perhitungan Jarak Data Latih Bagian 2

Data ke-	1	2	3	4	5	6
3	0.014596	0.008846	0	0.228018	0.125357	0.092758
4	0.289377	0.309932	0.228018	0	0.210178	0.473478
5	0.198753	0.188758	0.125357	0.210178	0	0.09988
6	0.125795	0.102249	0.092758	0.473478	0.09988	0

4.3.3.2 Perhitungan matriks hessian data latih

Matriks hessian dihitung dengan menggunakan fungsi kernel. Adapun kernel yang digunakan adalah kernel RBF. Perhitungan matriks hessian dilakukan dengan menggunakan jarak yang telah dihitung sebelumnya. Berikut adalah contoh perhitungan matriks hessian untuk data ke-2 dengan fungsi kernel RBF.

$$[R]_{1,2} = K(x_1 x_2) + \lambda^2 = (exp - \left(\frac{0.005067}{2\,x(0.005)^2}\right)) + 0.005^2 = 0.994921$$

Perhitungan matriks hessian untuk seluruh data latih contoh studi kasus ditunjukkan oleh Tabel 4.8.

Tabel 4.8 Hasil Perhitungan Matriks Hessian Data Latih

Ri,j	1	2	3	4	5	6
1	0.999975	0.994921	0.985486	0.748717	0.819736	0.881777
2	0.994921	0.999975	0.991169	0.733485	0.82797	0.902785
3	0.985486	0.991169	0.999975	0.796095	0.882163	0.911394
4	0.748717	0.733485	0.796095	0.999975	0.810425	0.622826
5	0.819736	0.82797	0.882163	0.810425	0.999975	0.904926
6	0.881777	0.902785	0.911394	0.622826	0.904926	0.999975

4.3.3.3 *Sequential Learning*

Dalam proses *sequential learning* dilakukan perhitungan *lagrange multiplier* (α_i^*-α_i) nilai error dan nilai $\delta\alpha_i^*$ dan δ. Untuk inisialisasi awal nilai *lagrange multiplier* adalah 0. Perhitungann *sequential learning* hanya dilakukan untuk data latih. Berikut adalah contoh perhitungan *sequential learning*.

Perhitungan nilai eror (E)

$$E_1 = y_1 - \sum_{i=1}^{i=6} (\alpha_i^* - \alpha_i) R_{1,6}$$

$$= 0.052349 - \Big(\big((0-0)*0.999975\big) + \big((0-0)*0.994921\big) + \big((0-0)*0.985486\big) + \big((0-0)*0.748717\big)\Big) + \big((0-0)*0.819736\big) + ((0-0)*0.881777) = 0.052349$$

Perhitungan $\delta\alpha_i^*$ dan $\delta\alpha_i$

$$\delta\alpha_1^* = \min(max[\gamma(E_1 - \sigma), -\alpha_1^*], C - \alpha_1^*)$$
$$= \min(\max[0.03227(0.052349 - 0.005), -0], 50 - 0)$$
$$= 0.000261$$

$$\delta\alpha_1 = \min(max[\gamma(-E_1 - \sigma), -\alpha_1], C - \alpha_1)$$
$$= \min(\max[0.03227(-0.052349 - 0.005), -0], 50 - 0) = 0$$

Perhitungan nilai α_i^* dan α_i baru

$$\alpha_1^* = \alpha_1^* + \delta\alpha_1^* = 0 + 0.000261 = 0.000261$$

$$\alpha_1 = \alpha_1 + \delta\alpha_1 = 0 + 0 = 0$$

Hasil perhitungan *sequential learning* untuk contoh studi kasus pada iterasi ke-1 ditunjukkan pada Tabel 4.9.

Tabel 4.9 Hasil Perhitungan *Sequential Learning* Iterasi 1

Nilai E	Nilai $\delta\alpha^*$	Nilai $\delta\alpha$	Nilai α^*	Nilai α
0.052349	0.000261	0	0.000261	0
0.144614	0.000723	0	0.000723	0
0.578539	0.002892	0	0.002892	0
0.406229	0.002031	0	0.002031	0
0.099269	0.000496	0	0.000496	0
0.171557	0.000857	0	0.000857	0

Untuk proses *sequential learning* dilakukan hingga iterasi mencapai maksimum atau mencapai konvergensi dengan syarat $\max(|\delta_{\alpha i}^*|) < \varepsilon$ dan $\max(|\delta_{\alpha i}|) < \varepsilon$. Hasil perhitungan *sequential learning* untuk studi kasus pada iterasi ke-15 ditunjukkan pada Tabel 4.10.

Tabel 4.10 Hasil Perhitungan *Sequential Learning* iterasi 15

Nilai E	Nilai $\delta\alpha^*$	Nilai $\delta\alpha$	Nilai α^*	Nilai α
-0.02451	-0.00012	0.000122	0.000889	0.000381
0.067665	0.000338	0	0.007805	0
0.499384	0.002496	0	0.040264	0
0.334647	0.001673	0	0.027651	0
0.025079	0.000125	0	0.004515	0
0.100042	0.0005	0	0.010036	0

4.3.3.4 Perhitungan nilai *f(x)* data latih

Perhitungan nilai *f(x)* dilakukan menggunakan nilai *lagrange multiplier* hasil dari *sequential learning* data latih. Berikut adalah contoh perhitungan nilai *f(x)* untuk data uji:

$$f(1) = \sum_{i=1}^{i=6} (a_i^* - a_i)(K(x, x_i) + \lambda^2)$$
$$= ((0.000889 - 0.000381) * 0.999975)$$
$$+ ((0.007805 - 0) * 0.994921)$$
$$+ ((0.040264 - 0) * 0.985486)$$
$$+ ((0.027651 - 0) * 0.748717)$$
$$+ ((0.004515 - 0) * 0.819736)$$
$$+ ((0.010036 - 0) * 0.881777) = 0.081206$$

Hasil perhitungan nilai *f(x)* data latih sesuai dengan studi kasus yang digunakan ditunjukkan pada Tabel 4.11.

Tabel 4.11 Nilai *f(x)* data latih

f(x)
0.081206
0.081299
0.083642
0.075719
0.078404
0.075534

4.3.4 *Testing*

Testing dilakukan untuk memperoleh nilai *f(x)* yang merupakan nilai hasil dari regresi. Sebelumnya data uji dihitung jaraknya dengan data latih dan matriks hessian seperti yang telah ditunjukkan di proses *training*.

4.3.4.1 Perhitungan Jarak Data Uji dan Data Latih

Perhitungan jarak pada pengujian dilakukan antara data uji dan data latih yang nantinya akan digunakan untuk menghitung matriks hessian data uji. Berikut adalah contoh perhitungan jarak antara data latih dengan data uji untuk data pertama.

$$\|x_1 - x_2\|^2 = (1 - 0.603963)^2 + (0.864627 - 0.827354)^2$$
$$+ (0.171557 - 0.60557)^2 = 0.170551$$

Hasil perhitungan jarak data uji dengan data latih sesuai dengan studi kasus dapat ditunjukkan oleh Tabel 4.12.

Tabel 4.12 Hasil Perhitungan Jarak data latih dengan data uji

Uji/latih	1	2	3	4	5	6
1	0.170551	0.154213	0.131244	0.460136	0.070257	0.01293
2	0.051402	0.039642	0.027333	0.198343	0.205173	0.197952

4.3.4.2 Perhitungan Matriks Hessian Data Uji

Selain untuk data latih, perhitungan matriks hessian juga dilakukan untuk data uji. Nilai dari matriks hessian akan digunakan dalam proses mencari nilai f(x). Berikut adalah contoh perhitungan matriks hessian data uji untuk data pertama:

$$[R]_{1,2} = K(x_1 x_2) + \lambda^2 = exp - \frac{0.170551}{2\,x(0.005)^2} + 0.005^2 = 0.843183$$

Perhitunggan matriks hessian untuk data uji sesuai dengan studi kasus dapat ditunjukkan dalam Tabel 4.13.

Tabel 4.13 Hasil Perhitungan Matriks Hessian Data Uji

Uji/latih	1	2	3	4	5	6
1	0.843183	0.857072	0.876984	0.631191	0.932133	0.987129
2	0.949874	0.961111	0.973014	0.820073	0.814491	0.820393

4.3.4.3 Perhitungan nilai f(x) data uji

Perhitungan nilai f(x) dilakukan menggunakan nilai *lagrange multiplier* hasil dari *sequential learning* data latih. Berikut adalah contoh perhitungan nilai f(x) untuk data uji:

$$\begin{aligned} f(x) = \sum_{i=1}^{i=6} (a_i^* - a_i)(K(x, x_i) + \lambda^2) \\ = \big((0.000889 - 0.000381) * 0.843183\big) \\ + \big((0.007805 - 0) * 0.857072\big) \\ + \big((0.040264 - 0) * 0.876984\big) \\ + \big((0.027651 - 0) * 0.631191\big) \\ + \big((0.004515 - 0) * 0.932133\big) \\ + \big((0.010036 - 0) * 0.987129\big) = 0.081206 \end{aligned}$$

Hasil Perhitungan nilai *f(x)* data uji sesuai dengan studi kasus dapat ditunjukkan oleh Tabel 4.14.

Tabel 4.14 Hasil Perhitungan nilai *f(x)* data uji

f(x)
0.073997
0.081748

4.3.5 Denormalisasi

Proses denormalisasi dilakukan untuk mengembalikan data sebelum dinormalisasi. Berikut adalah contoh perhitungan denormalisasi terhadap data uji:

f(x) = 0.2440206

x_{max} = 4741.86

x_{min} = 2.158

$$denormalisasi = \big(\mathrm{f(x)} * (\max - \min)\big) + \min$$
$$= \big(0.073997 * (4741.86 - 2.158)\big) + 2.158 = 352.8815$$

Hasil perhitungan denormalisasi dan data aktual untuk data latih dan data uji dapat ditunjukkan pada Tabel 4.15.

Tabel 4.15 Hasil Perhitungan Denormalisasi Dan Data Aktual Data Latih Dan Data Uji

Data	Denormalisasi	Data Aktual
Data latih 1	387.0511	289.274
Data latih 2	387.4896	250.276
Data latih 3	398.5962	687.583
Data latih 4	361.0433	2744.261
Data latih 5	373.7678	1927.563
Data latih 6	360.1644	472.665
Data Uji 1	352.8812	816.63
Data Uji 2	389.6189	1073.72

4.3.6 Perhitungan MAPE

Proses terakhir yang dilakukan adalah perhitungan nilai MAPE. Berikut contoh perhitungan nilai MAPE untuk data uji:

$$MAPE = \frac{1}{2} * \left(ABS\frac{(816.63 - 352.8812)}{352.8812}\right) + \left(ABS\frac{(1073.72 - 389.6189)}{389.6189}\right) * 100\% = 60.25\%$$

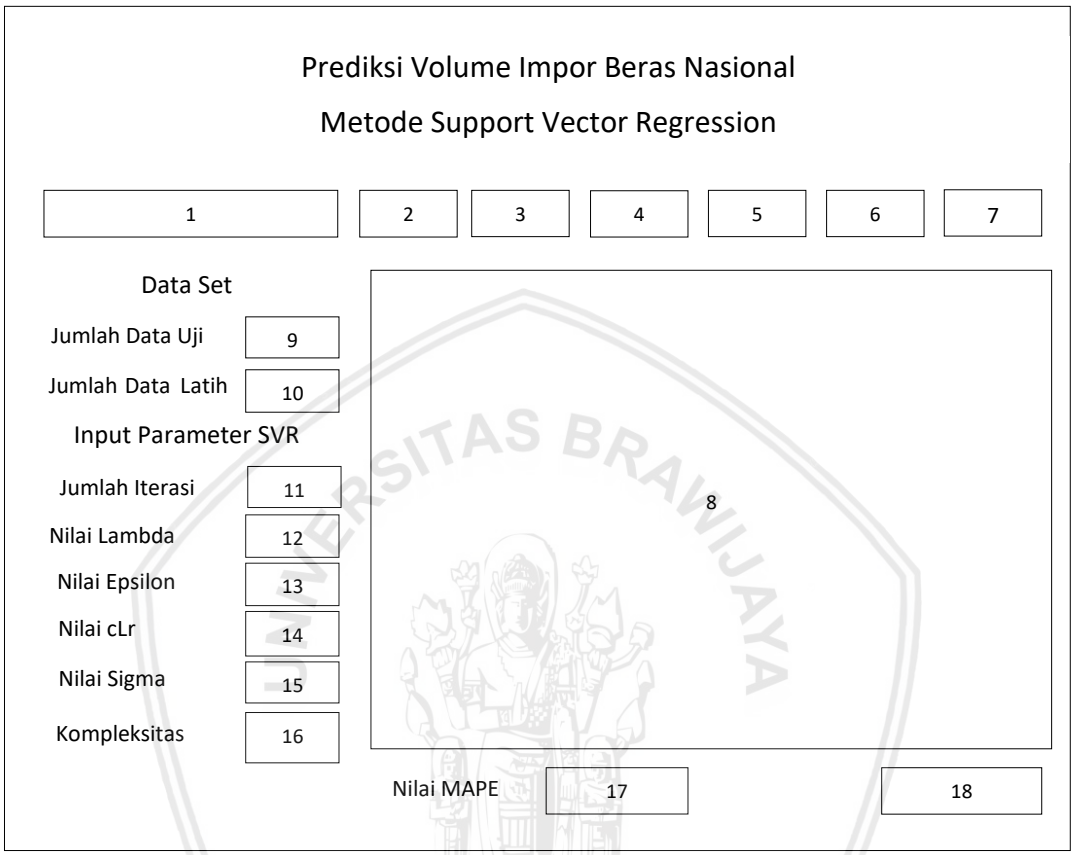
Hasil perhitungan nilai MAPE data latih dan data uji dapat ditunjukkan pada Tabel 4.16.

Tabel 4.16 Hasil Perhitungan Nilai MAPE Data Latih dan Data Uji

Data Latih	Data Uji
53.65%	60.25%

4.4 Perancangan User Interface

Perancangan user interface dilakukan guna merancang halaman antarmuka agar pengguna dapat berkomunikasi dengan sistem. Perancangan user interface ditunjukkan oleh Gambar 4.17.



Gambar 4.17 Perancangan *User interface*

Keterangan:

1. *Button Impor* yang digunakan untuk mengimpor data dalam bentuk excel
2. *Button* Data untuk melihat data yang digunakan dalam sistem
3. *Button* Normalisasi untuk melihat data yang telah dinormalisasi
4. *Button* Data Latih untuk melihat data latih yang digunakan prediksi
5. *Button* Data Uji untuk melihat data uji yang digunakan prediksi
6. *Button* Training untuk melihat hasil dari pelatihan sistem
7. *Button* Testing untuk melihat hasil pengujian data
8. *Dropdown* untuk memilih jumlah data Uji yang digunakan
9. *Textfield* untuk menginputkan jumlah data latih yang ingin digunakan
10. *Textfield* untuk menginputkan jumlah iterasi yang ingin dilakukan
11. *Textfield* untuk menginputkan nilai lambda

12. *Textfield* untuk menginputkan nilai epsilon
13. *Textfield* untuk menginputkan nilai cLR
14. *Textfield* untuk menginputkan nilai sigma
15. *Textfield* untuk menginputkan nilai kompleksitas
16. *Button* proses untuk mengambil nilai inputan dari *user*
17. *Label* hasil nilai MAPE
18. *Button* Hitung Prediksi untuk melakukan prediksi baru

BAB 5 IMPLEMENTASI

Pada bab ini akan dijelaskan hasil implementasi yang dilakukan berdasarkan perancangan yang telah dibuat sebelumnya. Didalamnya berisi kode program, penjelasan mengenai kode program dan *screenshoot* antarmuka yang telah di implementasikan.

5.1 Implementasi Membaca File

Proses awal ketika program dijalankan adalah user memilih file yang dijadikan sebagai inputan dengan format file xls atau excel. Kemudian sistem membaca file yang telah diinputkan oleh user. Implementasi dari proses membaca file ditunjukkan pada Kode Program 5.1.

```
1  private void bacaFile(File file) {
2          try {
3              Workbook workbook = Workbook.getWorkbook(file);
4              Sheet sheet = workbook.getSheet(0);
5              banyakKolom = sheet.getColumns();
6              banyakBaris = sheet.getRows();
7
8              /* deklarasi header tabel */
9              data = new String[banyakBaris][banyakKolom];
10             kolom = new String[banyakKolom];
11             minFitur = new Double[banyakKolom];
12             maxFitur = new Double[banyakKolom];
13
14             //Header dan isi dipisah
15             dataInDouble = new
16             Double[banyakKolom][banyakBaris];
17
18             /* ambil data */
19             for (int i = 0; i < banyakBaris; i++) {
20                 for (int j = 0; j < banyakKolom; j++) {
21                     Cell cell = sheet.getCell(j, i);
22                     if (cell.getType() == CellType.NUMBER)
23                     {
24                         NumberCell numberCell =(NumberCell)
25                         cell;
26                         Double angka =
27                         Double.valueOf(numberCell.getValue(
28                         )).doubleValue();
29                         data[i][j] =
30                         Double.toString(angka);
31                     } else if (cell.getType() ==
32                       CellType.DATE) {
33                         DateCell dateCell = (DateCell)
34                         cell;
35                         Date date = dateCell.getDate();
36                         data[i][j] = date.toString();
37                     } else {
38                         data[i][j] = cell.getContents();
39                     }
40                 }
41             }
42             //Isi header dan dataInDouble
```

```
43              int index = 0;
44              for (int i = 0; i < banyakBaris; i++) {
45                  for (int j = 0; j < banyakKolom; j++) {
46                      if (i == 0) {
47                          kolom[j] = data[i][j];
48                      } else if (i > 0) {
49                          dataInDouble[j][i] =
50                          Double.parseDouble(data[i][j]);
51                      }
52                  }
53              }
54              for (int i = 0; i < banyakKolom; i++) {
55                  System.out.print(kolom[i]);
56                  for (int j = 1; j < banyakBaris; j++) {
57                      System.out.print(dataInDouble[i][j]);
58                  }
59                  System.out.println("");
60              }
61          } catch (Exception e) {
62              JDialog.setDefaultLookAndFeelDecorated(true);
63              JOptionPane.showMessageDialog(null, "Error: " +
64              e, "Error!", JOptionPane.ERROR_MESSAGE);
65          }
66      }
67
```

Kode Program 5.1 Membaca File

Proses implementasi membaca file dilakukan dengan menggunakan *library* jxl. Data yang diinputkan user dalam bentuk excel kolom diinisialisasikan dengan variabel `banyakKolom` dan baris diinisialisasikan dengan `banyakBaris`. Data yang diambil dari dalam excel disimpan dalam array data bertipe string dan diubah tipenya kedalam bentuk double dan disimpan dalam array `dataInDouble`.

5.2 Implementasi Normalisasi Data

Implementasi normalisasi data terdapat beberapa proses yang dilakukan seperti perhitungan nilai min dan nilai max. Implementasi normalisasi data ditunjukkan oleh Kode Program 5.2.

```
1   private void normalisasiData() {
2           hasilNormalisasi = new
3           Double[banyakKolom][banyakBaris];
4              for (int i = 0; i < banyakKolom; i++) {
5                 for (int j = 1; j < banyakBaris; j++) {
6                   hasilNormalisasi[i][j] =
7                   ((dataInDouble[i][j] - minFitur[i]) /
8                   (maxFitur[i] - minFitur[i]));
9                 }
10             }
11  }
```

Kode Program 5.2 Normalisasi Data

Implementasi normalisasi data dilakukan dengan inisialisasi array `hasilNormalisasi` sebanyak `banyakKolom` dan `banyakBaris` untuk menyimpan hasil dari perhitungan normalisasi. Kemudian dilakukan perulangan sebanyak jumlah kolom dan jumlah baris untuk menghitung nilai normalisasi yaitu

dataInDouble dikurangi dengan minFitur dibagi dengan maxFitur dikurangi minFitur.

Proses pencarian nilai minimum merupakan bagian dari proses normalisasi. Pencarian nilai minimum ditunjukkan oleh Kode Program 5.3.

```
1   private void calculateMinFitur() {
2           for (int i = 0; i < banyakKolom; i++) {
3               minFitur[i] = dataInDouble[i][1];
4               for (int j = 2; j < banyakBaris; j++) {
5                   if (dataInDouble[i][j] < minFitur[i]) {
6                       minFitur[i] = dataInDouble[i][j];
7                   }
8               }
9           }
10   }
```

Kode Program 5.3 Pencarian Nilai Minimum

Implementasi pencarian nilai minimum dilakukan dengan melakukan perulangan sejumlah banyakKolom, inisialisasi array minFitur untuk menyimpan nilai minimum per fitur data, perulangan sejumlah banyakBaris dan kondisi if bila dataInDouble kurang dari minFitur maka minFitur bernilai dataInDouble.

Proses pencarian nilai maksimum merupakan bagian dari proses normalisasi. Pencarian nilai maksimum ditunjukkan oleh Kode Program 5.4

```
1   private void calculateMaxFitur() {
2           for (int i = 0; i < banyakKolom; i++) {
3               maxFitur[i] = dataInDouble[i][1];
4               for (int j = 2; j < banyakBaris; j++) {
5                   if (dataInDouble[i][j] > maxFitur[i]) {
6                       maxFitur[i] = dataInDouble[i][j];
7                   }
8               }
9           }
10      }
```

Kode Program 5.4 Pencarian Nilai Maksimum

Implementasi pencarian nilai maksimum dilakukan dengan melakukan perulangan sejumlah banyakKolom, inisialisasi array maxFitur untuk menyimpan nilai minimum per fitur data, perulangan sejumlah banyakBaris dan kondisi if bila dataInDouble lebih dari maxFitur maka maxFitur bernilai dataInDouble.

5.3 Implementasi Pelatihan

Implementasi pelatihan dilakukan dengan beberapa proses sesuai dengan perancangan yang telah dilakukan yaitu implementasi perhitungan jarak data latih, implementasi perhitungan matriks hessian data latih, implementasi *sequential learning* dan implementasi perhitungan f(X) data latih.

5.3.1 Implementasi Perhitungan Jarak Data Latih

Implementasi perhitungan jarak data latih dibagi menjadi 2 fungsi yaitu fungsi formula untuk menghitung jarak dan fungsi matriks dari perhitungan jarak. Fungsi formula menghitung jarak data latih ditunjukkan oleh Kode Program 5.5

```
1  private double formula(int batasIndexI, int indexJBase, int
2  indexJ) {
3          Double[] result = new Double[batasIndexI - 1];
4          double finalResult = 0;
5            for (int i = 0; i < batasIndexI - 1; i++) {
6              result[i]                                    =
7              dataLatihNormalisasi[i][indexJBase]          -
               dataLatihNormalisasi[i][indexJ];
8            }
9          finalResult = Math.pow(result[0], 2);
10           for (int i = 1; i < result.length; i++) {
11             finalResult += Math.pow(result[i], 2);
12           }
13
14         return finalResult;
15 }
```

Kode Program 5.5 Formula Menghitung Jarak Data Latih

Perhitungan jarak data latih dilakukan inisialisasi `result` dan `finalResult` untuk menyimpan hasil perhitungan jarak data latih. Perulangan dilakukan sejumlah `batasIndexI` yaitu banyak baris data latih dilanjutkan dengan perhitungan pengurangan untuk masing masing data latih dan disimpan dalam array `result`. Kemudian mengisi array `finalResult` dengan nilai kuadrat dari `result` dilanjutkan perulangan lagi sejumlah array `result`. Hasil akhir berupa `finalResult` ditambahkan dengan `result` yang dikuadratkan.

Adanya matriks jarak data latih dalam perhitungan jarak data latih digunakan untuk mempermudah perhitungan selanjutnya. Implementasi fungsi matriks perhitungan jarak ditunjukkan oleh Kode Program 5.6.

```
1  private void calculateDistanceLatih() {
2      distanceLatih = new
       Double[jumlahDataLatih][jumlahDataLatih];
3          for (int i = 0; i < jumlahDataLatih; i++) {
4              for (int j = 0; j < jumlahDataLatih; j++) {
5                  distanceLatih[i][j] = formula(banyakKolom,
                   i, j);
6              }
7          }
8  }
```

Kode Program 5.6 Matriks Jarak Data Latih

Implementasi matriks jarak data latih dilakukan dengan inisialisasi array `distanceLatih` untuk meyimpan hasil matriks jarak data latih. Kemudian dilakukan perulangan sebanyak 2 kali sejumlah data latih dilanjutkan dengan mengisi array `distanceLatih` dengan nilai dari fungsi `formula`.

5.3.2 Implementasi Perhitungan Matriks Hessian Data Latih

Implementasi perhitungan matriks hessian data latih sesuai dengan perancangan yang telah dibuat ditunjukkan dalam kode Kode Program 5.7.

```
1  private void calculateHessianLatih(double sigma, double
2  lambda, int jumlahData) {
3          maksHessian = 0.0;
4
```

```
5           resultHessianLatih = new
6           Double[jumlahDataLatih][jumlahDataLatih];
7           resultHessLatihGui = new
8           Double[jumlahDataLatih][jumlahDataLatih];
9           for (int i = 0; i < jumlahDataLatih; i++) {
10              for (int j = 0; j < jumlahDataLatih; j++) {
11                  resultHessianLatih[i][j] = (Math.exp(-
12                  distanceLatih[i][j] / (2 *
13                  (Math.pow(sigma, 2))))) +
14                  (Math.pow(lambda, 2));
15                  if (resultHessianLatih[i][j] >
16                  maksHessian) {
17                      maksHessian =
18                      resultHessianLatih[i][j];
19                  }
20              }
21          }
22          for (int i = 0; i < jumlahDataLatih; i++) {
23              for (int j = 0; j < jumlahDataLatih; j++) {
24                  resultHessLatihGui[j][i] =
25                  resultHessianLatih[i][j];
26              }
27          }
    }
```

Kode Program 5.7 Perhitungan Matriks Hessian Data Latih

Implementasi perhitungan matriks hessian diproses didalam perulangan sebanyak 2 kali. Hasil matriks hessian disimpan didalam array `resultHessianLatih` dengan menggunakan formulasi sesuai dengan rumus yang ada. Kemudian untuk dapat menampilkan hasilnya di GUI dilakukan perulangan lagi dan disimpan didalam `resultHessianLatihGui`.

5.3.3 Implementasi *Sequential Learning*

Implementasi *sequential learning* dilakukan dengan beberapa proses sesuai dengan perancangan yang telah dilakukan yaitu implementasi perhitungan nilai error, perhitungan nilai $\delta\alpha^*$ dan $\delta\alpha$ juga perhitungan nilai lagrange multiplier. Implementasi proses *sequential learning* ditunjukkan oleh Kode Program 5.8

```
1   private void sequentialLearning() {
2           aStar = new Double[jumlahDataLatih];
3           a = new Double[jumlahDataLatih];
4           Boolean isBreak = false;
5           for (int i = 0; i < jumlahDataLatih; i++) {
6               aStar[i] = 0.0;
7               a[i] = 0.0;
8           }
9           for (int i = 0; i < iterasi; i++) {
10              if (i == 0) {
11                  getValueE();
12                  nilaiD();
13                  lagrangeMultiplier();
14              } else {
15                  getValueE();
16                  nilaiD();
17                  for (int j = 0; j < jumlahDataLatih; j++)
18                  {
```

```
                        if ((Math.abs(Da[j]) < epsilon) &&
                        (Math.abs(DaStar[j]) < epsilon)) {
                            isBreak = true;
                            break;
                        } else {
                            lagrangeMultiplier();
                        }
                    }
                }
                if (isBreak) {
                    break;
                }
            }
}
```

Kode Program 5.8 *Sequential Learning*

Implementasi proses *sequential learning* dilakukan dengan inisialisasi nilai *lagrange multiplier* dengan nilai awal 0 dilanjutkan dengan perulangan sejumlah `iterasi` yang diinputkan oleh user untuk menghitung nilai error dengan memanggil fungsi `getValueE`, nilai $\delta\alpha^*$ dan $\delta\alpha$ dengan memanggil fungsi `nilaiD` dan nilai *lagrange multiplier* yang baru dengan memanggil fungsi `lagrangeMultiplier`.

5.3.3.1 Implementasi Perhitungan Nilai Error

Implementasi perhitungan nilai error sesuai dengan perancangan yang telah dibuat ditunjukkan dalam Kode Program 5.9.

```
private void getValueE() {
        nilaiError = new Double[jumlahDataLatih];
        Double jumlahPerKolom[] = new
        Double[jumlahDataLatih];
        for (int i = 0; i < jumlahDataLatih; i++) {
            for (int j = 0; j < jumlahDataLatih; j++) {
                if (null == jumlahPerKolom[i]) {
                    jumlahPerKolom[i] = 0.0;
                }
                jumlahPerKolom[i] += ((aStar[i] - a[i]) *
                resultHessianLatih[j][i]);
            }
            nilaiError[i] =
            dataLatihNormalisasi[banyakKolom - 1][i] -
            jumlahPerKolom[i];
        }
    }
```

Kode Program 5.9 Perhitungan Nilai Error

Implementasi perhitungan nilai error dilakukan dengan 2 perulangan sejumlah data latih dilanjutkan dengan pengisian array `jumlahPerKolom` dengan menjumlahkan hasil dari pengurangan `aStar` dan `a` dikalikan dengan matriks hessian. Selanjutnya menghitung nilai error dengan data actual dikurangi dengan hasil `jumlahPerKolom` dan disimpan dalam array `nilaiError`.

5.3.3.2 Implementasi Perhitungan Nilai δα* dan δα

Implementasi perhitungan nilai δα* dan δα sesuai dengan perancangan yang telah dibuat ditunjukkan dalam Kode Program 5.10.

```
1   private void nilaiD() {
2           gamma = clr / maksHessian;
3           DaStar = new Double[jumlahDataLatih];
4           Da = new Double[jumlahDataLatih];
5           for (int i = 0; i < jumlahDataLatih; i++) {
6               DaStar[i] = Math.min(Math.max(gamma *
7               (nilaiError[i] - epsilon), -aStar[i]),
8               komplesitas - aStar[i]);
9               Da[i] = Math.min(Math.max(gamma * ((-
10              nilaiError[i]) - epsilon), -a[i]), komplesitas
11              - a[i]);
12          }
13   }
```

Kode Program 5.10 Perhitungan Nilai δα* dan δα

Implementasi perhitungan nilai δα* dan δα dilakukan dengan menghitung nilai gamma dengan membagi `clr` inputan user dengan nilai maksimum matriks hessian. Selanjutnya menghitung nilai δα* dan δα dan hasilnya disimpan dalam array `DaStar` dan `Da`.

5.3.3.3 Implementasi Perhitungan *Lagrange Multiplier*

Implementasi perhitungan nilai *lagrange multiplier* sesuai dengan perancangan yang telah dibuat ditunjukkan dalam Kode Program 5.11.

```
1   private void lagrangeMultiplier() {
2           for (int i = 0; i < jumlahDataLatih; i++) {
3               if (null == aStar[i] || null == a) {
4                   aStar[i] = 0.0;
5                   a[i] = 0.0;
6               }
7               aStar[i] += DaStar[i];
8               a[i] += Da[i];
9           }
10      }
```

Kode Program 5.11 Perhitungan Nilai *Lagrange Multiplier*

Implementasi perhitungan nilai *lagrange multiplier* dilakukan dengan perulangan sejumlah data latih dan menghitung nilai `aStar` ditambah dengan nilai `DaStar` dan nilai `a` ditambah dengan nilai `Da`.

5.3.3.4 Implementasi Perhitungan Nilai *f(x)* Data Latih

Implementasi perhitungan nilai *f(x)* data latih sesuai dengan perancangan yang telah dibuat ditunjukkan dalam Kode Program 5.12

```
1       private void hitungFxLatih() {
2           fxLatih = new Double[jumlahDataLatih];
3           for (int i = 0; i < jumlahDataLatih; i++) {
4               if (null == fxLatih[i]) {
5                   fxLatih[i] = 0.0;
6               }
7               for (int j = 0; j < jumlahDataLatih; j++) {
```

```
                    fxLatih[i] += ((aStar[j] - a[j]) *
                     resultHessianLatih[i][j]);
                }
            }
        }
```

Kode Program 5.12 Perhitungan Nilai f(x) Data Latih

Implementasi perhitungan nilai *f(x)* data latih dilakukan dengan perulangan sebanyak 2 kali sejumlah data latih kemudian mengisi array `fxLatih` dengan penjumlahan nilai `aStar` dikurangi `a` dan kalikan `resultHessianLatih`.

5.3.4 Implementasi Pengujian

Implementasi pengujian dilakukan dengan beberapa proses sesuai dengan perancangan yang telah dilakukan yaitu implementasi perhitungan jarak uji dan latih, implementasi perhitungan matriks hessian data uji, implementasi dan implementasi perhitungan *f(x)* data uji.

5.3.4.1 Implementasi Perhitungan Jarak Data Uji dan Data Latih

Implementasi perhitungan jarak data uji dan data latih dibagi menjadi 2 fungsi yaitu fungsi formula untuk menghitung jarak dan fungsi matriks dari perhitungan jarak. Fungsi formula menghitung jarak data uji dan latih ditunjukkan oleh Kode Program 5.13 dan fungsi matriks perhitungan jarak ditunjukkan oleh Kode Program 5.14.

```
private double formulaCalculateDataLatihUji(Double[][]
dataLatih, Double[][] dataUji, int batasIndexI, int
indexJBase, int indexJ) {
        double[] result = new double[batasIndexI - 1];
        double finalResult = 0;
        for (int i = 0; i < batasIndexI - 1; i++) {
            result[i] = dataLatih[i][indexJBase] -
            dataUji[i][indexJ];
        }
        finalResult = Math.pow(result[0], 2);
        for (int i = 1; i < result.length; i++) {
            finalResult += Math.pow(result[i], 2);
        }
        return finalResult;
    }
```

Kode Program 5.13 Formula Menghitung Jarak Data Uji dan Data Latih

Perhitungan jarak data uji dan data latih dilakukan inisialisasi `result` dan `finalResult` untuk menyimpan hasil perhitungan jarak data uji dan data latih. Perulangan dilakukan sejumlah `batasIndexI` yaitu banyak baris data latih dilanjutkan dengan perhitungan pengurangan untuk data latih dan data uji dan disimpan dalam array `result`. Kemudian mengisi array `finalResult` dengan nilai kuadrat dari `result` dilanjutkan perulangan lagi sejumlah array `result`. Hasil akhir berupa `finalResult` ditambahkan dengan `result` yang dikuadratkan

Adanya matriks jarak data uji dan data latih dalam perhitungan jarak data uji dan data latih digunakan untuk mempermudah perhitungan selanjutnya.

Implementasi fungsi matriks perhitungan jarak data uji dan data latih ditunjukkan oleh Kode Program 5.14.

```
1  private void calculateDistanceUji() {
2          distanceUji = new
3          Double[jumlahDataLatih][jumlahDataUji];
4          for (int i = 0; i < jumlahDataLatih; i++) {
5              for (int j = 0; j < jumlahDataUji; j++) {
6                  distanceUji[i][j] =
7                  formulaCalculateDataLatihUji(dataLatihNormal
8                  isasi, dataUjiNormalisasi, banyakKolom, i,
9                  j);
10             }
11         }
12 }
```

Kode Program 5.14 Matriks Jarak Data Uji dan Latih

Implementasi matriks jarak data uji dan data latih dilakukan dengan inisialisasi array `distanceUji` untuk meyimpan hasil matriks jarak data uji dan data latih. Kemudian dilakukan perulangan sebanyak 2 kali sejumlah data latih dan data uji dilanjutkan dengan mengisi array `distanceUji` dengan nilai dari fungsi `formula`.

5.3.5 Implementasi Perhitungan Matriks Hessian Data Uji

Implementasi perhitungan matriks hessian data uji sesuai dengan perancangan yang telah dibuat ditunjukkan dalam Kode Program 5.15.

```
1  private void calculateHessianUji(double sigma, double
2  lambda, int jumlahData) {
3          resultHessianUji = new
4          Double[jumlahDataLatih][jumlahDataUji];
5          resultHessUjiGui = new
6          Double[jumlahDataUji][jumlahDataLatih];
7          for (int i = 0; i < jumlahDataLatih; i++) {
8              for (int j = 0; j < jumlahDataUji; j++) {
9                  resultHessianUji[i][j] = (Math.exp((-
10                 distanceUji[i][j]) / 2 * (Math.pow(sigma,
11                 2)))) + (Math.pow(lambda, 2));
12             }
13         }
14         for (int i = 0; i < jumlahDataLatih; i++) {
15             for (int j = 0; j < jumlahDataUji; j++) {
16                 resultHessUjiGui[j][i] =
17                resultHessianUji[i][j];
18             }
19         }
20     }
```

Kode Program 5.15 Perhitungan Matriks Hessian Data Uji

Implementasi perhitungan matriks hessian diproses didalam perulangan sebanyak 2 kali. Hasil matriks hessian disimpan didalam array `resultHessianUji` dengan menggunakan formulasi sesuai dengan rumus yang ada. Kemudian untuk dapat menampilkan hasilnya di GUI dilakukan perulangan lagi dan disimpan didalam `resultHessianUjiGui`.

5.3.6 Implementasi Perhitungan Denormalisasi

Implementasi perhitungan denormalisasi dilakukan untuk data uji maupun data latih. Implementasi perhitungan denormalisasi data latih dan data uji ditunjukkan dalam Kode Program 5.16 dan Kode Program 5.17.

```
1   private void denormalisasiDataLatih() {
2           hasilDenormalisasiLatih = new
3           Double[jumlahDataLatih];
4           for (int i = 0; i < jumlahDataLatih; i++) {
5               hasilDenormalisasiLatih[i] = ((fxLatih[i] *
6               (maxFitur[banyakKolom - 1] -
7               minFitur[banyakKolom - 1])) +
8               minFitur[banyakKolom - 1]);
9           }
10  }
```

Kode Program 5.16 Perhitungan Denormalisasi Data Latih

```
1   private void denormalisasiDataUji() {
2           hasilDenormalisasiUji = new Double[jumlahDataUji];
3           for (int i = 0; i < jumlahDataUji; i++) {
4               hasilDenormalisasiUji[i] = ((fxUji[i] *
5               (maxFitur[banyakKolom - 1] -
6               minFitur[banyakKolom - 1])) +
7               minFitur[banyakKolom - 1]);
8           }
9   }
```

Kode Program 5.17 Perhitungan Denormalisasi Data Uji

Implementasi perhitungan denormalisasi data latih dan data uji sama yang membedakan adalah perulangan untuk data latih sejumlah data latih, sedangkan untuk data uji sejumlah data uji. Hasil denormalisasi dihitung sesuai dengan perancangan. Untuk data latih, denormalisasi disimpan pada array `hasilDenormalisasiLatih` sedangkan untuk data uji disimpan dalam array `hasilDenormalisasiUji`.

5.3.7 Implementasi Perhitungan MAPE

Implementasi perhitungan MAPE dilakukan untuk data uji maupun data latih. Implementasi perhitungan MAPE data latih dan data uji ditunjukkan dalam Kode Program 5.18 dan Kode Program 5.19.

```
1   private void calculateMapeLatih() {
2           mapeLatih = 0;
3           double result = 0;
4           for (int i = 0; i < jumlahDataLatih; i++) {
5               result += (Math.abs((dataLatih[banyakKolom -
6               1][i]) - hasilDenormalisasiLatih[i]) /
7               (dataLatih[banyakKolom - 1][i]));
8           }
9           mapeLatih = (result / jumlahDataLatih) * 100;
10  }
```

Kode Program 5.18 Perhitungan MAPE Data Latih

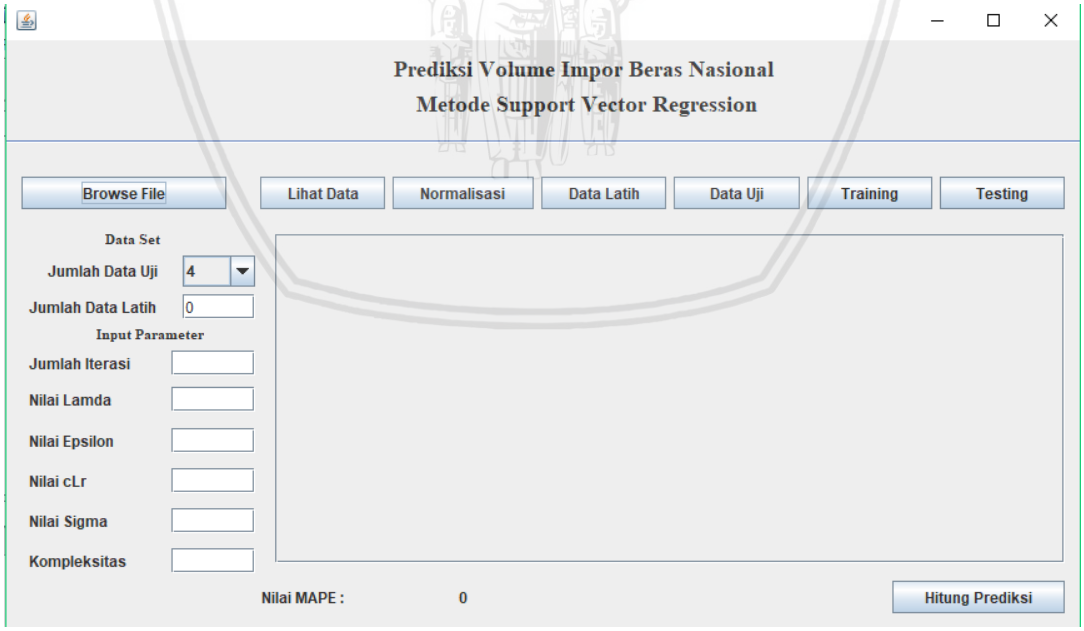
```
1   private void calculateMapeUji() {
2           mapeUji = 0;
3           double result = 0;
4           for (int i = 0; i < jumlahDataUji; i++) {
5               result += (Math.abs((dataUji[banyakKolom -
                1][i] - hasilDenormalisasiUji[i])) /
                (dataUji[banyakKolom - 1][i]));
6           }
7           mapeUji = (result / jumlahDataUji) * 100;
8       }
```

Kode Program 5.19 Perhitungan MAPE Data Uji

Implementasi perhitungan nilai MAPE data latih dan data uji sama yang membedakan adalah perulangan untuk data latih sejumlah data latih, sedangkan untuk data uji sejumlah data uji. Hasil denormalisasi dihitung sesuai dengan perancangan dan disimpan dalam variable `result`. Hasil akhirnya setelah dijadikan presentase, MAPE data latih disimpan pada array `mapeLatih` sedangkan untuk data uji disimpan dalam array `mapeUji`.

5.4 Implementasi Antarmuka

Implementasi antarmuka dilakukan sesuai dengan perancangan antarmuka yang telah dibuat sebelumnya. *Button browse* digunakan untuk mengimpor data dalam bentuk excell akan diproses dalam sistem. Beberapa *button* seperti Lihat Data, Normalisasi, Data Latih, Data Uji, Training da Testing merupakan *button* yang digunakan untuk menampilkan hasil kedalam tabel yang ada. Implementasi Antarmuka ditunjukan pada Gambar 5.1.



Gambar 5.1 Implementasi Antarmuka

Button Lihat Data digunakan untuk menampilkan data yang telah di impor dalam bentuk excel sebelumnya. *Button* Normalisasi digunakan untuk menampilkan hasil normalisasi seluruh data yang telah di impor ke dalam *table* yang ada. *Button* Data Latih dan Data Uji digunakan untuk menampilkan data yang

digunakan sebagai data latih maupun data uji yang telah dinormalisasi kedalam tabel. Banyak nya dapat diatur dengan *dropdown* jumlah data uji untuk data uji dan menentukan banyaknya data latih dengan mengisi *textfield* data latih yang ada. *Button* training dan testing digunakan untuk menampilkan hasil regresi, denormalisasi dan data aktual dari data latih yang telah melewati proses *sequential learning* dan data uji. Parameter SVR memelurkan input dari user untuk dapat menjalankan program komputasi. *Button* Hitung Prediksi digunakan untuk mendirect ke halaman perhitungan prediksi.

BAB 6 PENGUJIAN DAN ANALISIS

Bab ini akan membahas hasil pengujian sistem dan analisis yang telah dilakukan. Pengujian yang dilakukan sesuai dengan perancangan pada bab 3 yaitu pengujian nilai parameter SVR, jumlah iterasi, dan variasi data latih.

6.1 Pengujian Nilai Parameter SVR

Pengujian nilai parameter digunakan untuk mengetahui parameter parameter yang menghasilkan nilai MAPE terbaik dan mengetahui pengaruh nilai dari masing masing parameter yang diuji.

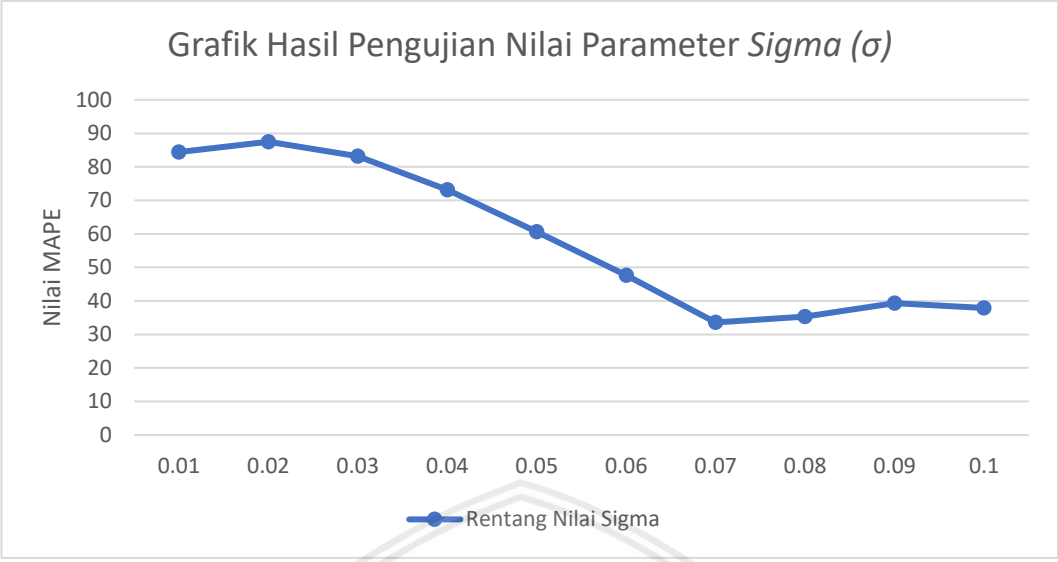
6.1.1 Pengujian Nilai Parameter *Sigma* (σ)

Skenario pengujian yang dilakukan untuk menguji nilai parameter *sigma* sesuai dengan perancangan pengujian yang telah dibuat. Dilakukan dengan mengubah ubah nilai dari parameter *sigma* sesuai dengan nilai *sigma* yang telah ditetapkan. Pada pengujian nilai *sigma* ini, inisialisasi parameter lain yang digunakan adalah nilai *lambda* = 0.1, nilai *epsilon* = 0.00001, nilai clr = 0.01, nilai *C* = 10 dan iterasi = 2000. Hasil pengujian nilai parameter sigma ditunjukkan dalam Tabel 6.1.

Tabel 6.1 Hasil Pengujian Rentang Nilai Parameter *Sigma (σ)*

No.	Nilai *Sigma*	Nilai MAPE
1	0.01	84.3864
2	0.02	87.4945
3	0.03	83.1370
4	0.04	73.1471
5	0.05	60.6033
6	0.06	47.6423
7	0.07	33.5849
8	0.08	35.3380
9	0.09	39.3312
10	0.1	37.9404

Hasil pengujian nilai parameter *sigma* dalam bentuk grafik disajikan untuk mengetahui pengaruh nilai *sigma* terhadap MAPE. Grafik hasil pengujian nilai parameter *sigma* ditunjukkan oleh Gambar 6.1.



Gambar 6.1 Grafik Hasil Pengujian Nilai Parameter *Sigma (σ)*

Hasil pengujian parameter nilai *sigma* sesuai dengan Gambar 6.1 menunjukkan nilai MAPE yang naik dan turun. Nilai *sigma* merupakan nilai yang mempengaruhi persebaran data pada proses learning (Furi, Jondri & Saepudin, 2015). Nilai *sigma* yang kecil membuat persebaran data yang tidak sesuai sehingga meningkatkan nilai *error rate.* Nilai *sigma* yang semakin besar menghasilkan nilai error rate yang semakin kecil karena persebaran yang dilakukan akan semakin merata ini dibuktikan dengan tren penurunan nilai MAPE ketika berada di rentang 0.02 hingga 0.07. Namun nilai *sigma* yang terlalu tinggi akan kembali menaikkan *error rate* seperti ketika nilai *sigma* sebesar 0.08 hingga 0,09. Nilai MAPE terbaik pada pengujian didapat ketika *sigma* bernilai 0.07.

6.1.2 Pengujian Nilai Parameter *Lambda (λ)*

Skenario pengujian yang dilakukan untuk menguji nilai parameter *lambda* sesuai dengan perancangan pengujian yang telah dibuat adalah dengan mengubah ubah nilai dari parameter *lambda* sesuai dengan nilai *lambda* yang telah ditetapkan. Pada pengujian nilai *sigma* ini, inisialisasi parameter lain yang digunakan adalah nilai *sigma* = 0.07, nilai *epsilon* = 0.00001, nilai clr = 0.01, nilai *C* = 10 dan iterasi = 2000. Hasil pengujian nilai *lambda* ditunjukkan dalam Tabel 6.2 dan Tabel 6.3.

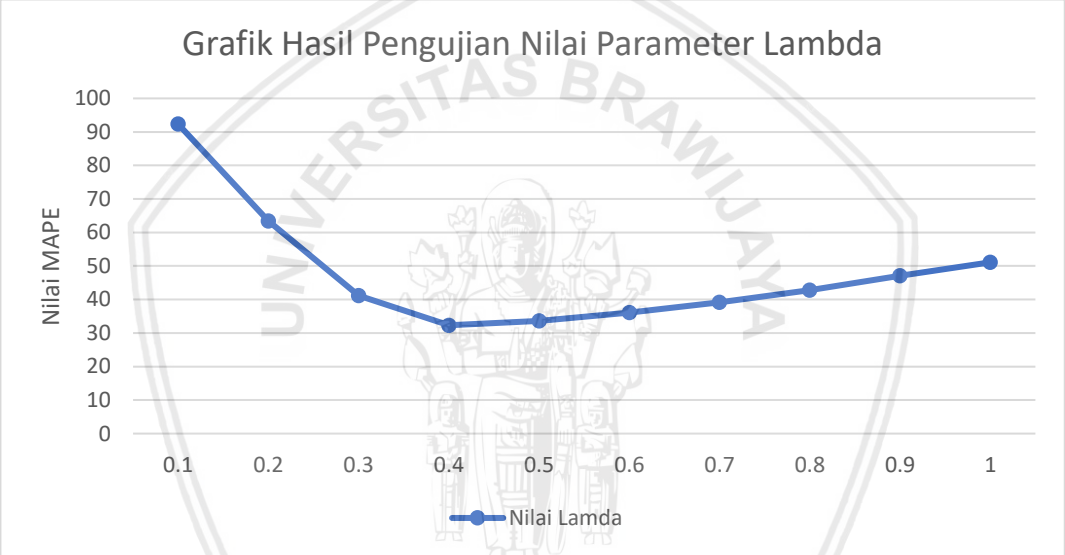
Tabel 6.2 Hasil Pengujian Nilai Parameter *Lambda (λ)* Bagian 1

No.	Nilai *Lambda*	Nilai MAPE
1	0.1	92.3131
2	0.2	63.3923
3	0.3	41.1565
4	0.4	32.3211
5	0.5	33.5849

Tabel 6.3 Hasil Pengujian Nilai Parameter *Lambda (λ)* Bagian 2

No.	Nilai *Lambda*	Nilai MAPE
6	0.6	36.1349
7	0.7	39.1272
8	0.8	42.7764
9	0.9	47.0696
10	1	51.1222

Hasil pengujian nilai parameter *lambda* akan disajikan dalam bentuk grafik untuk mengetahui pengaruh rentang nilai *lambda* terhadap MAPE. Grafik hasil pengujian rentang nilai parameter *lambda* ditunjukkan oleh Gambar 6.2.



Gambar 6.2 Grafik Hasil Pengujian Nilai Parameter *Lambda*

Hasil pengujian rentang nilai *lambda* sesuai dengan Gambar 6.2 menunjukkan nilai MAPE mengalami penurunan dan kenaikan. Nilai *lambda* merupakan ukuran yang meyatakan besaranya penskalaan ruang pemetaan pada kernel (Vijayakumar and Wu, 1999). Nilai *lambda* yang terlalu kecil membuat pemetaan pada kernel tidak seimbang sehingga *error rate* yang dihasilkan akan tinggi sesuai dengan grafik di angka 0.1 yang menghasilkan *error rate* yang sangat tinggi sebesar 92.313. Nilai *lambda* di angka 0.1 hingga 0.4 menunjukkan tren penurunan dan kembali menunjukkan tren kenaikan di angka 0.5 hingga 1. Ini menunjukkan bahwa nilai *lambda* yang terlalu besar juga membuat nilai *error rate* akan meningkat. Jadi besar nilai *lambda* yang digunakan harus sesuai atau tidak terlalu tinggi maupun rendah. Pada pengujian yang dilakukan nilai *lambda* terbaik adalah 0.4.

6.1.3 Pengujian Nilai cLR (*Constanta Learning Rate*)

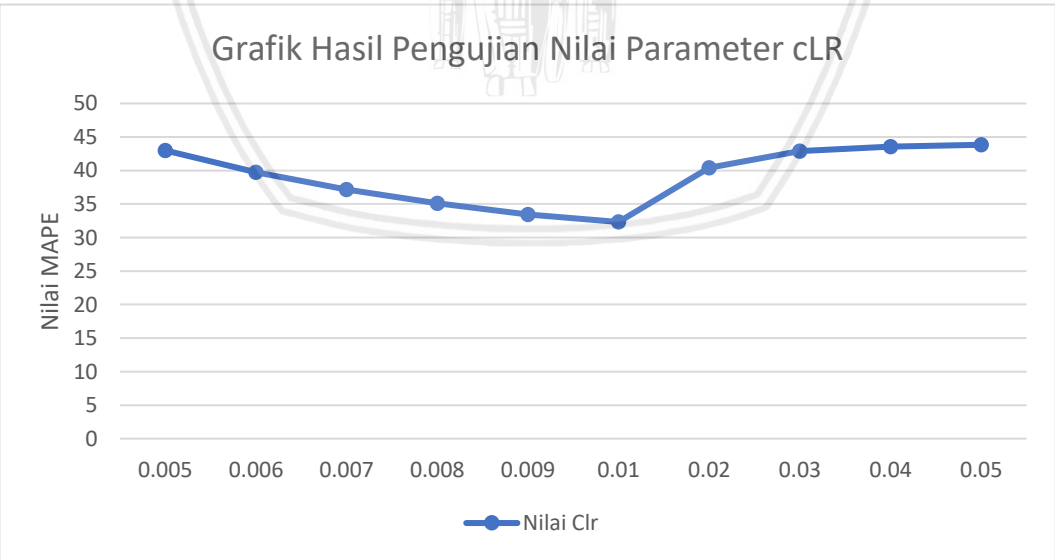
Skenario pengujian yang dilakukan untuk menguji nilai parameter cLR sesuai dengan perancangan pengujian yang telah dibuat adalah dengan mengubah ubah

nilai dari parameter cLR sesuai dengan nilai yang telah ditetapkan. Pada pengujian nilai sigma ini, inisialisasi parameter lain yang digunakan adalah nilai *sigma* = 0.07, nilai *lambda* = 0.4, nilai *epsilon* = 0.00001, nilai *C* = 10 dan iterasi = 2000. Hasil pengujian nilai parameter Clr ditunjukkan dalam Tabel 6.4.

Tabel 6.4 Hasil Pengujian Nilai Parameter cLR

No.	Nilai cLR	Nilai MAPE
1	0.005	43.0144
2	0.006	39.7227
3	0.007	37.1477
4	0.008	35.1027
5	0.009	33.4450
6	0.01	32.3201
7	0.02	40.3946
8	0.03	42.9141
9	0.04	43.5735
10	0.05	43.8403

Hasil pengujian nilai parameter cLR akan disajikan dalam bentuk grafik untuk mengetahui pengaruh rentang nilai cLR terhadap MAPE. Grafik hasil pengujian nilai parameter cLR ditunjukkan oleh Gambar 6.3.



Gambar 6.3 Grafik Hasil Pengujian Nilai Parameter cLR

Hasil pengujian rentang nilai cLR sesuai dengan Gambar 6.3 menunjukkan nilai MAPE yang mengalami penurunan dan kenaikan. Penurunan terjadi ketika cLR bernilai 0.005 hingga 0.01 dan mengalami kenaikan ketika bernilai 0.02. Nilai cLR digunakan dalam menentukan nilai gamma yang berfungsi sebagai pengatur

dari laju *learning* system. Semakin besar nilai cLR maka semakin kecil nilai MAPE yang dihasilkan namun bila nilai cLR terlalu besar angkanya, *error rate* yang dihasilkan akan meningkat. Dalam proses *learning* yang dilakukan, nilai cLR berpengaruh terhadap nilai *alpha* dan *alpha star* secara tidak langsung. Nilai *alpha* dan *alpha star* dipengaruhi oleh nilai gamma dimana perhitungan nilainya menggunakan nilai cLR. Nilai cLR yang pas akan menghasilkan nilai gamma yang pas pula sehingga *alpha* dan *alpha star* yang dihasilkan akan membuat hasil prediksi memiliki *error rate* yang kecil. Nilai parameter cLR terbaik yang diperoleh dalam pengujian ini adalah 0.01.

6.1.4 Pengujian Nilai Parameter Kompleksitas (*C*)

Skenario pengujian yang dilakukan untuk menguji nilai parameter *C* sesuai dengan perancangan pengujian yang telah dibuat adalah dengan mengubah ubah nilai dari parameter *C* sesuai dengan nilai yang telah ditetapkan. Pada pengujian nilai *C* ini, inisialisasi parameter lain yang digunakan adalah nilai *sigma* = 0.07, nilai *lambda* = 0.4, nilai *epsilon* = 0.00001, nilai cLR = 0.1 dan iterasi = 2000. Hasil pengujian nilai parameter *C* ditunjukkan dalam

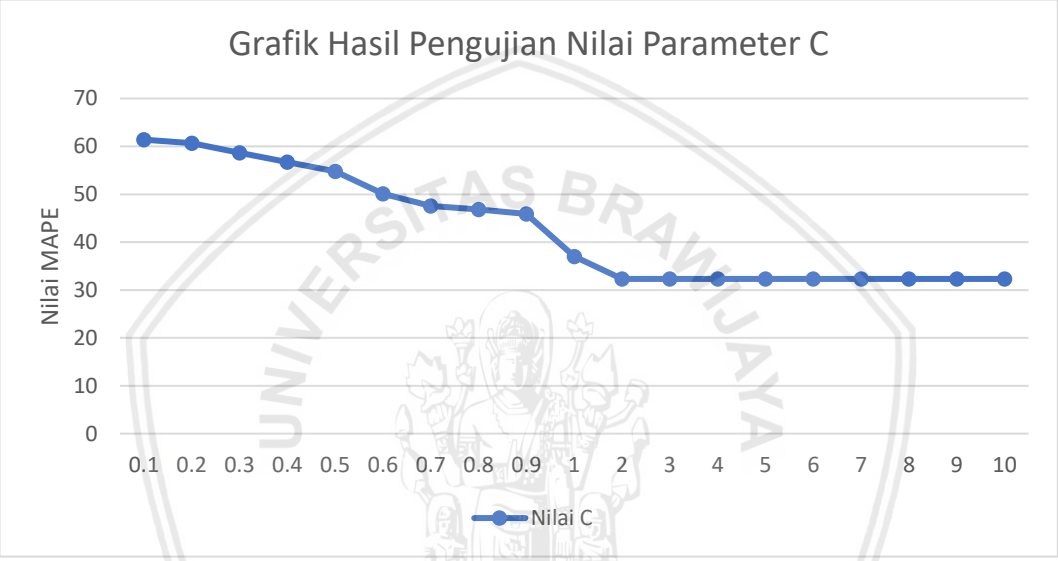
Tabel 6.5 Hasil Pengujian Nilai Parameter Kompleksitas (*C*) bagian 1

No.	Nilai *C*	Nilai MAPE
1	0.1	61.4004
2	0.2	60.6703
3	0.3	58.6840
4	0.4	56.7235
5	0.5	54.7592
6	0.6	50.0761
7	0.7	47.5351
8	0.8	46.8258
9	0.9	45.8969
10	1	37.0224
11	2	32.3202
12	3	32.3202
13	4	32.3202
14	5	32.3202
15	6	32.3202
16	7	32.3202

Tabel 6.6 Hasil Pengujian Nilai Parameter Kompleksitas (*C*) bagian 2

No.	Nilai *C*	Nilai MAPE
17	8	32.3202
18	9	32.3202
19	10	32.3202

Hasil pengujian nilai parameter *C* akan disajikan dalam bentuk grafik untuk mengetahui pengaruh nilai *C* terhadap MAPE. Grafik hasil pengujian rentang nilai parameter *C* ditunjukkan oleh Gambar 6.4



Gambar 6.4 Grafik Hasil Pengujian Nilai Parameter Kompleksitas (*C*)

Hasil pengujian nilai *C* berdasarkan Gambar 6.4 menunjukkan nilai MAPE yang menurun ketika *C* bernilai 0.1 hingga 1 kemudian cenderung konstan ketika *C* bernilai 1 hingga 10. Nilai *C* merupakan konstanta regulasi yang menentukan nilai penalti terhadap fungsi objektif (Basak, Pal & Patranabis, 2007). Nilai *C* yang semakin besar menunjukkan bahwa model yang digunakan semakin tidak menolerir kesalahan yang dilakukan system ketika proses *learning* sehingga nilai *error rate* yang dihasilkan akan semakin kecil dan cenderung konstan sesuai dengan grafik ketika nilai *C* sebesar 0.9 hingga 10. Pada pengujian nilai *C* didapatkan nilai *C* terbaik yaitu sebesar 2 hingga 10.

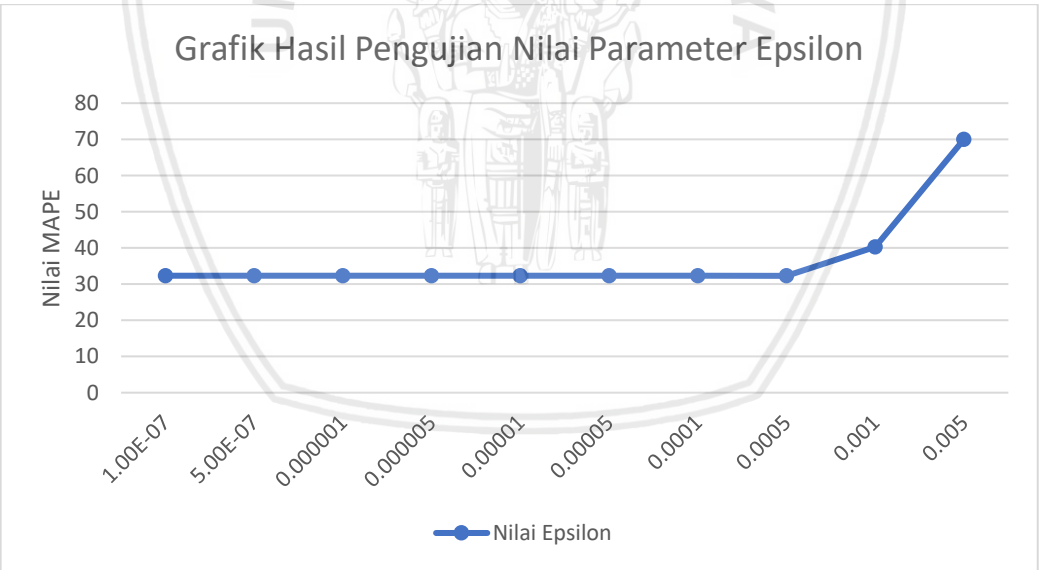
6.1.5 Pengujian Nilai Parameter *Epsilon (ε)*

Skenario pengujian yang dilakukan untuk menguji nilai parameter *epsilon* sesuai dengan perancangan pengujian yang telah dibuat adalah dengan mengubah ubah nilai dari parameter *epsilon* sesuai dengan nilai yang telah ditetapkan. Pada pengujian nilai *epsilon* ini, inisialisasi parameter lain yang digunakan adalah nilai *sigma* = 0.07, nilai *lambda* = 0.4, nilai cLR = 0.1, nilai *C* = 10 dan iterasi = 2000. Hasil pengujian rentang nilai parameter epsilon ditunjukkan dalam Tabel 6.7.

Tabel 6.7 Hasil Pengujian Nilai Parameter *Epsilon (ε)*

No.	Nilai *Epsilon*	Nilai MAPE
1	1E-07	32.3217
2	5E-07	32.3212
3	0.000001	32.3211
4	0.000005	32.3207
5	0.00001	32.3201
6	0.00005	32.3155
7	0.0001	32.3097
8	0.0005	32.2779
9	0.001	40.3102
10	0.005	70.0168

Hasil pengujian rentang nilai parameter *epsilon* disajikan dalam bentuk grafik untuk mengetahui pengaruh rentang nilai *epsilon* terhanap MAPE. Grafik hasil pengujian rentang nilai parameter *epsilon* ditunjukkan oleh Gambar 6.5



Gambar 6.5 Grafik Hasil Pengujian Rentang Nilai *Epsilon*

Hasil pengujian rentang nilai *epsilon* sesuai dengan Gambar 6.5 menunjukkan nilai MAPE yang cenderung konstan dan mengalami kenaikan seiring dengan semakin besar nilai *epsilon*. Nilai *epsilon* merupakan nilai yang menunjukkan tingkat ketelitian dalam model regresi yang digunakan. Semakin kecil angka *epsilon* yang digunakan maka tingkat ketelitian dalam model regresi yang digunakan semakin besar pula. Dibuktikan dengan grafik yang menunjukkan tren konstan ketika berada di rentang 1E-07 hingga 0.0005. Sebaliknya, nilai *epsilon* yang terlalu besar membuat *error rate* yang dihasilkan akan semakin

meningkat karena model regresi yang dihasilkan menjadi semakin tidak telitti sesuai dengan grafik ketika *epsilon* berada di nilai 0.001 hingga 0.005. Dalam pengujian ini didapatkan nilai *epsilon* terbaik yaitu 0.0004.

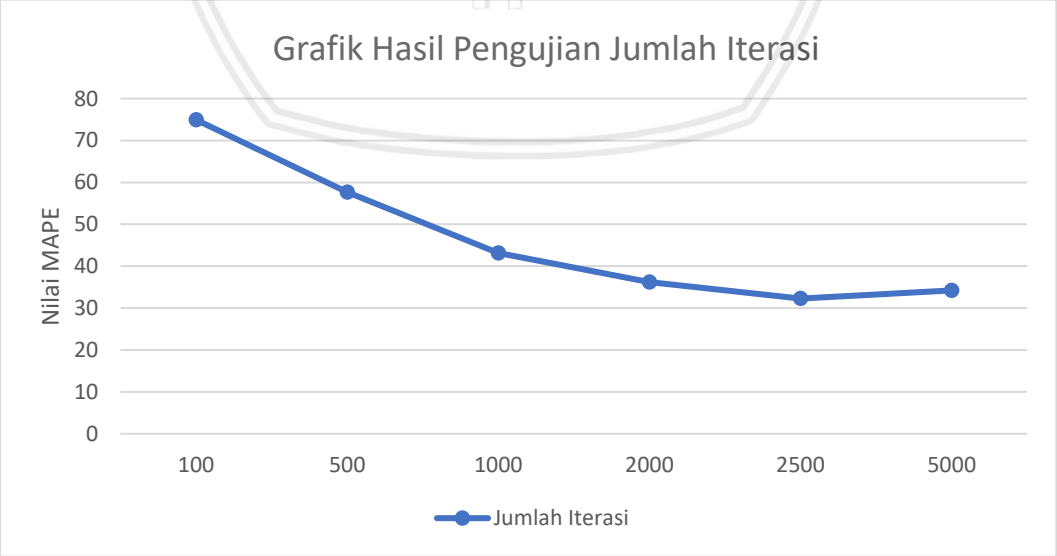
6.2 Pengujian Jumlah Iterasi

Skenario pengujian yang dilakukan untuk menguji jumlah iterasi sesuai dengan perancangan pengujian yang telah dibuat adalah dengan mengubah ubah jumlah iterasi sesuai dengan jumlah yang telah ditetapkan. Pada pengujian jumlah iterasi ini, inisialisasi parameter lain yang digunakan adalah nilai *sigma* = 0.07, nilai *lambda* = 0.4, nilai cLR = 0.1, nilai *C* = 10 dan nilai *epsilon* = 0.0004. Hasil pengujian jumlah iterasi ditunjukkan dalam Tabel 6.8

Tabel 6.8 Hasil Pengujian Jumlah Iterasi

Jumlah iterasi	Nilai MAPE
100	74.9407
500	57.6583
1000	43.1189
2000	36.1908
2500	32.2748
5000	34.2349

Hasil pengujian jumlah iterasi akan disajikan dalam bentuk grafik untuk mengetahui pengaruh jumlah iterasi terhadap nilai MAPE. Grafik hasil pengujian jumlah iterasi ditunjukkan oleh Gambar 6.6



Gambar 6.6 Grafik Hasil Pengujian Jumlah Iterasi

Hasil pengujian jumlah iterasi sesuai dengan Gambar 6.6 menunjukkan nilai MAPE yang cenderung mengalami penurunan. Penurunan terjadi ketika iterasi berjumlah 500 hingga 2500, dan kembali mengalami kenaikan ketika iterasi bernilai 5000. Ini membuktikan bahwa semakin banyak jumlah iterasi maka semakin kecil nilai MAPE yang didapatkan, karena bila jumlah iterasi yang dilakukan semakin besar maka kemampuan observasi sistem terhadap pola data akan semakin bagus juga. Namun bila jumlah iterasi yang dilakukan terlalu besar akan meningkatkan nilai error rate karena observasi terhadap pola data akan tidak stabil. Bila dilihat dari grafik hasil pengujian jumlah iterasi ketika iterasi mencapai angka 5000 justru nilai MAPE mengalami kenaikan. Dalam penelitian ini jumlah iterasi terbaik yang didapatkan adalah sebanyak 2000 iterasi.

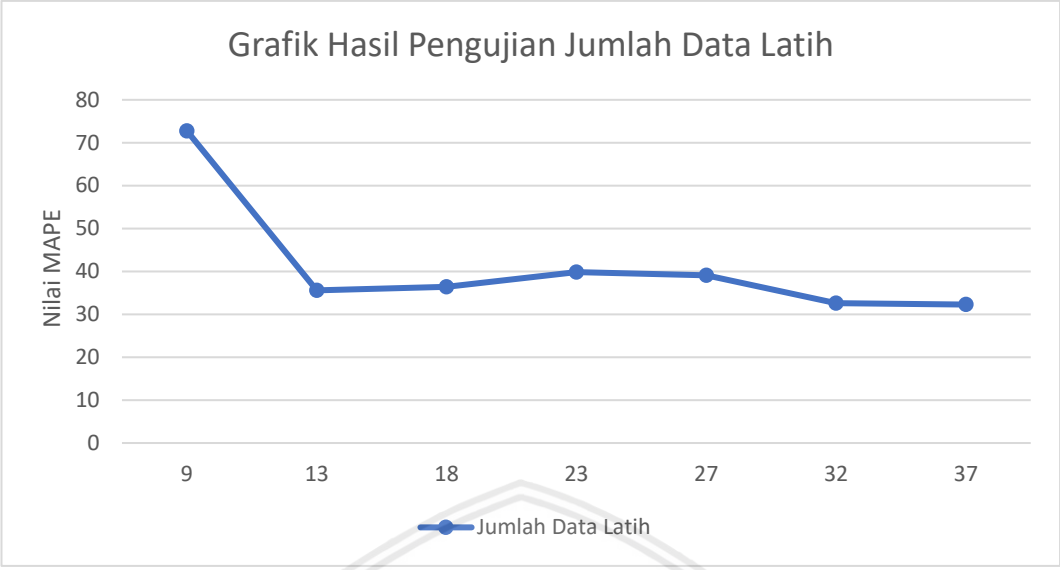
6.3 Pengujian Variasi Jumlah Data Latih

Skenario pengujian yang dilakukan digunakan untuk menguji jumlah data latih sesuai dengan perancangan pengujian yang telah dibuat adalah dengan mengubah ubah jumlah data sesuai dengan jumlah yang telah ditetapkan. Pada pengujian variasi jumlah data latih ini, inisialisasi parameter lain yang digunakan adalah nilai *sigma* = 0.07, nilai *lambda* = 0.4, nilai cLR = 0.1, nilai *C* = 10, nilai *epsilon* = 0.0004 dan jumlah iterasi = 2000. Hasil pengujian variasi jumlah data latih disajikan dalam Tabel 6.9.

Tabel 6.9 Hasil Pengujian Variasi Jumlah Data Latih

Jumlah Data Latih	Jumlah Data Uji	Nilai MAPE
9	9	72.7575
13	9	35.6126
18	9	36.3825
23	9	39.8142
27	9	39.1185
32	9	32.5775
37	9	32.2748

Hasil pengujian variasi jumlah data latih akan disajikan dalam bentuk grafik untuk mengetahui pengaruh variasi jumlah data latih terhadap nilai MAPE. Grafik hasil pengujian variasi jumlah data latih ditunjukkan oleh Gambar 6.7.



Gambar 6.7 Grafik Hasil Pengujian Variasi Data Latih

Hasil pengujian variasi data latih sesuai dengan Gambar 6.7 menunjukkan nilai MAPE yang mengalami penurunan ketika data latih bernilai 13 dan mengalami kenaikan yang tidak signifikan di angka 18 dan 23 kemudian kembali mengalami penurunan di angka 32 dan 37. Dapat dilihat dari grafik bahwa semakin banyak jumlah data latih maka semakin kecil nilai MAPE yang dihasilkan meskipun terjadi peningkatan yang tidak signifikan pada nilai MAPE diangka 18 dan 13. Secara umum jumlah data latih yang semakin banyak digunakan dalam proses *training* akan menjadikan sistem lebih banyak melakukan proses *learning* yang akan meningkatkan nilai akurasi pada system. Namun, jumlah data latih yang benyak juga tidak menjamin *error rate* yang dihasilkan akan turun. Hal ini disebabkan oleh data latih yang digunakan banyak yang nilai nya terlalu tinggi ataupun terlalu rendah yang justru membuat nilai *error rate* nya meningkat. Hal ini ditunjukkan ketika variasi data latih sebanyak 23 dan 27. Pengujian variasi jumlah data latih yang dilakukan didapatkan jumlah data latih terbaik yaitu sebesar 37 data latih.

6.4 Analisis Pembahasan

Setelah didapatkan parameter terbaik dari hasil pengujian, dilakukan pengujian kembali menggunakan parameter terbaik yang telah didapatkan. Adapun nilai parameter yang digunakan adalah *Sigma* = 0.07, *Lambda* = 0.4, cLr = 0.01, *C* = 10, *Epsilon* = 0.0004, Iterasi = 2000, dan data latih = 37. Perbandingan hasil prediksi menggunakan parameter terbaik dengan data aktual disajikan dalam Tabel 6.10.

Tabel 6.10 Perbandingan Hasil Prediksi dengan Data Aktual

Tahun	Prediksi	Aktual
2008	288.8654	289.274
2009	274.6315	250.276

Tabel 6.11 Perbandingan Hasil Prediksi dengan Data Aktual bagian 2

Tahun	Prediksi	Aktual
2010	584.7153	687.583
2011	600.4956	2744.261
2012	649.1006	1927.563
2013	622.1802	472.665
2014	622.1864	815.285
2015	622.1864	816.63
2016	622.0207	1073.72

Hasil pengujian dengan parameter terbaik menghasilkan nilai MAPE dengan kategori cukup yaitu sebesar 32.2748. Salah satu yang penyebab nilai MAPE hasil penelitian tergolong dalam kategori cukup adalah variabel bebas yang digunakan kemungkinan tidak berpengaruh terhadap variabel terikat. Untuk mengetahui apakah variabel bebas (konsumsi, produksi, impor tahun sebelumya) memiliki pengaruh atau tidak terhadap variabel terikat, penulis melakukan analisis korelasi.

Analisis korelasi yang dilakukan menggunakan metode product moment yang dicetuskan oleh *Karl Pearson*. Metode ini dipilih karena sesuai dengan data yang digunakan yaitu data kontinum atau data yang didapat dari hasil pengukuran dan termasuk jenis data rasio (Sugiyono, 2015). Dalam mengambil keputusan analisis korelasi menggunakan SPSS terdapat 3 pedoman mengambilan keputusan yaitu berdasarkan pada nilai r (*pearson correlations*). Nilai r menunjukkan arah dan hubungan korelasi antar variabel. Selanjutnya pengambilan keputusan nilai signifikansi Sig. (2-tailed) dengan membandingkan nilai signifikansi yang dihasilkan dengan signifikansi yang digunakan dalam analisis (0.05/0.01). Jika nilai Sig. (2-tailed) > 0.05/0.01 maka tidak ada hubungan yang signifikan antara kedua variabel. Yang terakhir adalah pengambilan keputusan berdasarkan tanda bintang yang ada dalam *output* SPSS. Bila terdapat tanda bintang (* atau **) pada *pearson correlation* maka dapat disimpulkan bahwa variabel mempunyai hubungan dengan variabel lain. Intepretasi koefisien korelasi (Sugiyono, 2015) dapat dilihat pada Tabel 6.12.

Tabel 6.12 Intepretasi terhadap Koefisien Korelasi (r)

Interval Koefisien *Pearson* (r)	Tingkat Hubungan
0.00-0.199	Sangat Rendah
0.20-0.399	Rendah
0.40-0.599	Sedang
0.60-0.799	Kuat
0.80-1.000	Sangat Kuat

6.4.1 Analisis Korelasi Variabel Konsumsi dengan Variabel Volume Impor Beras

Hasil analisis korelasi product moment variabel konsumsi dan volume impor beras (target) menggunakan aplikasi SPSS dapat dilihat dalam Gambar 6.8.

Correlations

		kosumsi	target
kosumsi	Pearson Correlation	1	.000
	Sig. (2-tailed)		.998
	N	46	46
target	Pearson Correlation	.000	1
	Sig. (2-tailed)	.998	
	N	46	46

Gambar 6.8 Hasil Analisis Korelasi Variabel Konsumsi dengan Variabel Volume Impor Beras

Berdasarkan hasil analisis korelasi yang dilakukan menunjukkan nilai *pearson correlation* sebesar 0. Nilai signifikansi Sig. (2-tailed) bernilai 0.998. Karena 0.998 > 0.05 maka tidak ada hubungan yang signifikan antar kedua variabel. Sehingga dari kedua dasar pengambilan keputusan analisis korelasi pearson dapat disimpulkan bahwa tidak ada hubungan antara variabel konsumsi dan variabel impor beras.

6.4.2 Analisis Korelasi Variabel Produksi dengan Variabel Volume Impor Beras

Hasil analisis korelasi product moment variabel produksi dan volume impor beras (target) menggunakan aplikasi SPSS dapat dilihat dalam Gambar 6.9.

Correlations

		produksi	target
produksi	Pearson Correlation	1	-.020
	Sig. (2-tailed)		.897
	N	46	46
target	Pearson Correlation	-.020	1
	Sig. (2-tailed)	.897	
	N	46	46

Gambar 6.9 Hasil Analisis Korelasi Variabel Produksi dengan Variabel Volume Impor Beras

Berdasarkan hasil analisis korelasi yang dilakukan menunjukkan nilai *pearson correlation* sebesar -0.020 yang berarti hubungan antar variabel sangat rendah dan negatif. Nilai signifikansi Sig. (2-tailed) bernilai 0.897. Karena 0.897 > 0.05 maka tidak ada hubungan yang signifikan antara kedua variabel. Sehingga dari kedua dasar pengambilan keputusan analisis korelasi pearson dapat disimpulkan bahwa hubungan antara variabel konsumsi dan variabel impor beras sangat lemah dan tidak signifikan.

6.4.3 Analisis Korelasi Variabel Volume Impor 1 Tahun Sebelumnya dengan Variabel Volume Impor Beras

Hasil analisis korelasi product moment variabel volume impor 1 tahun sebelumnya (impor) dan volume impor beras (target) menggunakan aplikasi SPSS dapat dilihat dalam Gambar 6.10.

Correlations

		impor sebelum	target
impor sebelum	Pearson Correlation	1	.433**
	Sig. (2-tailed)		.003
	N	46	46
target	Pearson Correlation	.433**	1
	Sig. (2-tailed)	.003	
	N	46	46

**. Correlation is significant at the 0.01 level (2-tailed).

Gambar 6.10 Hasil Analisis Korelasi Variabel Volume Impor 1 Tahun Sebelumnya dengan Variabel Volume Impor Beras

Berdasarkan hasil analisis korelasi yang dilakukan menunjukkan nilai *pearson correlation* sebesar 0.433 yang berarti hubungan antar variabel sedang dan positif. Nilai signifikansi Sig. (2-tailed) bernilai 0.003. Karena 0.003 < 0.01 maka ada hubungan yang signifikan antara kedua variabel. Adanya tanda bintang pada *pearson correlation* menunjukkan hubungan secara nyata antar kedua variabel. Sehingga dari ketiga dasar pengambilan keputusan analisis korelasi pearson dapat disimpulkan bahwa ada hubungan yang nyata antara variabel volume impor 1 tahun sebelumnya dan variabel impor beras dengan tingkat hubungan yang sedang dan signifikan.

BAB 7 PENUTUP

7.1 Kesimpulan

Kesimpulan yang dapat diambil berdasarkan penelitian yang telah dilakukan adalah:

1. Parameter-parameter metode SVR mempengaruhi hasil prediksi yang dilakukan dalam penelitian. Parameter *sigma* yang semakin tinggi cenderung membuat hasil prediksi yang dilakukan semakin baik. Begitu pula parameter *C*, semakin tinggi angkanya cenderung semakin baik dan memberikan hasil evaluasi yang konstan. Parameter lain seperti *lambda*, cLr dan *epsilon* yang tinggi juga memberikan nilai evaluasi yang baik, namun akan mendapatkan nilai evaluasi yang semakin buruk bila angka yang digunakan teralu tinggi.
2. Evaluasi yang dihasilkan dari penelitian prediksi volume impor beras nasional menghasilkan nilai MAPE terbaik sebesar 32.2748, ini termasuk dalam kategori cukup. Adapun parameter terbaik yang digunakan dalam menghasilkan nilai MAPE terbaik adalah *Sigma* = 0.07, *Lambda* = 0.4, cLr = 0.01, *C* = 10, *Epsilon* = 0.0004, Iterasi = 2000, dan data latih = 37. Nilai MAPE yang termasuk dalam kategori cukup membuktikan bahwa metode *Support Vector Regression* dengan variabel konsumsi, produksi dan volume impor beras 1 tahun sebelumnya dapat digunakan dalam memprediksi volume impor beras nasional.
3. Analisis korelasi yang dilakukan terhadap variabel bebas dan terikat menghasilkan kesimpulan bahwa variabel bebas konsumsi beras yang digunakan dalam penelitian tidak berkorelasi terhadap variabel volume impor beras. Variabel produksi beras berkorelasi dengan tingkat hubungan sangat rendah dan tidak signifikan terhadap variabel impor beras. Sedangkan variabel impor beras berkorelasi dengan tingkat hubungan sedang dan signifikan dengan volume impor beras 1 tahun sebelumnya

7.2 Saran

Adapun saran dari penulis untuk penelitian selanjutnya adalah:

1. Menambahkan variabel bebas yang digunakan dalam penelitian. Variabel yang sekiranya berpengaruh terhadap volume impor beras nasional. Dapat berasal dari lingkup ekonomi, sosial dan lainnya.
2. Karena hasil yang didapatkan tergolong cukup, maka disarankan menggunakan optimasi agar parameter yang didapat lebih optimal sehingga hasil evaluasi yang diperoleh juga lebih maksimal.

DAFTAR REFERENSI

Basak, D., Pal, S. and Patranabis, D. C. (2007) *Support Vector Regression*.

Caraka, R. E., Yasin, H. and Basyiruddin, A. W. (2017) 'Peramalan Crude Palm Oil (CPO) Menggunakan Support Vector Regression Kernel Radial Basis', *Jurnal Matematika*, 7(1), p. 43. doi: 10.24843/JMAT.2017.v07.i01.p81.

Chang, P., Wang, Y. and Liu, C. (2007) 'The development of a weighted evolving fuzzy neural network for PCB sales forecasting', 32, pp. 86–96. doi: 10.1016/j.eswa.2005.11.021.

Chen, W. U. *et al.* (2013) 'Research on Parameter Selection of Support Vector Regression', 344, pp. 219–225. doi: 10.4028/www.scientific.net/AMM.344.219.

Christianto, E. (2013) 'FAKTOR YANG MEMENGARUHI VOLUME IMPOR BERAS DI INDONESIA', 7(2), pp. 38–43.

Dewi, C. and Himawati, W. W. (2015) 'Prediksi Tingkat Pengangguran Menggunakan Adaptif Neuro Fuzzy Inference System (ANFIS)', (2004), pp. 9–10.

Febrianty, H. (2013) *ANALISIS PERKEMBANGAN IMPOR BERAS DI INDONESIA*. Universitas Muhammadiyah Sumatra Utara.

Furi, R. P., Jondri and Saepudin, D. (2015) 'Prediksi Financial Time Series Menggunakan Independent Component Analysis dan Support Vector Regression Studi Kasus : IHSG dan JII', in, pp. 3608–3618.

Khotimah, K. A. (2018) *Analisis faktor-faktor yang mempengaruhi impor beras di indonesia tahun 1980 – 2016*. Universitas Muhammadiyah Surakarta.

Kurniyawan, H. (2013) *FAKTOR-FAKTOR YANG MEMPENGARUHI IMPOR BERAS DI INDONESIA TAHUN 1980-2009*. Semarang.

Kusuma, B. S. (2015) 'Analisa Peramalan Permintaan Air Minum Dalam Kemasan Pada PT . XYZ Dengan Metode Least Square dan Standard Error of Estimate', 4(1), pp. 42–47.

Li, C., Lu, Z. and Zhou, K. (2005) 'SVR-Parameters Selection for Image Watermarking', in *Proceedings of the 17th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'05)*. Wuhan.

Maharesi, R. (2013) 'Penggunaan Support Vector Regression (SVR) Pada Prediksi Return Saham Syariah BEI', in *PESAT (Psikologi, Ekonomi, Sastra, Arsitektur & Teknik Sipil)*. Bandung, pp. 8–9.

Mustakim, Buono, A. and Hermadi, I. (2015) 'Support Vector Regression Untuk Prediksi', 12(2), pp. 179–188.

Patel, V. R. and Mehta, R. G. (2011) 'Impact of outlier removal and normalization approach in modified k-means clustering algorithm', *IJCSI International*

Journal of Computer Science Issues, 8(5), pp. 331–336. Available at: www.IJCSI.org.

Putri, N., Santoso, E. and Adinugroho, S. (2017) 'Prediksi Volume Impor Beras Nasional dengan Metode Multi-Factors High-Order Fuzzy Time Series', *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer*, 1(12), pp. 1771–1778.

Raharyani, M. P., Putri, R. R. M. and Setiawan, B. D. (2018) 'Implementasi

Algoritme Support Vector Regression Pada Prediksi Jumlah Pengunjung Pariwisata', *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer*, 2(4), pp. 1501–1509.

Rifqi, M. R., Setiawan, B. D. and Bacthiar, F. A. (2018) 'Support Vector Regression Untuk Peramalan Permintaan Darah : Studi Kasus Unit Transfusi Darah Cabang – PMI Kota Malang', 2(10), pp. 3332–3342.

Sugiyono (2015) *STATISTIK untuk PENELITIAN*. Bandung: ALFABETA, cv.

Vijayakumar, S. and Wu, S. (1999) 'Sequent Vector Classifier and Regression', in *Proc. International Conference on Soft Computing (SOCO'99)*.

Wu, C., Ho, J. and Lee, D. T. (2004) 'Travel-Time Prediction With Support Vector Regression', 5(4), pp. 276–281.

Yousefi, M. *et al.* (2015) 'Support vector regression methodology for prediction of output energy in rice production', *Stochastic Environmental Research and Risk Assessment*, 29(8), pp. 2115–2126. doi: 10.1007/s00477-015-1055-z.

Yuniastari, N. L. A. K. and Wirawan, I. G. P. (2016) 'Peramalan Permintaan Produk Perak Menggunakan Metode Simple Moving Average Dan Single Exponential Smoothing', *Sistem dan Informatika STIKOM Bali*, pp. 97–106.