PERAMALAN HARGA PASAR TELUR AYAM RAS DI KOTA MALANG DENGAN MENGGUNAKAN METODE "SVR – PSO"

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:
Nuriya Fadilah
NIM: 145150201111064

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

PENGESAHAN

PERAMALAN HARGA PASAR TELUR AYAM RAS DI KOTA MALANG DENGAN MENGGUNAKAN METODE "SVR-PSO"

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh :
Nuriya Fadilah
NIM: 145150201111064

Skripsi ini telah diuji dan dinyatakan lulus pada
30 Juli 2018
Telah diperiksa dan disetujui oleh:

Dosen Pembimbing

Arief Andy Soebroto, S.T, M.Kom
NIP: 19720425 199903 1 002

Mengetahui
Ketua Jurusan Teknik Informatika

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

PERNYATAAN ORISINALITAS

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

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

Malang, 15 Juli 2018

Nuriya Fadilah

NIM: 145150201111064

KATA PENGANTAR

Puji dan syukur penulis panjatkan ke hadirat Allah SWT yang telah melimpahkan kasih dan sayang-Nya kepada kita, sehingga penulis bisa menyelesaikan skripsi dengan tepat waktu, yang kami beri Judul "Peramalan Harga Telur Ayam Ras Di Kota Malang Dengan Menggunakan Metode SVR-PSO".

Tujuan dari penyusunan skripsi ini guna memenuhi salah satu syarat untuk bisa menempuh ujian sarjana komputer pada Fakultas Ilmu Komputer (FILKOM) Program Studi Teknik Informatika di Universitas Brawijaya Malang.

Didalam pengerjaan skripsi ini telah melibatkan banyak pihak yang sangat membantu dalam banyak hal. Oleh sebab itu, disini penulis sampaikan rasa terima kasih sedalam-dalamnya kepada :

1. Bapak Arief Andy Soebroto, S.T, M.Kom selaku Dosen Pembimbing I yang telah membimbing dalam penyusunan Skripsi ini hingga selesai.
2. Bapak Wayan Firdaus Mahmudy, S.Si, M.T, Ph.D selaku Dekan Fakultas Ilmu Komputer Universitas Brawijaya yang telah memberikan ijin penelitian.
3. Bapak Tri Astoto Kurniawan, S.T, M.T, Ph.D selaku Ketua Jurusan Teknik Informatika Fakultas Ilmu Komputer Universitas Brawijaya yang telah menyetujui permohonan penyusunan Skripsi.
4. Bapak Agus Wahyu Widodo, S.T, M.Cs selaku Ketua Program Studi Teknik Informatika Ketua Jurusan Teknik Informatika Fakultas Ilmu Komputer Universitas Brawijaya.
5. Orang tua tercinta yang telah banyak memberikan doa dan dukungan kepada penulis secara moril maupun materil hingga skripsi ini dapat selesai.
6. Kakak dan adik tercinta juga anggota keluarga dan kerabat yang senatiasa memberikan doa dan dukungan semangat kepada penulis.
7. Sahabat dan rekan seperjuangan tercinta yang tiada henti memberi dukungan dan motivasi kepada penulis.
8. Semua pihak yang telah banyak membantu dalam penyusunan skripsi ini yang tidak bisa penulis sebutkan semuanya.

Malang, 30 Juli 2018

Penulis

nuriyafdlh@gmail.com

ABSTRAK

Nuriya Fadilah, Peramalan Harga Pasar Telur Ayam Ras di Kota Malang dengan Menggunakan Metode "SVR-PSO"

Dosen Pembimbing : Arief Andy Soebroto

Telur ayam ras menjadi salah satu sumber protein favorit masyarakat karena harganya cukup terjangkau dibandingkan sumber protein hewani lainnya yang dijual bebas di pasaran. Permasalahan utama yang dihadapi konsumen adalah fluktuasi harga pasar telur ayam ras di Kota Malang, ada kalanya harga naik dan ada kalanya harga turun. Hal ini akan menjadi masalah jika harga pasar naik tajam dari harga pada bulan-bulan sebelumnya. Pada penelitian ini dibuat sistem yang mampu meramalkan harga pasar dengan menggunakan metode *Support Vector Regression* (SVR) untuk melakukan peramalan dan metode *Particle Swarm Optimization* (PSO) untuk mengoptimasi parameter SVR. Proses optimasi terdiri dari 4 tahapan utama, yaitu tahapan normalisasi, tahapan pelatihan SVR, tahapan PSO, dan tahapan pengujian SVR. Pada pengujian SVR didapatkan nilai MAPE terkecil yaitu sebesar 6,2186% dan pada pengujian SVR-PSO didapatkan nilai MAPE terkecil sebesar 1,8840%.

Kata kunci: *support vector regression, particle swarm optimization*, peramalan

ABSTRACT

Nuriya Fadilah, Market Price Forecasting for Purebred Chicken Egg in Malang Using "SVR-PSO"

Dosen Pembimbing : Arief Andy Soebroto

Purebred chicken egg become one of the most favorite protein sources in the community because the price is quite affordable than other protein sources that are sold freely in the market. The main problem is the fluctuation of the market price of eggs in Malang, there are times when prices rise and there are times when prices fall. This would be a problem if the price rose from the price in the preceding months. In this research, a system capable to predict the egg market price by using Support Vector Regression (SVR) method to forecast and Particle Swarm Optimization (PSO) method to optimize SVR parameters. The optimization process consists of 4 main stages, namely the normalization stage, SVR training stage, PSO stages, and testing stages. In the SVR performance testing yields the smallest MAPE value of 6.2186%. As for the SVR-PSO performance testing has the smallest MAPE value of 1.8840%.

Keywords: support vector regression, particle swarm optimization, forecasting

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR PERSAMAAN

DAFTAR LAMPIRAN

BAB 1 PENDAHULUAN

1.1 Latar belakang

Populasi penduduk dan taraf hidup masyarakat Kota Malang terus mengalami peningkatan dari tahun ke tahun. Hal ini sebanding dengan peningkatan kesadaran masyarakat terhadap konsumsi makanan sehat berprotein hewani yang akan mendorong peningkatan kebutuhan pangan hewani seperti, daging, telur, susu dan lain-lain (Ferdiansyah, 2013). Telur ayam ras menjadi salah satu sumber protein favorit masyarakat karena harganya cukup terjangkau dibandingkan sumber protein hewani lainnya yang dijual bebas di pasaran. Oleh karena itu, diperlukan adanya tindakan untuk menstabilkan harga telur ayam ras di pasaran seiring dengan peningkatan konsumsi telur ayam ras di kalangan masyarakat.

Harga pasar merupakan sesuatu yang fluktuatif, ada kalanya harga naik dan ada kalanya harga turun. Hal ini akan menjadi masalah jika harga pasar naik tajam dari harga pada bulan-bulan sebelumnya (Astuti, 2016). Menjaga kestabilan harga pasar telur ayam ras adalah tugas dari Bidang Peternakan dan Kesehatan Hewan Dinas Pertanian dan Ketahanan Pangan Kota Malang. Tindakan nyata untuk menjaga kestabilan harga telur ayam ras dengan memantau harga pasaran komoditas telur tiap bulannya. Tindakan preventif diperlukan untuk mencegah terjadinya anomali harga yaitu dengan menaikkan atau menurunkan suplai telur ayam ras. Tindakan preventif tersebut dapat dilakukan dengan adanya proyeksi kondisi pasar untuk meramalkan harga pasar telur ayam ras untuk periode yang akan datang dengan data harga pasar dari bulan ke bulan.

Peramalan diperlukan biasanya untuk memperkirakan apa yang akan terjadi di masa yang akan datang, sehingga para pengambil keputusan bisa membuat perencanaan untuk masa depan. *Forecasting* (peramalan) merupakan suatu prediksi untuk memperkirakan suatu nilai pada periode tertentu yang akan terjadi masa mendatang dengan menggunakan penjelasan secara matematik dan statistik (Montgomery, et al., 2015). Peramalan dapat dilakukan dengan dua macam pendekatan yaitu pendekatan *Artificial Intelligence* (AI) dan pendekatan statistik. *Machine Learning* (ML) adalah cabang *Artificial Intelligence* (AI). AI berfokus pada pemahaman bagaimana mereplikasi pengetahuan ke dalam mesin (sistem atau agen). Secara historis, teknik dan pendekatan ML sangat mengandalkan daya komputasi. Di sisi lain, teknik pendekatan statistik kebanyakan dikembangkan di mana kekuatan komputasi bukanlah pilihan. Akibatnya, TS sangat bergantung pada sampel kecil dan asumsi berat tentang data dan distribusinya (Hassibi, 2016). Peramalan dengan pendekaan AI dapat dilakukan dengan menggunakan metode *Autoregressive Moving Average (ARMA), Artificial Neural Network (ANN), Adaptive Neural-Based Fuzzy Inference System (ANFIS), Genetic Programming (GP), dan Support Vector Machines (SVM)* (Wang, et al., 2009).

Support Vector Regression (SVR) merupakan salah satu metode yang biasa digunakan untuk melakukan peramalan data *time series*. Pada penelitian (Lin, et al., 2006) menyimpulkan bahwa SVR digunakan karena performa pada saat komputasi yang baik dan hasil akurasi yang lebih tinggi. Dalam penelitian tersebut

menyatakan pada data *time series* dengan SVR menghasilkan bobot yang berbeda untuk data *history* yang berbeda yang menyebabkan parameter-parameter SVR dapat disesuaikan dengan data sampel. Untuk menyesuaikan parameter SVR dengan data diperlukan optimasi dalam menentukan nilai parameter SVR agar metode SVR mendapatkan hasil peramalan yang optimal.

Kombinasi metode SVR dan PSO dapat dilakukan untuk menaikkan kinerja SVR dengan menggunakan PSO untuk memilih parameter dari SVR. Pada penelitian mengenai peramalan jumlah kecelakaan, hasil pengujiannya mengindikasikan bahwa metode SVR-PSO memiliki hasil yang lebih baik dibandingkan dengan metode BP *neural network* (Zeng, et al., 2009). Pada penelitian mengenai peramalan volume penjualan mobil dengan SVR-PSO menyatakan bahwa SVR-PSO lebih baik dibandingkan dengan metode SVR-GA dalam tingkat efisiensi dan akurasi prediksi (Lu & Geng, 2011).

Berdasarkan permasalahan tersebut, untuk memudahkan petugas Dinas Perdagangan Kota Malang dalam meramalkan harga pasar telur ayam ras di Kota Malang maka penulis mengangkat judul Peramalan Harga Telur Ayam di Kota Malang dengan menggunakan Metode SVR – PSO. Diharapkan dengan adanya sistem ini dapat memberikan hasil peramalan yang baik sehingga dapat memudahkan petugas melakukan tindakan preventif untuk menstabilkan harga pasar telur ayam ras sebelum kenaikan harga terjadi.

1.2 Rumusan masalah

Berdasarkan uraian latar belakang, maka dapat dirumuskan permasalahan pada penelitian Peramalan Harga Pasar Telur Ayam Ras di Kota Malang Dengan Menggunakan Metode SVR – PSO antara lain:

1. Bagaimana merancang sistem peramalan harga pasar telur ayam ras di Kota Malang dengan menggunakan metode SVR – PSO.
2. Bagaimana mengimplementasikan sistem peramalan harga pasar telur ayam ras di Kota Malang dengan menggunakan metode SVR – PSO.
3. Bagaimana menguji sistem peramalan harga pasar telur ayam ras di Kota Malang dengan menggunakan metode SVR – PSO.

1.3 Tujuan

Tujuan dari penelitian skripsi ini adalah meramalkan harga telur ayam ras di Kota Malang menggunakan metode SVR – PSO.

1.4 Manfaat

Manfaat yang diharapkan dari penelitian sistem peramalan harga pasar telur ayam ras di Kota Malang dengan menggunakan metode SVR – PSO antara lain:

Bagi pemerintah:

1. Membaca kondisi pasar di periode kedepan untuk melakukan tindakan pencegahan fluktuasi harga telur ayam ras di Kota Malang.

2. Mengetahui tingkat permintaan komoditas telur ayam ras di pasar untuk mengantisipasi *over production* oleh peternak ayam.

Bagi penulis:

1. Sebagi sarana untuk mengaplikasikan ilmu yang telah didapat selama perkuliahan dalam masyarakat.
2. Mengetahui pengembangan dari bidang ilmu yang telah didapat selama masa perkuliahan.
3. Mengenal hubungan bidang ilmu yang ditekuni (teknologi informasi) dengan bidang ilmu lain.

Bagi peneliti:

1. Sebagai sarana pembelajaran penerapan metode yang digunakan dengan studi kasus telur ayam ras di Kota Malang.
2. Mengetahui efektivitas dan efisiensi metode optimasi *Particle Swarm Optimization* untuk perbaikan metode *Support Vector Regression*.
3. Sebagai sumber referensi dalam penelitian-penelitian di masa yang akan datang.

Bagi masyarakat:

1. Mengetahui informasi terkait harga telur ayam ras di pasaran.
2. Membuat perkiraan pengeluaran belanja komoditas telur ayam di pasaran.

1.5 Batasan masalah

Pada bagian ini merupakan batasan-batasan masalah yang diterapkan pada penelitian ini adalah sebagai berikut :

1. Data harga telur ayam ras yang digunakan, diperoleh dari Dinas Perdagangan Kota Malang dengan rentang waktu 2012-2017 yang menggunakan data *time series* dengan satuan rupiah per kilogram.
2. Menggunakan metode *Support Vector Regression* dengan optimasi *Particle Swarm Optimization.*
3. Menggunakan bahasa pemrograman Java.
4. Menggunakan pengujian SVR, pengujian banyak iterasi pelatihan SVR, pengujian rentang pencarian dimensi parameter SVR, pengujian SVR-PSO, pengujian banyak iterasi PSO, dan pengujian jumlah partikel PSO.

1.6 Sistematika pembahasan

Sistematika penulisan digunakan untuk memberikan gambaran serta uraian dari suatu penelitian. Berikut merupakan sistematika penulisan dari penelitian Peramalan Harga Pasar Telur Ayam Ras di Kota Malang Dengan Menggunakan Metode SVR – PSO:

BAB 1 Pendahuluan

Bab ini membahas tentang latar belakang, rumusan masalah, tujuan, manfaat, batasan masalah dan sistematika penulisan Peramalan Harga Pasar Telur Ayam Ras di Kota Malang Dengan Menggunakan SVR – PSO.

BAB 2 Landasan Kepustakaan

Bab ini membahas tentang tinjauan pustaka mengenai teori dan referensi yang mendasari pembuatan Peramalan Harga Pasar Telur Ayam Ras di Kota Malang Dengan Menggunakan SVR – PSO.

BAB 3 Metodologi

Bab ini membahas tentang metode yang digunakan dalam penelitian yang terdiri dari studi literatur, perancangan sistem perangkat lunak, pengujian dan analisis, serta penulisan laporan dalam penelitian Peramalan Harga Pasar Telur Ayam Ras di Kota Malang Dengan Menggunakan Metode SVR – PSO.

BAB 4 Perancangan

Bab ini membahas tentang kebutuhan dan perancangan Prediksi Harga Pasar Telur Ayam Ras di Kota Malang Dengan Menggunakan Metode SVR – PSO.

BAB 5 Hasil

Bab ini membahas tentang bagaimana implementasi perangkat lunak sesuai dengan perancangan sistem yang telah dibuat dalam penelitian Peramalan Harga Pasar Telur Ayam Ras di Kota Malang Dengan Menggunakan SVR – PSO.

BAB 6 Pembahasan

Bab ini membahas tentang hasil pengujian dari aplikasi berdasarkan parameter-parameter yang telah diterapkan yang kemudian dilakukan analisa terhadap hasil pengujian.

BAB 7 Penutup

Bab ini membahas tentang kesimpulan yang diperoleh setelah mendapatkan solusi dari permasalahan melalui perhitungan menggunakan metode SVR dengan optimasi PSO.

BAB 2 LANDASAN KEPUSTAKAAN

Pada bab ini akan diuraikan penjelasan mengenai kajian pustaka dan dasar teori tentang metode dan objek penelitian. Landasan kepustakaan meliputi kajian pustaka, dasar teori telur ayam ras, dasar teori metode peramalan *support vector regression* (SVR), dan dasar teori metode optimasi *particle swarm optimization* (PSO).

2.1 Kajian Pustaka

Penelitian ini didasarkan pada studi literatur beberapa penelitian yang sebelumnya pernah dilakukan yang berkaitan dengan metode yang akan digunakan oleh penulis yaitu *Support Vector Regression* (SVR) dan *Particle Swarm Optimization* (PSO). Studi literatur ini dilakukan untuk memperkuat pemahaman permasalahan dan mencari solusi yang terbaik. Beberapa penelitian yang terdahulu terdapat dalam Tabel 2.1.

Kajian pustaka pertama berjudul "*Particle Swarm Optimization For Traveling Salesman Problem*" dengan menggunakan pengujian nilai jarak terkecil. Berdasarkan hasil pengujian, metode PSO dengan menggunakan 100 partikel dapat mengoptimasi permasalahan *Travelling Salesman Problem* dengan 14 node dengan jarak tempuh awal 59,8462 dioptimasi hingga 30,8785 dalam 0,064% ruang pencarian (Wang, et al., 2003).

Kajian pustaka kedua berjudul "*Travel Time Prediction with Support Vector Regression"* dengan menggunakan dua parameter pengujian yaitu RME dan RMSE, lalu membandingkan akurasi metode SVR dengan *current travel time prediction* dan *historical mean prediction*. Berdasarkan hasil pengujian, metode SVR menghasilkan nilai RME dan RMSE terkecil yaitu 1,21% dan 1,63%. Hal ini menunjukkan bahwa metode SVR cukup baik untuk meramalkan waktu perjalanan dan analisis data lalu lintas (Wu, et al., 2003).

Kajian pustaka ketiga berjudul "*Bankruptcy Prediction using Support Vector Machines with Optimal Choice of Kernel Function Parameters"* dengan menggunakan pengujian perbandingan akurasi metode SVM dengan *Backpropagation Neural Network* (BPNN), *Multiple Discriminant Analysis* (MDA) dan *Logistic Regression Analysis* (Logit). Berdasarkan hasil pengujian, metode SVM memiliki akurasi sebesar 83,0688%, sedangkan akurasi BPNN sebesar 82,5397%, MDA sebesar 79,1391%, dan Logit sebesar 78,3069%. Hal ini menujukkan bahwa metode SVM lebih baik daripada BPNN, MDA, dan Logit untuk peramalan kepailitan perusahaan (Min & Lee, 2005).

Kajian pustaka keempat berjudul "*Support Vector Regression For Financial Time Series Forecasting*" dengan menggunakan dua tahap pengujian yaitu pengujian nilai RMSE dan nilai MSE. Pada penelitian ini, penulis menggunakan *Radial Basis Function* sebagai fungsi kernelnya. Berdasarkan hasil pengujian, metode ini menghasilkan nilai MAE berkisar 0,8281 sampai dengan 32,5634 dan nilai RMSE berkisar 0,8791 sampai dengan 38,1064. Hal ini menunjukkan bahwa metode SVR

lebih efektif dan efisien dalam peramalan indeks saham dibandingkan dengan SVR standar dan model permalan *time series* tradisional (Hao & Yu, 2006).

Kajian pustak kelima berjudul "*A Forecasting Methodology Using Support Vector Regression and Dynamic Feature Selection"* dengan menggunakan tiga parameter pengujian yaitu MAPE, MAE, dan RMSE, lalu membandingkan metode *Support Vector Machine*-UP (SVM-UP) dengan *Autoregressive-moving-average model (*ARMAX) dan *Neural Network*-UP. Berdasarkan hasil pengujian, metode SVM-UP menghasilkan nilai MAPE, MAE, dan RMSE terkecil yaitu sebesar 10.16, 254, dan 350 dibandingkan dengan metode ARMAX dan NN-UP. Hal ini membuktikan bahwa SVM-UP lebih baik dalam memprediksi data penjualan (Guajardo, et al., 2006).

Kajian pustaka keenam berjudul "*Application of Support Vector Regression and Particle Swarm Optimization in Traffic Accident Forecasting*" dengan menggunakan pengujian perbandingan nilai *error* dengan metode *Backpropagation Neural Network*. Berdasarkan hasil pengujian, metode PSO-SVR menghasilkan nilai MAPE 2,39% pada tahun 2003 dan 2,11% pada tahun 2004. Sedangkan metode BP *Neural Network* menghasilkan nilai MAPE 3,45% pada tahun 2003 dan 3,26% pada tahun 2004. Hal ini menunjukkan bahwa metode PSO-SVR memiliki akurasi lebih baik dibandingkan dengan BP *Neural Network* pada peramalan jumlah kasus kecelakaan lalu lintas (Zeng, et al., 2009).

Kajian pustaka yang ketujuh berjudul "*Support Vector Regression on Particle Swarm Optimization for Rainfall Forecasting*" dengan menggunakan pengujian perbandingan nilai error hasil prediksi SVR dan hasil prediksi SVR yang dioptimasi dengan PSO. Pengujian ini menggunakan empat macam perhitungan nilai error yaitu, MAPE, RMSE, MAE, dan PR. Berdasarkan hasil pengujian, dapat disimpulkan bahwa SVR-PSO lebih baik dibandingkan dengan SVR. Perbedaan nilai error antara metode SVR dan metode SVR-PSO cukup besar. Pada pengujian menggunakan 45 data latih, metode SVR menghasilkan nilai MAPE 19,72% sedangkan metode SVR-PSO memiliki nilai MAPE 3,54% (Zhao & Wang, 2010).

Kajian pustaka yang kedelapan berjudul "*Grid Resources Prediction with Support Vector Regression and Particle Swarm Optimization*" dengan menggunakan dua tahap pengujian yaitu pengujian nilai error dan pengujian lama komputasi sistem. Pada pengujian lama komputasi, metode SVR-PSO menghabiskan waktu komputasi 177 detik sedangkan metode SVR menghabiskan waktu komputasi selama 514 detik. Pada pengujian nilai error, metode SVR-PSO menghasilkan nilai NMSE 0,2477, metode SVR menghasilkan nilai NMSE 0,6187, dan metode BPNN menghasilkan nilai NMSE 0,3022. Berdasarkan hasil pengujian, dapat disimpulkan metode SVR-PSO menghabiskan waktu komputasi lebih kecil dan menghasilkan akurasi yang lebih baik dibandingkan dengan metode SVR dan metode BPNN (Hu, et al., 2010).

Kajian pustaka yang kesembilan berjudul "*Car Sales Volume Prediction Based on Particle Swarm Optimization Algoritm and Support Vector Regression*" dengan menggunakan pengujian nilai MAPE. Pada pengujian penulis membandingkan

hasil prediksi metode SVR-PSO dengan *Support Vector Regression – Genetic Algorithm (*SVR-GA). Berdasarkan hasil pengujian, dapat disimpulkan bahwa metode SVR-PSO lebih akurat dibandingkan dengan SVR-GA (Lu & Geng, 2011).

Kajian pustaka yang kesepuluh berjudul "*Short-term Load Forecasting for Electric Power Systems Using the PSO-SVR and FCM Clustering Techniques"* dengan menggunakan parameter *average error* untuk membandingkan metode FCM-PSO-SVR dengan metode PSO-SVR. Berdasarkan hasil pengujian, *average error* metode FCM-PSO-SVR adalah 1,218% sedangkan metode PSO-SVR sebesar 1,542%. Hal ini menujukkan bahwa metode FCM-PSO-SVR bekerja lebih baik dibandingkan metode SVR-PSO, walaupun perbedaannya tidak terlalu signifikan (Duan, et al., 2011).

Kajian pustaka yang kesebelas berjudul "*A Hybrid Particle Swarm Optimization and Support Vector Regression Model for Financial Time Series Forecasting"* dengan menggunakan enam parameter pengujian *error* yaitu RMSE, MAD, MAPE, DS, CU, dan CD untuk membandingkannya dengan metode SVR tradisional. Berdasarkan hasil pengujian, metode PSO-SVR menghasilkan nilai RMSE, MAD, dan MAPE sebesar 49.10, 36.32, dan 0.58% sedangkan metode SVR tradisional menghasilkan nilai RMSE, MAD, dan MAPE sebesar 49.27, 36.67, dan 0.59%. Dari hasil pengujian dapat disimpulkan bahwa metode PSO-SVR menghasilkan tingkat akurasi yang lebih tinggi dan nilai *error* yang lebih rendah dibandingkan metode SVR tradisional (Hsieh, et al., 2011).

Kajian pustaka yang kedua belas berjudul "*Study on Coal Logistics Demand Forecast Based on PSO-SVR*" dengan menggunakan pengujian nilai *relative error*. Pada penelitian ini penulis mencoba membandingkan akurasi hasil prediksi metode PSO-SVR dengan akurasi hasil prediksi metode *Backpropagation* (BP). Hasil prediksi menggunakan metode BP menghasilkan nilai *relative error* 4,08% pada 2009, 3,47% pada tahun 2010, dan 3,14% pada tahun 2011. Sedangkan, hasil prediksi menggunakan metode PSO-SVR menghasilkan nilai *relative error* 1,11% pada tahun 2009, 1,26% pada tahun 2010, dan 1,28% pada tahun 2011. Dari hasil pengujian dapat disimpulkan bahwa metode SVR-PSO menghasilkan tingkat akurasi yang lebih tinggi dan nilai *error* yang lebih rendah dibandingkan metode BP (Pei-you & Lu, 2013).

Kajian pustaka yang ketiga belas berjudul "*The Use of Hybrid SVR-PSO model to Predict Pipes Failure Rates"* dengan menggunakan tiga parameter pengujian yaitu R^2, RMSE, dan NRMSE, untuk membandingkan kinerja empat tipe kernel SVR yaitu kernel RBF, kernel eRBF, kernel Polinomial, dan kernel Spline. Berdasarkan hasil pengujian, kernel RBF menghasilkan akurasi terbaik dengan nilai R^2 sebesar 0,9999, nilai RMSE dan NRMSE sebesar 0,0025 dan 0,007 (Kalanaki, et al., 2013).

Kajian pustaka yang keempat belas berjudul "*Prediction of Earthquake Induced Displacement of Slopes Using Hybrid Support Vector Regression With Particle Swarm Optimization"* dengan menggunakan lima parameter pengujian untuk mengukur kinerja sistem yaitu R^2, MSE, RMSE, VAF, dan MAPE. Berdasarkan hasil pengujian, metode SVR-PSO menghasilkan nilai R^2, MSE, RMSE, VAF, dan MAPE

sebesar 0.9959, 0.00068, 0.026, 99.58, 3,4547. Hal ini menunjukkan bahwa metode SVR-PSO menghasilkan akurasi yang baik dalam memprediksi pergeseran lereng akibat gempa bumi (Fattahi, 2015).

Kajian pustaka yang kelima belas berjudul "*PSO-SVR: A Hybrid Short-term Traffic Flow Forecasting Method*" dengan menggunakan pengujian nilai RMSE. Pada penelitian ini penulis membandingkan nilai RMSE yang dihasilkan oleh beberapa metode prediksi lainnya, yaitu GA-SA (*Hybrid Genetic Algorithm-Stimulated Annealing*), KNN-LWNN *(K-Nearest Neighbor based on Linear Wavelet Neural Network*) dan SSA (*Singular Spectrum Analysis*). Hasil pengujian prediksi lalu lintas pada tanggal 01/05/2013, nilai RMSE metode PSO-SVR adalah 16,1457, metode GA-SA 16,5248, metode KNN-LWNN 17,0231 dan metode SSA 16,7412. Berdasarkan hasil pengujian, metode PSO-SVR cenderung memiliki nilai RMSE lebih kecil dan akurasi lebih tinggi dibandingkan metode yang diuji lainnya (Hu, et al., 2015).

Kajian pustaka keenam belas berjudul "Prediksi Tinggi Muka Air (TMA) Untuk Deteksi Dini Bencana Banjir Menggunakan SVR-TVIWPSO" menggunakan pengujian nilai *Mean Absolute Error* (MAE). Hasil pengujian yang didapat dari 10 data bulanan yang berbeda menunjukkan bahwa didapatkan nilai error terkecil sebesar 0.0075 dengan menggunakan *Mean Absolute Error* (MAE) untuk data Juni 2007 dengan menggunakan integrasi metode SVR-TVIWPSO (Soebroto, et al., 2015).

Kajian pustaka ketujuh belas berjudul "Optimasi Pemenuhan Kebutuhan Gizi Keluarga Menggunakan *Particle Swarm Optimization*" dengan menggunakan lima tahap pengujian yaitu pengujian banyaknya partikel, pengujian kombinasi ω_{min} dan ω_{max}, pengujian banyaknya jumlah iterasi, pengujian jumlah maksimum iterasi dan pengujian global. Berdasarkan hasil pengujian, nilai *fitness* terbaik didapatkan dari kombinasi ω_{min} dan ω_{max} sebesar 0,4 dan 0,7 serta iterasi maksimal 60. Hasil rekomendasi dari sistem mampu menghemat biaya pengeluaran untuk konsumsi sebesar 39,31% dan mampu memenuhi kebutuhan gizi keluarga dengan selisih ±10% dari kebutuhan gizi yang dianjurkan (Eliantara, et al., 2016).

Kajian pustaka kedelapan belas berjudul "*Optimizing SVR using Local Best For PSO for Software Effort Estimation*" dengan menggunakan tiga tahap pengujian yaitu, pengujian jumlah partikel PSO, pengujian lama komputasi, dan pengujian konvergensi. Berdasarkan hasil pengujian, jumlah partikel dengan nilai dan waktu komputasi paling optimal adalah 15 partikel dengan jumlah iterasi 8. Hasil tersebut menunjukkan bahwa metode SVR-PSO membutuhkan waktu komputasi lebih sedikit karena kemampuan *fast convergence* (Novitasari, et al., 2016).

Kajian pustaka kesembilan belas berjudul *"Particle Swarm Optimization with Fitness Adjustment Parameter*" dengan menggunakan pengujian perbandingan performa 3 variasi metode PSO yaitu SPSO, APSOVI, dan PSOFAP. Dari hasil pengujian, rata-rata jarak tempuh yang diperoleh metode PSOFAP lebih kecil dibandingkan dengan metode SPSO dan APSOVI. Hal ini menunjukkan bahwa

metode PSOFAP lebih efektif dalam menyelesaikan masalah pencarian jalur terpendek (Li & Cheng, 2017).

Kajian pustaka kedua puluh berjudul "*A Simplified Particle Swarm Optimization for Job Scheduling in Cloud Computing*" dengan menggunakan pengujian perbandingan nilai *makespan* GA dan PSO. Berdasarkan hasil pengujian, PSO dapat menyelesaikan permasalahan penjadwalan dalam ukuran besar dan memiliki kemampuan untuk meminimalkan *makespan* dari *job-scheduling*. Jika dibandingkan dengan GA, metode PSO memiliki kemampuan untuk konvergen lebih cepat (Attiya & Zhang, 2017).

Berdasarkan beberapa penelitian diatas, maka penulis mengajukan penelitian dengan judul "Prediksi Harga Pasar Telur Ayam Ras di Kota Malang dengan Menggunakan Metode *Support Vector Regression - Particle Swarm Optimization* (SVR-PSO)". Sistem prediksi ini menggunakan *Support Vector Regression* sebagai metode pelatihan data *time series* dan *metode Particle Swarm Optimization* untuk mengoptimasi parameter metode awal *Support Vector Regression* sehingga sistem ini diharapkan menghasilkan hasil prediksi dengan akurasi yang lebih baik dan waktu komputasi yang lebih cepat.

Tabel 2.1 Daftar kajian pustaka

No	Judul	Objek Input	Metode Proses	Hasil
1	*Particle Swarm Optimization For Travelling Salesman Problem* (Wang, et al., 2003)	Data *TSP benchmark problem* dengan 14 node.	Menggunakan metode *Particle Swarm Optimization* (PSO), proses: 1. Inisialisasi kecepatan dan posisi partikel awal 2. Inisialisasi pBest dan Gbest 3. Lakukan perulangan hingga kondisi berhenti terpenuhi 4. Update kecepatan 5. Update posisi dan evaluasi *fitness* 6. Tentukan pBest dang Best	Dari hasil pengujian menunjukkan bahwa metode ini memiliki sifat *fast convergence* dan hasil pencarian optimal.
2	*Travel Time Prediction with Support Vector Regression* (Wu, et al., 2003)	Data lalu lintas diambil dari Intelligent Transportation Web Service Project (ITWS), Taiwan. Dataset terpilih adalah data lalu lintas tanggal 15 Februari 2003 sampai 21 Maret 2003.	Menggunakan metode Support Vector Regression (SVR), proses : 1. Inisialisasi parameter SVR 2. Hitung matriks hessian 3. Sequential leaning 4. Testing data 5. Hasil prediksi	Hasil menunjukkan bahwa prediktor SVR secara signifikan mengungguli metode prediksi yang lain. Untuk jarak 350 km, nilai RMSE SVR paling kecil dibandingkan *Current-time predictor* dan *historical-mean predictor* yaitu sebesar 1,63%
3	*Bankruptcy prediction using support vector machine with optimal choice of kernel function parameters*(Min & Lee, 2005)	Data diambil dari Organisasi Penjamin Kredit terbesar di Korea. Data terdiri dari data kredit dari 1888 perusahaan termasuk 944 dinyatakan pailit dan 944 dinyatakan tidak pailit.	Menggunakan metode *Support Vector Machine* (SVM), proses : 1. Analisis dan pre-processing data 2. Analisis grafis dari hasil prediksi dengan model kernel SVM	Untuk memvalidasi kinerja prediksi model ini, secara statistic membandingkan akurasi prediksi SVM dengan tiga layer BPN, MDA, dan Logit,. Hasil analisis empiris menunjukkan bahwa SVM mengungguli metode lain. Dengan hasil ini, data dinyatakan bahwa SVM dapat berfungsi sebagai alternatif yang menjanjikan untuk prediksi masalah tersebut.

			3. Perbaikan bertahap hasil regresi dan pemilihan rasio keuangan 4. Membandingkan akurasi dari SVM, BPN, MDA, dan Logit	
4	*Support Vector Regression For Financial Time Series Forecasting* (Hao & Yu, 2006)	Data indeks harga saham gabungan *Shanghai Stock Exchange* antara 21 Januari 2002 hingga 27 September 2002.	Menggunakan metode *Modified Support Vector Regression*, proses: 1. Membagi data ke dalam data latih dan data uji 2. *Preprocessing* data 3. *Training* SVR 4. Evaluasi nilai RMSE dan nilai MAE	Dari hasil pengujian menunjukkan bahwa metode *Modified* SVR ini lebih efektif dan efisien dibandingkan dengan SVR standar dan metode peramalan *time series* tradisional.
5	*A Forecasting Methodology Using Support Vector Regression and Dynamic Feature Selection* (Guajardo, et al., 2006)	Data penjualan mingguan dari Januari 2001 sampai September 2004	Menggunakan metode Support Vector Regression (SVR), proses : 1. Normalisasi data 2. Menentukan parameter dasar (epsilon, C, dan K) 3. Seleksi fitur 4. Seleksi model 5. Hasil model prediksi	Membandingkan hasil masing-masing metode yaitu ARMAX, NN-UP dan SVR-UP menunjukkan bahwa metode SVR-UP sedikit lebih baik dan juga menyediakan pilihan fitur yang paling penting. Keuntungan utama dari metodologi yang diusulkan diharapkan ketika berhadapan dengan fenomena dinamis, di mana kinerja peramalan model dapat ditingkatkan secara signifikan dengan melakukan pembaruan model.
6	*Application of Support Vector Regression and Particle Swarm Optimization in Traffic Accident Forecasting* (Zeng, et al., 2009)	Data jumlah kasus kecelakaan di jalan tertentu di Cina dari tahun 1995 hingga tahun 2004	Menggunakan metode PSO-SVR, proses: 1. Inisialisasi kecepatan dan posisi tiap partikel 2. Perulangan hingga kondisi berhenti tercapai 3. Evaluasi *fitness* tiap partikel	Dari hasil pengujian, metode PSO-SVR menghasilkan akurasi yang lebih baik dibandingan dengan metode BP *neural network* dalam peramalan jumlah kasus kecelakaan lalu lintas.

			4. Perbarui nilai kecepatan dan posisi partikel	
7	*Support Vector Regression Based on Particle Swarm Optimization for Rainfall* (Zhao & Wang, 2010)	Data curah hujan bulan yang diambil di kota Guangxi pada tahun 1954-2008	Metode *Support Vector Regression* dengan *Particle Swarm Optimization*, proses : 1. Inisialisasi data, parameter SVR, iterasi maksimal PSO, range, populasi, jumlah fitur 2. Normalisasi data 3. Inisialisasi partikel awal 4. Proses SVR 5. Hitung *fitness* 6. Lakukan perulangan a. *Update* nilai w b. *Update* nilai c1 dan c2 c. Lakukan perulangan 1. *Update* kecepatan 2. *Update* posisi 3. SVR 4. Hitung *fitness* 5. *Update* pBest d. Iterasi sama dengan jumlah populasi, perulangan berhenti e. Update nilai gBest 7. Mencapai iterasi maksimal, perulangan berhenti	Hasil pengujian metode SVR dengan dengan 45 data latih menghasilkan nilai MAPE 19,72%, nilai RMSE 368,3353, nilai MAE 296,8337, dan nilai PR 0,7582. Sedangkan hasil pengujian metode SVR-PSO dengan 45 data latih menghasilkan nilai MAPE 3,54%, nilai RMSE 62,8356, nilai MAE 49,1035 dan nilai PR 0,9878. Dari hasil pengujian dapat disimpulkan bahwa metode SVR yang dioptimasi dengan PSO menghasilkan hasil prediksi yang lebih baik dibandingkan dengan metode SVR.

			8. Hasil peramalan curah hujan	
8	*Grid Resources Prediction with Support Vector Regression and Particle Swarm Optimization* (Hu, et al., 2010)	Data *Grid resource* berupa informasi *resource host load*, penggunaan CPU, konsumsi memori, dan lain-lain.	Metode *Support Vector Regression – Particle Swarm Optimization* (SVR-PSO), proses : 1. Inisialisasi parameter SVR 2. Inisialisasi partikel 3. Evaluasi nilai *fitness* a. Proses SVR b. Hasil prediksi c. Hitung error 4. *Update* pBest dan gBest 5. *Update* kecepatan dan posisi partikel 6. Ulangi sampai optimal	• Dari hasil pengujian lama komputasi, metode SVR-PSO menghabiskan lama komputasi 177 dengan nilai MSE 0,0103. Sedangkan pada metode SVR menghabiskan lama komputasi 514 dengan nilai MSE 0,0185. • Dari hasil perbandingan nilai NMSE antara metode SVR, SVR-PSO, dan BPNN, metode SVR-PSO mendapatkan nilai NMSE terendah dibandingkan SVR dan BPNN sebesar 0,2477.
9	*Car Sales Volume Prediction Based on Particle Swarm Optimization Algorithm and Support Vector Regression* (Lu & Geng, 2011)	Meramalkan volume penjualan mobil tahunan di Taiwan tahun 2003-2009	Metode *Support Vector Regression* dengan *Particle Swarm Optimization*, proses : 1. Inisialisasi parameter PSO 2. Inisialisasi posisi dan kecepatan secara acak 3. *Training* model SVR 4. Evaluasi nilai *fitness* 5. *Update* posisi dan kecepatan partikel 6. *Update* pBest dang Best 7. Ulangi langkah 3 sampai optimal	Hasil empiris menyimpulkan bahwa akurasi prediksi metode SVR-PSO lebih tinggi dibandingkan dengan metode SVR-GA.
10	*Short-term Load Forecasting for Electeric Power Systems Using the PSO-SVR and FCM*	Data *load forecasting* dari Kota Chongqing, Cina	Metode SVR-PSO dan Fuzzy C-Means Clustering, proses : 1. Input dan pre-processing data	Klasterisasi FCM optimal digunakan untuk dapatkan pembagian pola data optimal dari sampel history load dan set sampel pelatihan terbaik. Keteraturan data hubungan fungsi input-

	Clustering Techniques (Duan, et al., 2011)		2. Analisis fitur 3. Optimasi klaster FCM untuk membentuk himpunan sample 4. Hitung jarak data input dan pusat klaster 5. Pilih jarak terkecil sebagai himpunan model sample prediksi SVR-PSO 6. Menentukan parameter SVR dengan optimasi PSO 7. Menentukan fungsi objektif dan membentuk rumus regresi optimal 8. Hasil peramalan load forecasting	output dalam sampel pelatihan dapat ditingkatkan, konsistensi karakteristik data dapat dipastikan. Integrasi efektif PSO-SVR dan algoritme pengelompokan FCM optimal tercapai.
11	*A Hybrid Particle Swarm Optimization and Support Vector Regression for Financial Time Series Forecasting* (Hsieh, et al., 2011)	Data indeks harga saham penutupan TAIEX dari 2 Januari 2003 sampai 27 Februari 2006	Metode *Support Vector Regression* dengan *Particle Swarm Optimization*, proses : 1. Inisialisasi partikel acak PSO 2. Training SVR partikel dan evaluasi *fitness* 3. Update posisi dan kecepatan partikel 4. Update *fitness*, pBest, dan gBest 5. Lakukan sampai PSO mencapai iterasi maksimal	Dibandingkan dengan model SVR tradisional, hasil eksperimen menunjukkan bahwa model PSO-SVR yang diusulkan dapat menghasilkan kesalahan prediksi yang lebih rendah dan akurasi prediksi yang lebih tinggi dalam dataset. Selain itu, waktu CPU untuk yang diusulkan model sekitar delapan kali lebih cepat dari itu ketika menggunakan metode SVR tradisional. Karena itu model PSO-SVRseharusnya memang alternatif yang lebih baik daripada model SVR tradisional karena dapat menghemat banyak waktu pemodelan di atas kemampuan meramal yang lebih baik.

12	*Study on Coal Logistics Demand Forecast Based on PSO-SVR* (Pei-you & Lu, 2013)	Data permintaan logistik batubara, data dari kereta pengangkut batu bara di China tahun 1995-2011	Metode *Support Vector Regression* dengan *Particle Swarm Optimization*, proses : 1. Inisialisasi parameter PSO dan SVR 2. *Update* kecepatan dan posisi partikel 3. Evaluasi nilai *fitness* → *training* SVR 4. *Update* pBest dan gBest 5. Ulangi sampai iterasi maksimal	Dari hasil pengujian didapatkan bahwa metode PSO-SVR memiliki tingkat akurasi lebih tinggi dibandingkan metode BPNN.
13	*The Use of Hybrid SVR-PSO Model to Predict Pipes Failure Rates* (Kalanaki, et al., 2013)	Dataset kondisi jaringan distribusi air sebuah kota di Iran. Dataset ini terdiri dari 2438 data dari tahun 2005 sampai 2006, tersimpan informasi diameter, tahun implementasi, kedalaman pipa, jumlah kecelakaan yang terjadi, dan rata-rata tekanan air.	Metode *Support Vector Regression* dengan *Particle Swarm Optimization*, proses : 1. Inisialisasi parameter PSO 2. Inisialisasi posisi dan kecepatan secara acak 3. *Training* model SVR 4. Evaluasi nilai *fitness* 5. *Update* posisi dan kecepatan partikel 6. *Update* pBest dang Best 7. Ulangi langkah 3 sampai maksimal iterasi	Menggunakan algoritme PSO untuk mengoptimasi parameter SVR. Parameter ini memiliki akurasi lebih baik dan hasil menunjukkan kinerja yang lebih baik antara data aktual dan data prediksi.
14	*Prediction of Earthquake Induced Displacements of Slopes Using Hybrid Support Vector Regression with Particle Swarm*	Dataset pergerakan lereng akibat gempa bumi diterapkan dalam penelitian ini untuk menentukan hubungan antara variabel input dan output yang dikumpulkan dari literatur sumber terbuka. Data asli yang mencakup 45 studi kasus disajikan	Metode *Support Vector Regression* dengan *Particle Swarm Optimization*, proses : 1. Inisialisasi partikel acak PSO 2. Training SVR partikel dan evaluasi *fitness*	Hasil pengujian menunjukkan bahwa metode SVR-PSO menghasilkan akurasi yang cukup memuaskan. Dari 9 kasus yang diujikan nilai MSE rata-rata yang dihasilkan cukup kecil yaitu 0,0146.

	Optimization (Fattahi, 2015)	dalam 36 kasus (80%) digunakan untuk pelatihan dan 9 kasus (20%) digunakan untuk pengujian.	3. Update posisi dan kecepatan partikel 4. Update *fitness*, pBest, dan gBest 5. Lakukan sampai PSO mencapai iterasi maksimal 6. Prediksi pergerakan lereng	
15	*PSO-SVR : A Hybrid Short-term Traffic Flow Forecasting Method* (Hu, et al., 2015)	Data jalur kemacetan lalu lintas, data *time series* di dapat dari Highways Agency di Inggris	Metode *Support Vector Regression* dengan *Particle Swarm Optimization*, proses : 1. *Training* SVR Proses PSO 2. Evaluasi PSO → *Training* SVR 3. Ulangi proses 2 sampai iterasi maksimal 4. Input data *real-time* 5. Prediksi SVR	• Hasil perbandingan nilai RMSE metode PSO-SVR lebih baik dibandingkan dengan metode GA-SA, KNN-LWNN, dan SSA • Untuk menyelesaikan masalah *noise* pada data latih yang dapat mengurangi nilai akurasi, peneliti menggunakan metode prediksi arus lalu lintas dengan *historical momentum* berdasarkan kesamaan pada histori data.
16	Prediksi Tinggi Muka Air (TMA) Untuk Deteksi Dini Bencana Banjir Menggunakan SVR-TVIWPSO (Soebroto, et al., 2015)	Memprediksi tinggi muka air (TMA) untuk deteksi dini bencana banjir menggunakan SVR-TVIWPSO	Menggunakan metode SVR-TVIWPSO, proses : 1. Inisialisasi parameter SVR 2. Iterasi TVIWPSO 3. Hitung kernel RBF *timevalue* 4. Hitung kernel matriks Rij 5. Iterasi kecil SVR 6. Hitung f(x) 7. Iterasi kecil SVR berhenti 8. Cari Gbest 9. Hitung kecepatan	Metode SVR cukup akurat sebagai peramalan Tinggi Muka Air, dengan adanya integrasi SVR yang dioptimalisasi menggunakan TVIWPSO memberikan hasil yang jauh lebih akurat, semua nilai MAE pada 10 data bulanan yang digunakan dalam uji coba SVR-TVIWPSO lebih kecil dibanding MAE pada hasil uji coba SVR dengan kisaran nilai selisih MAE sebesar 0.0740 hingga 0.1323, nilai MAE terkecil didapatkan pada data Juni 2007 dengan nilai sebesar 0.0075 dengan SVR yang dioptimasi menggunakan TVIWPSO. Sedangkan untuk uji coba iterasi SVR nilai MAE paling kecil

			10. Hitung posisi 11. Seleksi *fitness* 12. Iterasi TVIWPSO berhenti 13. Hasil parameter SVR optimal 14. Iterasi besar SVR 15. Hitung Ei 16. Hitung f(x) 17. Iterasi besar SVR berhenti 18. Hasil peramalan TMA	didapat pada jumlah iterasi *training* dan *testing* sebesar 100000 dengan nilai sebesar 0.1426.
17	Optimasi Pemenuhan Kebutuhan Gizi Keluarga Menggunakan Particle Swarm Optimization (Eliantara, et al., 2016)	Optimasi takaran asupan gizi keluarga, data diambil melalui kuesioner	Metode *Particle Swarm Optimization*, proses: 1. Inisialisasi parameter PSO 2. Inisialisasi posisi awal dan kecepatan awal partikel 3. Menghitung *fitness* masing-masing partikel 4. Menentukan pBest dang Best 5. Menghitung kecepatan partikel untuk iterasi selanjutnya 6. Menghitung posisi partikel untuk iterasi selanjutnya 7. Menghitung *fitness* tiap partikel 8. Lakukan proses diatas sampai pada jumlah iterasi maksimum	• Hasil optimal didapatkan pada parameter PSO yaitu jumlah partikel 35 dan bobot inersia 0,4 • Hasi rekomendasi dari sistem mampu memenuhi kebutuhan gizi keluarga dengan selisih ±10% dan dapat menghemat biaya pengeluaran untuk konsumsi sebesar 39,31%.

18	*Optimizing SVR using Local Best PSO for Software Effort Estimation* (Novitasari, et al., 2016)	Desharnais dataset yang berisi 81 proyek perangkat lunak dengan 11 variabel, 9 variabel independen dan 2 variabel dependen	Menggunakan metode *Local Best* SVR-PSO, proses: 1. Normalisasi data 2. Membagi data latih dan data uji 3. Inisialisasi populasi 4. Menghitung nilai *fitness* dengan menghitung nilai rata-rata dari *training* SVR 5. Menentukan pBest baru tiap partikel 6. Menghitung nilai bobot inersia baru 7. Menghitung kecepatan baru tiap partikel 8. Menghitung posisi baru tiap partikel 9. Perulangan berhenti jika mencapai iterasi maksimal, jika belum mencapai iterasi maksimal maka ulangi langkah ke-2	• *Local Best* SVR-PSO menghasilkan nilai *error* terendah dibandingkan dengan PSO-SVR dan T-SVR. • Dari hasil pengujian, *Local Best* SVR-PSO dapat meningkatkan performa dari PSO
19	*Particle Swarm Optimization with Fitness Adjustment Parameters* (Li & Cheng, 2017)	Permasalahan *open vehicle routing problem with time windows* (OVRPTW), *data set* diambil dari Solomon *benchmark*	Menggunakan metode *Particle Swarm Optimization based on fitness performance* (PSOFAP), proses: 1. Inisialisasi populasi partikel swarm 2. Evaluasi nilai *fitness* tiap partikel 3. Update pBest dang Best	• Pada penelitian ini menggunakan metode PSOFAP yang fokus pada pencegahan konvergensi dini dan meningkatkan efisiensi untuk menambah keakuratan solusi. • Hasil pengujian dan analisis statistik menunjukkan bahwa metode PSOFAP memberikan solusi optimal dengan iterasi yang lebih sedikit dan waktu

			4. Update kecepatan tiap partikel 5. Update posisi tiap partikel 6. Cek kondisi berhenti, jika *false* maka lakukan langkah ke-2	komputasi yang lebih cepat dibandingkan SPSO dan APSOVI.
20	*A Simplified Particle Swarm Optimization for Job Scheduling in Cloud Computing* (Attiya & Zhang, 2017)	Data yang digunakan diambil dari *test case* cloudSim dan mengukur *fitness* nya menggunakan MI dan MIPS.	Menggunakan metode Simplified PSO, proses: 1. *Input scheduling problem* 2. Inisialisasi parameter 3. Inisialisasi partikel *swarm* 4. Hitung nilai *fitness* setiap partikel di dalam *swarm* 5. Cari *global best* 6. Masuk ke perulangan 7. *Update* nilai kecepatan setiap partikel 8. *Update* posisi partikel 9. Evaluasi nilai *fitness* tiap partikel 10. Cari *global best* 11. Cek kondisi berhenti, jika kondisi berhenti belum terpenuhi ulangi langkah ke 7.	Dari hasil pengujian untuk berbagai macam *test case* menunjukkan bahwa metode PSO dapat menemukan solusi optimal lebih cepat. Selain itu, secara signifikan lebih baik daripada metode GA dalam hal *makespan*, khususnya pada masalah penjadwalan menengah ke atas.
21	Peramalan Harga Telur Ayam Ras Di Kota Malang Dengan Menggunakan Metode "SVR-PSO" (Usulan)	Data yang digunakan berupa data *time series* yang diambil dari Dinas Perdagangan Kota Malang tahun 2012-2017.	Menggunakan metode SVR-PSO, proses: 1. Inisialisasi parameter PSO dan SVR 2. Inisialisasi kecepatan awal partikel PSO	

			3. Inisialisasi posisi awal partikel PSO 4. *Training* SVR 5. Evaluasi nilai *fitness* 6. Tentukan pBest dan gBest 7. *Update* kecepatan setiap partikel PSO 8. *Update* posisi partikel PSO 9. *Training* SVR 10. Evaluasi nilai *fitness* 11. Update pBest dan gBest 12. Cek kondisi berhenti, jika belum terpenuhi, maka ulangi langkah 7	

Berdasarkan beberapa kajian pustaka mengenai metode SVR dan PSO yang dijabarkan pada Tabel 2.1, dapat disimpulkan bahwa metode SVR adalah metode peramalan yang memiliki akurasi yang lebih baik dibandingkan dengan metode peramalan lainnya seperti BPNN, MDA, dan Logit (Min & Lee, 2005). Parameter inisial SVR sangat sensitif terhadap hasil peramalan. Bagaimana memilih nilai parameter awal SVR yang optimal dan masuk akal merupakan hal yang sangat penting (Lin, et al., 2006). Pemilihan parameter awal SVR dapat dilakukan dengan menggunakan metode optimasi, penulis menggunakan metode PSO untuk mengoptimasi kombinasi parameter SVR yang tepat untuk akurasi peramalan yang lebih baik. Metode PSO merupakan metode optimasi yang cukup sederhana. Kelebihan dari metode ini adalah lebih cepat konvergen dibandingkan metode algoritme genetika (Lu & Geng, 2011). Berdasarkan kajian pustaka diatas, penggunaan PSO untuk mengoptimasi parameter SVR terbukti meningkatkan nilai akurasi peramalan dibandingkan dengan peramalan dengan metode SVR tradisional (Hsieh, et al., 2011).

2.2 Telur ayam ras

Kebutuhan telur ayam ras saat ini mengalami peningkatan sebanding dengan peningkatan kesadaran masyarakat akan pentingnya konsumsi makanan bergizi. Telur ayam ras merupakan salah satu sumber makanan protein hewani yang mudah dijangkau. Semakin meningkatnya kesadaran masyarakat untuk mengkonsumsi makanan bergizi, mengakibatkan meningkatnya permintaan pangan berprotein hewani. Karena pada dasarnya kebutuhan protein hewani tidak dapat digantikan dengan protein lainnya.

Data konsumsi telur ayam ras di Indonesia pada tahun 2009 tercatat sebesar 5.827 kg/kapita dan terjadi peningkatan sebesar 13,24% pada tahun 2010 yang tercatat sebesar 6.709 kg/kapita. Pada segi produksi telur ayam ras di Indonesia menunjukkan produksi telur ayam masih belum stabil dan mengalami fluktuasi. Tahun 2007 produksi sebesar 944.136 ton, tahun 2008 sebesar 955.999 ton, tahun 2009 sebesar 909.519 ton, tahun 2010 sebesar 945.635 ton dan tahun 2011 produksi telur ayam ras sebesar 986.794 ton. Data statistik produksi telur ayam ras di Kota Malang dari tahun 2006 hingga tahun 2010 terus mengalami penurunan. Pada data statistik tercatat produksi telur ayam tertinggi terjadi pada tahun 2006 sebesar 48.307 kg/tahun dan produksi telur ayam terendah terjadi pada tahun 2010 sebesar 38.180 kg/tahun (Badan Pusat Statistik Kota Malang, 2011).

Tabel 2.2 Hasil produksi telur ayam ras di Indonesia

No	Tahun	Produksi (ton)
1	2007	944.136
2	2008	955.999
3	2009	909.519
4	2010	945.635

5	2011	986.794	

(Sumber: BPS Kota Malang)

Banyak faktor yang mempengaruhi elastisitas permintaan telur ayam ras di Kota Malang, salah satunya adalah faktor sosial ekonomi seperti, harga telur ayam ras, pendapatan per kapita, pendidikan dan jenis kelamin, serta banyak anggota keluarga. Nilai elastisitas harga terhadap permintaan telur ayam ras bersifat elastis dengan nilai -2,301 (Febrianto & Putritamara, 2013).

2.3 Peramalan

Untuk menyelesaikan permasalahan di masa yang akan datang yang tidak dapat dipastikan orang biasanya menyelesaikannya dengan model pendekatan. Model pendekatan yang digunakan adalah dengan melakukan pendekatan yang sesuai dengan perilaku ataupun pola data aktual, begitu juga dalam melakukan peramalan. Peramalan diperlukan biasa;nya untuk memperkirakan apa yang akan terjadi di masa yang akan datang, sehingga para pengambil keputusan bisa membuat perencanaan untuk masa depan. Peramalan merupakan suatu prediksi untuk memperkirakan suatu nilai pada periode tertentu yang akan terjadi masa mendatang dengan menggunakan penjelasan secara matematik dan statistik. Peramalan adalah masalah penting yang mencakup banyak bidang pengetahuan termasuk, bisnis dan industri, pemerintahan, ekonomi, lingkungan obat-obatan, sosial, politik, dan keuangan (Montgomery, et al., 2015). Data yang digunakan untuk peramalan adalah data yang relevan yang dapat dijelaskan secara statistik maupun matematik, dan biasanya berupa data yang digunakan dari masa lalu maupun saat ini.

Sebagian besar masalah peramalan melibatkan penggunaan data runtut waktu (*time series*), tetapi ada juga yang menggunakan analisis pola sebab-akibat. Alasan sangat pentingnya peramalan adalah meramalkan masa depan merupakan input penting dalam banyak jenis perencanaan dan pengambilan keputusan (Montgomery, et al., 2015). Peramalan secara umum dapat dibedakan dari beberapa segi tergantung cara pandangnya antara lain (Montgomery, et al., 2015):

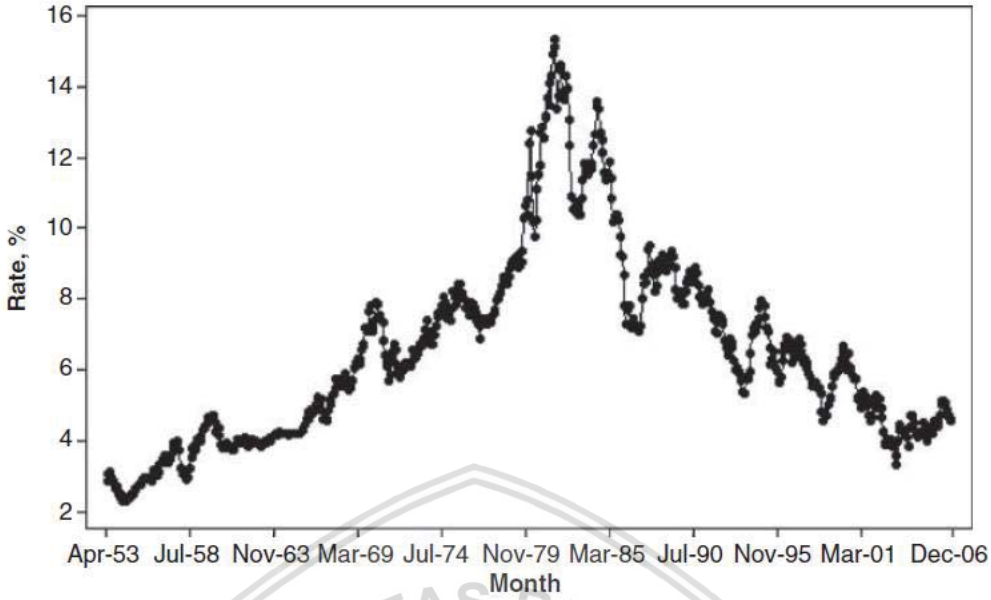
- Peramalan berdasarkan jangka waktu

    1. Peramalan jangka pendek, yaitu peramalan yang dilakukan dalam jangka waktu kurang dari satu tahun. Peramalan seperti ini biasanya digunakan dalam peramalan permintaan konsumen, peramalan curah hujan, penjadwalan kerja, tingkat produktivitas.
    2. Peramalan jangka menengah, yaitu peramalan yang dilakukan dalam jangka waktu rentang satu sampai dua tahun. Peramalan seperti ini digunakan dalam perencanaan penjualan, perencanaan anggaran produksi.
    3. Peramalan jangka panjang, yaitu peramalan yang dilakukan dalam jangka waktu rentang tiga tahun hingga lebih dari tiga tahun. Peramalan seperti

ini diperlukan untuk perencanaan anggaran modal dan perencanaan pembelian produk baru.

- Peramalan berdasarkan metode

1. Peramalan kualitatif yaitu peramalan yang didasarkan atas data kualitatif (non-numerik) yang terjadi di masa lalu. Peramalan seperti ini merupakan peramalan yang bersifat subyektif yaitu hasil bergantung pada opini, pengalaman, dan pengetahuan orang yang mengambil keputusan.
2. Peramalan kuantitatif yaitu peramalan yang didasarkan pada data kuantitatif (numerik) masa lalu. Model peramalan ini meringkas pola pada data dan didapatkan hubungan statistik nilai variabel sebelum dan sekarang. Lalu model ini akan memproyeksikan pola tersebut untuk meramalkan masa depan. Peramalan seperti ini merupakan peramalan yang bersifat obyektif karena hasil peramalan bergantung pada metode yang digunakan. Ada beberapa tiga jenis model peramalan kuantitatif yang umum digunakan, yaitu (Montgomery, et al., 2015):
    a. Model regresi yaitu model peramalan yang memanfaatkan hubungan sebab dan akibat. Model regresi sering disebut sebagai model kausal karena variabel predictor (sebab) diasumsikan untuk menggambarkan kekuatan yang menyebabkan variabel yang diamati menghasilkan *variable of interest* (akibat). Misalnya, untuk meramalkan jumlah penjualan furnitur dengan menggunakan data pembelian rumah sebagai variabel prediktor.
    b. *Smoothing model* biasanya menggunakan fungsi sederhana dari pengamatan sebelumnya dapat memberikan perkiraan *variable of interest* (variabel akibat). Model ini mungkin memiliki dasar statistik formal, tetapi model ini sering digunakan secara heuristik karena model ini memiliki akurasi perkiraan yang memuaskan.
    c. *Time series model* menggunakan statistik data historis untuk menentukan model formal dan kemudian memperkirakan parameter yang tidak diketahui. *Time series* adalah hasil pengamatan berorientasi waktu atau urutan kronologis. Contohnya adalah data penjualan bulanan, harga saham tertentu, dan lain-lain. Berikut contoh grafik data *time series* yang ditunjukkan oleh Gambar 2.1.



(Sumber : US Treasury)

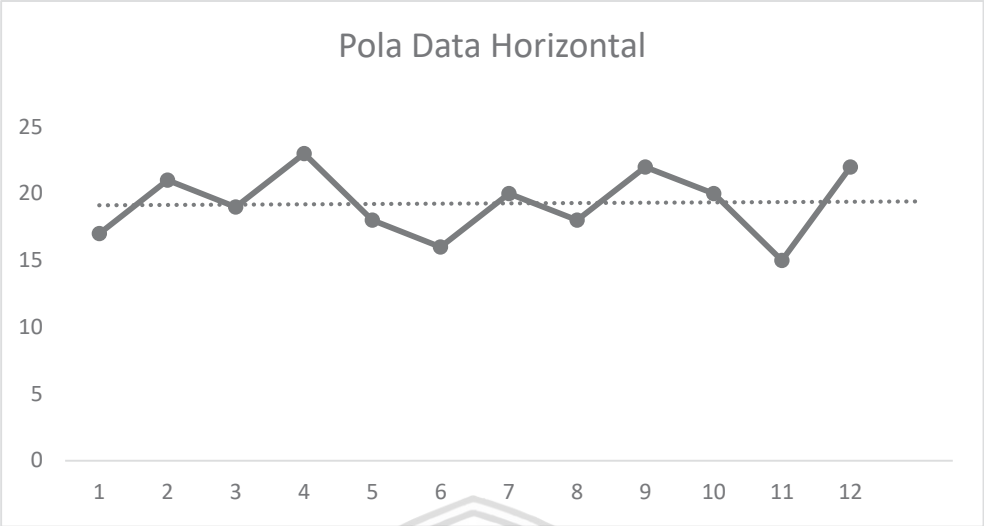
Gambar 2.1 Contoh plot *time series* hasil pemasaran US Treasury Securities dalam 10 tahun

2.4 Time Series

Time series adalah urutan pengamatan pada variabel yang diukur pada titik-titik urut waktu atau selama periode waktu berturut-turut. Pengukuran dapat dilakukan setiap jam, hari, minggu, bulan, atau tahun, atau pada interval reguler lainnya. Pola data adalah faktor penting dalam memahami bagaimana perilaku *time series*. Jika perilaku data tersebut bisa harapkan akan berlanjut di masa depan, maka kita dapat menggunakan pola masa lalu untuk membimbing kita dalam memilih metode peramalan yang tepat. Untuk mengidentifikasi pola dasar dalam data, langkah pertama yang penting adalah membangun plot *time series*. Plot *time series* adalah presentasi grafis dari hubungan antar waktu dan variabel *time series*, waktu berada pada sumbu horizontal dan nilai-nilai rangkaian waktu pada sumbu vertikal. Berikut beberapa tipe pola data yang teridentifikasi didalam plot *time series*, yaitu (Makridakis, et al., 1983):

1. **Pola Horizontal**

   Pola horizontal ada ketika data berfluktuasi di sekitar rata-rata konstan. Plot *time series* untuk data *time series* stasioner akan selalu memperlihatkan pola horizontal. Tapi hanya mengamati pola horizontal bukanlah bukti yang cukup untuk menyimpulkan bahwa *time series* tersebut stasioner. Berikut contoh plot data time series horizontal yang ditunjukkan oleh Gambar 2.2.

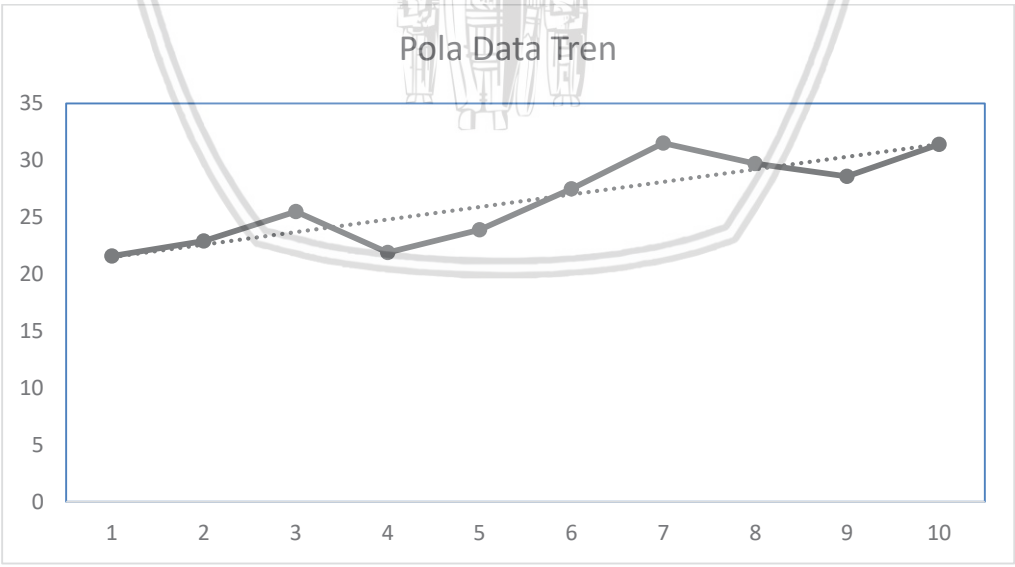


Sumber : (Makridakis, et al., 1983)

Gambar 2.2 Pola data horizontal

2. Pola Tren

Meskipun data *time series* umumnya menunjukkan fluktuasi yang acak, *time series* juga dapat menampilkangerakan bertahap ke nilai yang relatif lebih tinggi atau lebih rendah selama periode waktu yang lebih lama. Jika plot *time series* menunjukkan perilaku seperti ini, bisa dikatakan bahwa ada pola tren. Sebuah tren biasanya hasil dari faktor jangka panjang seperti populasi meningkat atau menurun, perubahan karakteristik demografi populasi, teknologi, dan keinginan konsumen. Berikut contoh plot data time series tren yang ditunjukkan oleh Gambar 2.3.



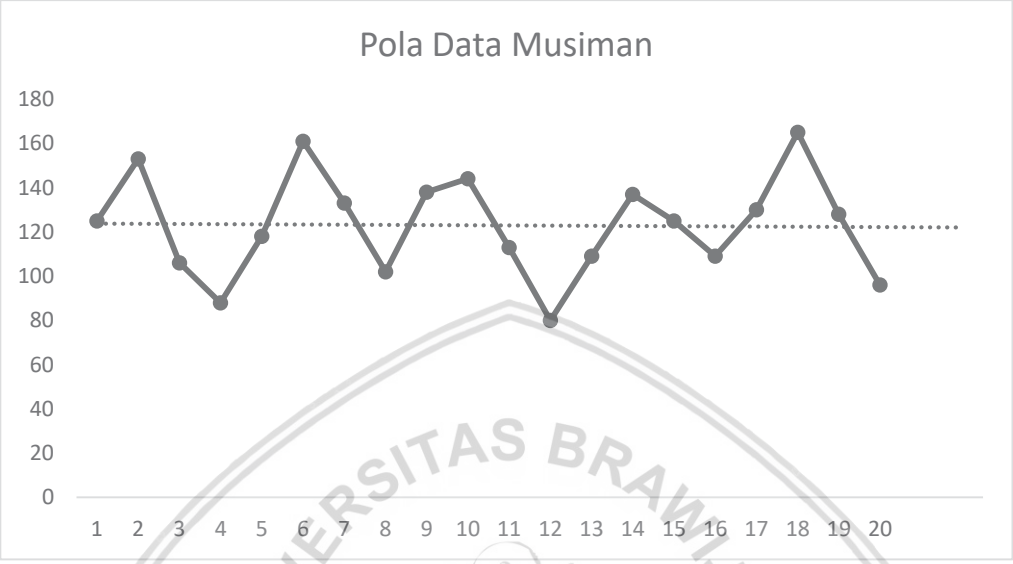
Sumber : (Makridakis, et al., 1983)

Gambar 2.3 Pola data tren

3. Pola Musiman

Tren data *time series* dapat diidentifikasi dengan menganalisis gerakan data di beberapa periode dalam sejarah data. Pola musiman dapat dikenali dengan melihat pola yang sama berulang secara

berurutan untuk periode waktu tertentu. Pergerakan pola musiman dalam *time series* tidak hanya terjadi di dalam satu tahun, data *time series* juga dapat menunjukkan pola musiman kurang dari satu tahun lamanya. Berikut contoh plot data *time series* musiman yang ditunjukkan oleh Gambar 2.4.

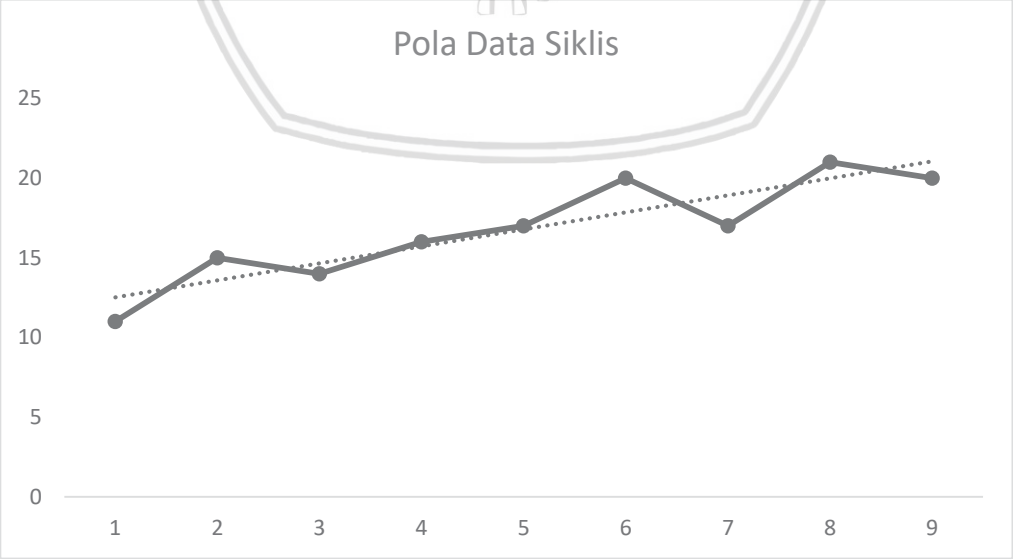


Sumber : (Makridakis, et al., 1983)

Gambar 2.4 Pola data musiman

4. Pola Siklis

Pola siklis ada jika plot *time series* menunjukkan urutan poin bergantian di bawah dan di atas garis tren yang berlangsung lebih dari satu tahun. Seringkali, pola siklis dari *time series* adalah karena siklus bisnis yang bertahun-tahun. Pola siklis sering terjadi karena efek tren jangka panjang yang disebut *trend-cycle effect*. Berikut contoh plot data *time series* siklis yang ditunjukkan oleh Gambar 2.5.



Sumber : (Makridakis, et al., 1983)

Gambar 2.5 Pola data siklis

.

2.5 Normalisasi

Normalisasi adalah teknik scaling atau teknik pemetaan data. Normalisasi memegang peran penting dalam hal *soft computing* dan *cloud computing* dalam memanipulasi data dengan menaikkan atau menurunkan rentang data (*range*) sebelum data digunakan untuk tahapan yang lebih jauh. Dalam peramalan, proses normalisasi adalah salah satu tahap penting untuk meminimalkan variasi dari data dan nilai *error* prediksi. Ada 3 teknik normalisasi yang populer, yaitu (Patro & Sahu, 2015):

1. **Min-Max Normalization**

   Teknik ini memberikan transformasi linier pada rentang data asli dan mempertahankan hubungan antar data asli. Teknik normalisasi ini secara spesifik menyesuaikan data dalam kisaran nilai yang telah ditentukan sebelumnya. Rumus *Min-Max Normalization* ditunjukkan pada Persamaan (2.1)

   $$\boldsymbol{A' = \left(\frac{A - \min value\ of\ A}{\max value\ of\ A - \min value\ of\ A}\right) \times (D - C) + C} \quad \textbf{(2.1)}$$

   Keterangan

   A' : data setelah normalisasi *min-max*

   $[C, D]$: batasan yang ditentukan

2. **Z-Score Normalization**

   Normalisasi jenis ini menggunakan konsep rata-rata dan standar deviasi. *Z-Score Normalization* tidak dapat bertransformasi lagi ke data asli. Rumus normalisasi *Z-Score* ditunjukkan pada Persamaan (2.2) dan rumus standar deviasi ditunjukkan pada Persamaan (2.3).

   $$\boldsymbol{v_i' = \frac{v_i - E}{std(E)}} \quad \textbf{(2.2)}$$

   Keterangan

   v_i' : data setelah *normalisasi z-score*

   v_i : data sebelum dinormalisasi

   E : rata-rata data

   $std(E)$: standar deviasi data

   $$\boldsymbol{std = \left(\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2\right)^{\frac{1}{2}}} \quad \textbf{(2.3)}$$

   Keterangan

   n : banyaknya data

   x_i : data ke – i

   $\bar{x}$: rata-rata data x

3. **Decimal Scaling Normalization**

Teknik ini menormalisasikan data dalam rentang nilai -1 dan 1. *Decimal scaling normalization* dihitung menggunakan Persamaan (2.4)

$$\boldsymbol{v^i} = \frac{\boldsymbol{v}}{\boldsymbol{10^j}} \quad \textbf{(2.4)}$$

Keterangan

$\boldsymbol{v^i}$: data setelah normalisasi *decimal scaling*

v : data sebelum dinormalisasi

j : integer terkecil $MAX(|vi|) < 1$

2.6 *Support vector regression* (SVR)

Support Vector Regression (SVR) merupakan penerapan *Support Vector Machine* (SVM) untuk kasus regresi. Dalam kasus regresi output berupa bilangan riil atau kontinyu. SVR merupakan metode yang dapat mengatasi *overfitting*, sehingga akan menghasilkan performansi yang bagus (Smola, 2003).

Metode SVR menerapkan konsep dari metode SVM yang memetakan masukan vektor secara non linier ke dalam ruang fitur berdimensi tinggi (Cortes & Vapnik, 1995). Metode SVR menerapkan prinsip *Structural Risk Minimisation* (SRM) dimana metode SVR fokus menemukan *hyperplane* yang optimal dan meminimalisasi kesalahan antara data latih dan *ε-insentive loss function* (Ju & Hong, 2013). Pada ruang fitur menggunakan fungsi regresi linier dengan Persamaan (2.5)

$$\boldsymbol{f(x)} = \boldsymbol{(w.x)} + \boldsymbol{b} \quad \textbf{(2.5)}$$

Keterangan

w : vektor pembobot

x : data

b : bias

Penyelesaian masalah non-linear dapat diselesaikan menggunakan metode *Sequential Learning* SVR. Sebelum masuk ke dalam metode tersebut, dilakukan proses normalisasi data terlebih dahulu dengan dengan tujuan meningkatkan akurasi regresi dengan mengurangi tingkat *error* dalam komputasi. Proses normalisasi dilakukan sebelum melakukan pelatihan menggunakan SVR.

2.6.1 Fungsi kernel

Fungsi kernel merupakan bagian terpenting dari metode SVR. Kernel adalah sebuah algoritme yang digunakan untuk analisis dan pengenalan pola. Tugas umum dari analisis dan pengenalan pola adalah menemukan dan mempelajari hubungan umum (misalnya klaster, klasifikasi, korelasi, komponen utama) pada data seperti *sequence*, teks dokumen, vektor, citra, graf, dan lain-lain (Souza, 2010).

Metode kernel memetakan data ke ruang dimensi yang lebih tinggi, sehingga data lebih mudah dipisahkan atau lebih terstruktur. Dengan penggunaan fungsi kernel, algoritme dapat dengan mudah dibawa ke ruang dimensi yang lebih tinggi tanpa harus secara eksplisit memetakan titik input data. Hal ini tentu memudahkan, karena terkadang ruang fitur dimensi bersifat *infinite-dimensional* dan tidak bisa dihitung. Berikut jenis-jenis fungsi kernel yang sering digunakan dalam analisis pola metode SVR (Souza, 2010):

1. Kernel Linier

Kernel linier merupakan fungsi kernel paling sederhana. Kernel linier didapatkan dari inner product <x,y> ditambah dengan konstanta c. Metode yang menggunakan fungsi kernel linier biasanya ekuivalen dengan data bandingan non-kernelnya. Kernel linier dihitung menggunakan Persamaan (2.6) berikut.

$$k(x, y) = x^T y + c \quad \textbf{(2.6)}$$

Keterangan

$k(x, y)$: fungsi kernel

x : produk x

y : produk y

c : konstanta

2. Kernel Polinomial

Kernel polinomial adalah jenis kernel non-stasioner. Kernel ini baik digunakan untuk permasalahan yang dimana semua data latih telah dinormalisasi. Parameter yang dapat disesuaikan adalah *alpha,* konstanta c, dan derajat polinomial d. Kernel polinomial dihitung menggunakan Persamaan (2.7) berikut.

$$k(x, y) = (\alpha x^T + c)^d \quad \textbf{(2.7)}$$

Keterangan

$k(x, y)$: fungsi kernel

x : produk x

d : derajat polinomial

c : konstanta

3. Kernel Gaussian RBF

Kernel Gaussian adalah contoh dari kernel *radial basis function* (RBF). Parameter *sigma* (σ) dapat disesuaikan dan memainkan peran besar dalam kinerja kernel, sehingga harus diinisialisasikan sesuai dengan masalah yang akan diselesaikan. Jika terlalu tinggi, eksponensial akan berperilaku hampir secara linier dan proyeksi dimensi yang lebih tinggi akan mulai kehilangan daya non-liniernya. Di sisi lain, jika diremehkan, fungsinya akan kekurangan regularisasi dan batas keputusan akan sangat sensitif terhadap *noise* dalam data latih. Kernel *Gaussian* RBF dihitung menggunakan Persamaan (2.8) berikut.

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

Keterangan

$k(x,y)$: fungsi kernel

x : produk x

y : produk y

σ : *sigma* atau *noise*

4. Kernel Eksponensial

Kernel eksponensial terikat erat dengan kernel *Gaussian,* hanya saja kuadrat pada jarak dibuang. Kernel ini juga termasuk jenis kernel *radial basis*. Kernel eksponensial dihitung menggunakan Persamaan (2.9) berikut.

$$k(x,y) = \exp\left(-\frac{\|x-y\|}{2\sigma^2}\right) \quad \textbf{(2.9)}$$

Keterangan

$k(x,y)$: fungsi kernel

x : produk x

y : produk y

σ : *sigma* atau *noise*

5. Kernel Laplacian

Kernel *laplacian* sama halnya dengan kernel eksponensial, hanya saja pada kernel *laplacian* kurang sensitif terhadap parameter *sigma*. Kernel ini juga termasuk jenis kernel *radial basis*. Penting untuk dicatat bahwa pengmatan yang dibuat tentang parameter *sigma* untuk kernel *Gaussian* juga berlaku untuk kernel eksponensial dan kernel *laplacian*. Kernel *laplacian* dihitung menggunakan Persamaan (2.10) berikut.

$$k(x,y) = exp\left(-\frac{\|x-y\|}{\sigma}\right) \quad \textbf{(2.10)}$$

Keterangan

$k(x,y)$: fungsi kernel

x : produk x

y : produk y

σ : *sigma* atau *noise*

6. Kernel ANOVA

Kernel ANOVA juga merupakan kernel RBF seperti halnya kernel *Gaussian*. Kernel jenis ini bekerja baik pada permasalahan regresi multidimensional. Kernel ANOVA dihitung menggunakan Persamaan (2.11) berikut.

$$k(x,y) = \sum_{k=1}^{n} exp(-\sigma\,(x^k - y^k)^2)^d \quad \textbf{(2.11)}$$

Keterangan

$k(x, y)$: fungsi kernel

x : produk x

y : produk y

σ : sigma atau noise

k : derajat multidimensional

d : derajat polinomial

7. Kernel Hyperbolic Tangent (Sigmoid)

Kernel *hyperbolic tangent* juga dikenal sebagai kernel *sigmoid* atau kernel *multilayer perceptron* (MLP). Kernel *sigmoid* berasal dari bidang *Neural Network*, di mana fungsi *sigmoid* bipolar sering digunakan sebagai fungsi aktivasi untuk neuron buatan. Sangat menarik untuk dicatat bahwa model SVM menggunakan fungsi kernel *sigmoid* setara dengan dua *layer*, perceptron pada jaringan syaraf tiruan. Kernel ini cukup populer untuk mendukung mesin vektor karena asalnya dari teori jaringan saraf. Selain itu, meskipun hanya kondisional positif yang pasti, ia telah terbukti berkinerja baik dalam praktiknya. Ada dua parameter yang dapat disesuaikan dalam kernel *sigmoid*, *alpha* dan konstanta c. Nilai yang umum untuk *alpha* adalah 1 / N, di mana N adalah dimensi data. Kernel *hyperbolic tangent* dihitung menggunakan Persamaan (2.12) berikut.

$$k(x, y) = \tan(\alpha x^T + c) \quad \textbf{(2.12)}$$

Keterangan

$k(x, y)$: fungsi kernel

x : produk x

y : produk y

c : konstanta

8. Kernel Rational Quadratic

Kernel *rational quadratic* kurang intensif secara komputasi daripada kernel Gaussian dan dapat digunakan sebagai alternatif ketika menggunakan Gaussian menjadi terlalu rumit. Kernel *rational quadratic* dihitung menggunakan Persamaan (2.13) berikut.

$$k(x, y) = 1 - \frac{\|x-y\|^2}{\|x-y\|^2+c} \quad \textbf{(2.13)}$$

Keterangan

$k(x, y)$: fungsi kernel

x : produk x

y : produk y

c : konstanta

9. Kernel Multiquadric

Kernel *Multiquadric* dapat digunakan dalam situasi yang sama dengan kernel Rasional Kuadratik. Seperti halnya dengan kernel Sigmoid, ini juga merupakan contoh dari kernel definitif non-positif. Kernel *multiquadric* dihitung menggunakan Persamaan (2.14) berikut.

$$k(x,y) = \sqrt{\|x-y\|^2 + c^2} \quad \textbf{(2.14)}$$

Keterangan

$k(x,y)$: fungsi kernel

x : produk x

y : produk y

c : konstanta

10. Kernel Inverse Multiquadric

Seperti halnya kernel Gaussian, ia menghasilkan matriks kernel dengan pangkat penuh dan dengan demikian membentuk ruang fitur dimensi tak terbatas. Kernel *inverse multiquadric* dihitung menggunakan Persamaan (2.15) berikut.

$$k(x,y) = \frac{1}{\sqrt{\|x-y\|^2+c^2}} \quad \textbf{(2.15)}$$

Keterangan

$k(x,y)$: fungsi kernel

x : produk x

y : produk y

c : konstanta

11. Kernel Circular

Kernel *circular* digunakan dalam aplikasi geostatik. Ini adalah contoh dari kernel stasioner isotropik dan pasti positif dalam R^2. Kernel *circular* dihitung menggunakan Persamaan (2.16) berikut.

$$k(x,y) = \frac{2}{\pi} arc\ cos\left(-\frac{\|x-y\|}{\sigma}\right) - \frac{2}{\pi}\frac{\|x-y\|}{\sigma}\sqrt{1-\left(\frac{\|x-y\|}{\sigma}\right)^2} \quad \textbf{(2.16)}$$

$$jika\ \|x-y\| < \sigma, maka\ 0$$

Keterangan

$k(x,y)$: fungsi kernel

x : produk x

y : produk y

σ : sigma atau noise

12. Kernel Spherical

Kernel *spherical* sama halnya dengan kernel *circular*, namun bersifat pasti positif pada R^3. Kernel *spherical* dihitung menggunakan Persamaan (2.17) berikut.

$$k(x,y) = 1 - \frac{2}{3}\frac{\|x-y\|}{\sigma} + \frac{1}{2}\left(\frac{\|x-y\|}{\sigma}\right)^3 \quad \textbf{(2.17)}$$

$$jika\ \|x-y\| < \sigma, maka\ 0$$

Keterangan

$k(x,y)$: fungsi kernel

x : produk x

y : produk y

σ : sigma atau noise

13. Kernel Power

Kernel *power* juga dikenal sebagai kernel triangular (tidak terukur). Kernel ini adalah contoh dari kernel skala-invariant dan juga hanya kondisional positif yang pasti. Kernel *power* dihitung menggunakan Persamaan (2.18) berikut.

$$k(x,y) = -\|x-y\|^d \quad \textbf{(2.18)}$$

Keterangan

$k(x,y)$: fungsi kernel

x : produk x

y : produk y

d : derajat power

14. Kernel Log

Kernel *Log* tampaknya sangat menarik untuk gambar, tetapi hanya kondisional positif yang pasti. Kernel *log* dihitung menggunakan Persamaan (2.19) berikut.

$$k(x,y) = -\log(\|x-y\|^d + 1) \quad \textbf{(2.19)}$$

Keterangan

$k(x,y)$: fungsi kernel

x : produk x

y : produk y

d : derajat power

15. Kernel Spline

Kernel *spline* dikenal sebagai *piece-wise* kubik polinomial. Kernel *spline* dihitung menggunakan Persamaan (2.20) berikut.

$$k(x,y) = 1 + xy + xy\min(x,y) - \frac{x+y}{2}\min(x,y)^2 + \frac{1}{3}\min(x,y)^3 \quad \textbf{(2.20)}$$

Keterangan

$k(x,y)$: fungsi kernel

x : produk x

y : produk y

16. Kernel B-Spline

Kernel *B-Spline* didefinisikan pada interval [-1, 1]. Kernel *B-spline* dihitung menggunakan Persamaan (2.21) berikut.

$$k(x,y) = \prod_{p=1}^{d} B_{2n+1}(x_p - y_p) \quad \textbf{(2.21)}$$

Alternatifnya, B_n bisa didapatkan dengan menggunakan rumus eksplisit pada Persamaan (2.22) berikut.

$$B_n(x) = \frac{1}{n!}\sum_{k=0}^{n+1}\binom{n+1}{k}(-1)^k\left(x + \frac{n+1}{2} - k\right)_+^n \quad \textbf{(2.22)}$$

$$x_+^d = \begin{cases} x^d, jika\ x > 0 \\ 0, lain - lain \end{cases}$$

Keterangan

$k(x,y)$: fungsi kernel

x : produk x

y : produk y

17. Kernel Bessel

Kernel Bessel terkenal dalam teori ruang fungsi kehalusan fraksional. Kernel *Bessel* dihitung menggunakan Persamaan (2.23) berikut.

$$k(x,y) = \frac{J_{v+1}(\sigma\|x-y\|)}{\|x-y\|^{-n(v+1)}} \quad \textbf{(2.23)}$$

Keterangan

$k(x,y)$: fungsi kernel

x : produk x

y : produk y

σ : sigma atau noise

18. Kernel Cauchy

Kernel *cauchy* berasal dari distribusi *Cauchy*. Ini adalah kernel berekor panjang dan dapat digunakan untuk memberikan pengaruh jangka panjang dan kepekaan atas ruang dimensi tinggi. Kernel *cauchy* dihitung menggunakan Persamaan (2.24) berikut.

$$k(x,y) = \frac{1}{1 + \frac{\|x-y\|^2}{\sigma^2}} \quad \textbf{(2.24)}$$

Keterangan

$k(x, y)$: fungsi kernel

x : produk x

y : produk y

σ : sigma atau noise

19. Kernel Chi-Square

Kernel *chi-square* berasal dari distribusi *chi-square*. Versi kernel ini hanya *conditionally positive-definite* (CPD) dan cocok untuk digunakan dengan metode selain *support vector machine* (SVM). Kernel *chi-square* dihitung menggunakan Persamaan (2.25) berikut.

$$k(x, y) = \sum_{i=1}^{n} \frac{2x_i y_i}{(x_i + y_i)} \quad \textbf{(2.25)}$$

Keterangan

$k(x, y)$: fungsi kernel

x : produk x

y : produk y

n : banyak data

20. Kernel Histogram Intersection

Kernel *histogram intersection* dikenal sebagai kernel min dan telah terbukti berguna pada klasifikasi citra. Kernel *histogram intersection* dihitung menggunakan Persamaan (2.26) berikut.

$$k(x, y) = \sum_{i=1}^{n} \min(x_i, y_i) \quad \textbf{(2.26)}$$

Keterangan

$k(x, y)$: fungsi kernel

x : produk x

y : produk y

n : banyak data

21. Kernel Generalized Histogram Intersection

Kernel *generalized histogram intersection* dibangun berdasarkan pada kernel *histogram intersection* untuk klasifikasi citra tetapi berlaku dalam konteks yang jauh lebih besar. Kernel *generalized histogram intersection* dihitung menggunakan Persamaan (2.27) berikut.

$$k(x, y) = \sum_{i=1}^{m} \min(|x_i|^{\alpha}, |y_i|^{\beta}) \quad \textbf{(2.27)}$$

Keterangan

$k(x, y)$: fungsi kernel

x : produk x

α : derajat alfa

β : derajat beta

22. Kernel Generalized T-Student

Kernel *generalized t-student* telah dikenal sebagai Kernel Mercel, sehingga memiliki matriks Kernel semi-definit positif. Kernel *generalized t-student* dihitung menggunakan Persamaan (2.28) berikut.

$$k(x,y) = \frac{1}{1+\|x-y\|^d} \quad \textbf{(2.28)}$$

Keterangan

$k(x,y)$: fungsi kernel

x : produk x

y : produk y

d : derajat polinomial

23. Kernel Bayesian

Kernel *bayesian* dihitung menggunakan Persamaan (2.29) berikut.

$$k(x,y) = \prod_{l=1}^{N} K_l(x_l, y_l) \quad \textbf{(2.29)}$$

$$K_l(a,b) = \sum_{c\in\{0;1)} P(Y=c|X_l=a)\, P(Y=c|X_l=b)$$

Keterangan

$k(x,y)$: fungsi kernel

$K(x,y)$: probabilitas K

x : produk x

y : produk y

N : jumlah data

24. Kernel Wavelet

Kernel Wavelet berasal dari teori Wavelet, dimana a dan c adalah dilatasi wavelet dan koefisien translation. Versi translation-invarian dari kernel ini dapat dilihat pada Persamaan (2.30) berikut.

$$k(x,y) = \prod_{i=1}^{N} h\left(\frac{x_i - y_i}{a}\right) \quad \textbf{(2.30)}$$

$$h(x) = \cos(1{,}75x)\, exp\left(-\frac{x^2}{2}\right)$$

Keterangan

$k(x,y)$: fungsi kernel

x : produk x

y : produk y

$h(x)$: fungsi wavelet

2.6.2 Sequential learning

Terdapat penyempurnaan persamaan fungsi regresi nonlinier berdasarkan penelitian sebelumnya yaitu terdapat pada nilai *bias*. Nilai b dapat digantikan dengan vektor skalar (λ) sehingga menghasilkan nilai regresi yang lebih baik. Berikut langkah-langkah proses *sequential learning Support Vector Regression* (Vijayakumar & Wu, 1999):

1. Inisialisasi parameter SVR. Parameter SVR diantaranya yaitu parameter λ (variabel skalar atau *lambda*), cLR (*constanta learning rate*), ε (nilai *epsilon*), C (*complexity*), inisialisasi $\alpha_{i*}\ dan\ \alpha_i$ = 0 dan iterasi maksimum.
2. Menghitung matriks *hessian* dengan Persamaan (2.31):

$$\boldsymbol{R_{ij} = (k(x_i, x_j) + \lambda^2)} \quad \textbf{(2.31)}$$

Keterangan

R_{ij} : Matriks *hessian* baris ke-i kolom ke-j

$k(x_i, x_j)$: Fungsi Kernel

λ^2 : varibel skalar atau *lambda*

Keluaran dari matriks *hessian* adalah nilai parameter gamma (γ) yang digunakan pada tahap selanjutnya yaitu proses *sequential learning*. Nilai parameter *gamma* (γ) dapat dihitung dengan Persamaan (2.32):

$$\boldsymbol{\gamma = \frac{cLR}{\max(matriks\ hessian)}} \quad \textbf{(2.32)}$$

Keterangan

γ : parameter *gamma*

cLR : *constanta learning rate*

3. Pada data latih lakukan perhitungan Tahap a, b, dan c untuk tiap data *training point* yaitu:
   a. Hitung nilai *error* dengan Persamaan (2.33) :

$$\boldsymbol{E_i = y_i - \sum_{i=1}^{n}(\alpha_i^* - \alpha_i)R_{ij}} \quad \textbf{(2.33)}$$

Keterangan

E_i : Nilai *error* ke-i

y_i : Nilai aktual

α_{i*}, : Nilai *Lagrange multiplier*

R_{ij} : Matriks *hessian* baris ke-i kolom ke-j

i, : indeks

b. Hitung perubahan nilai *Lagrange multiplier* yang merupakan batas-batas untuk menentukan nilai *Lagrange multipler* yang terbaru dengan Persamaan (2.34) dan Persamaan (2.35).

$$\boldsymbol{\delta_{ai*} = MIN\{MAX(\gamma(E_i - \varepsilon), -\alpha_i^*), C - \alpha_i^*\}} \quad \textbf{(2.34)}$$

$$\boldsymbol{\delta_{\alpha i} = MIN\{ MAX(\gamma(-E_i - \varepsilon), -\alpha_i), C - \alpha_i\}} \quad \textbf{(2.35)}$$

Keterangan

$\boldsymbol{\delta_{ai*}}$: perubahan nilai batas atas

$\boldsymbol{\delta_{\alpha i}}$: perubahan nilai batas bawah

E_i : Nilai *error* ke-i

γ : Nilai *gamma*

ε : *epsilon*

C: konstanta

$\boldsymbol{\alpha_i^*}$: batas atas *lagrange multiplier*

$\boldsymbol{\alpha_i}$: batas bawah *lagrange multiplier*

c. Hitung nilai *Lagrange multiplier* yang baru yaitu *update* nilai α_i dan α_i^* dengan Persamaan (2.36) dan Persamaan (2.37).

$$\boldsymbol{\alpha_i^*(baru) = \delta_{ai*} + \alpha_{i*}} \quad \textbf{(2.36)}$$

$$\boldsymbol{\alpha_i(baru) = \delta_{ai} + \alpha_i} \quad \textbf{(2.37)}$$

Keterangan

$\boldsymbol{\delta_{ai*}}$: perubahan nilai batas atas

$\boldsymbol{\delta_{\alpha i}}$: perubahan nilai batas bawah

$\boldsymbol{\alpha_i^*}$: batas atas *lagrange multiplier*

$\boldsymbol{\alpha_i}$: batas bawah *lagrange multiplier*

4. Langkah ke-3 diulangi sampai iterasi maksimum yang telah diinisialisasi di awal, atau telah mencapai konvergensi dengan syarat ditunjukkan pada Persamaan (2.38)

$$\mathrm{MAX}(|\boldsymbol{\delta_{\alpha i}^*}|) < \varepsilon \; dan \; \mathrm{MAX}(|\boldsymbol{\delta_{\alpha i}}|) < \varepsilon \quad \textbf{(2.38)}$$

Keterangan

$\boldsymbol{\delta_{ai*}}$: perubahan nilai batas atas

$\boldsymbol{\delta_{\alpha i}}$: perubahan nilai batas bawah

ε : *epsilon*

5. Hitung fungsi regresi dengan Persamaan (2.39)

$$\boldsymbol{f(x) = \sum_{i=1}^{l}(\alpha_{i*} - \alpha_i)(K(x_i, x_j) + \lambda^2)} \quad \textbf{(2.39)}$$

Keterangan

$f(x)$: nilai regresi

$K(x_i, x_j)$: Nilai kernel

$\boldsymbol{\alpha_i^*}$: batas atas *lagrange multiplier*

$\boldsymbol{\alpha_i}$: batas bawah *lagrange multiplier*

2.7 *Particle swarm optimization* (PSO)

Optimasi (*optimization*) diartikan sebagai proses pemilihan solusi dari beberapa pilihan solusi yang memenuhi sejumlah batasan (*constraint*). Misalkan pada permasalahan TSP pencarian jalur terpendek dalam mengunjungi beberapa tempat atau kota. Pada kasus TSP ini tentu saja terdapat banyak alternatif jalur (solusi). Solusi yang terpilih disesuaikan dengan tujuan (*objective*) dari permasalahan ini, misalkan memilih jalur terpendek atau jalur dengan waktu tempuh tercepat. Batasan yang ada misalkan setiap kota harus dikunjungi tepat satu kali (Mahmudy, 2015).

Particle Swarm Optimization (PSO) adalah sebuah teknik evolutionary computation yang dikembangkan oleh Kennedy dan Eberhart pada tahun 1995. Konsep PSO berasal dari penyederhanaan sistem sosial. Tujuan PSO adalah untuk secara grafis mensimulasikan pergerakan burung yang anggun, tetapi tidak dapat diprediksi. Simulasi awal dimodifikasi untuk mencocokkan kecepatan tetangga terdekat, menghilangkan variabel pendukung, dan menggabungkan teknik pencarian multidimensional dengan percepatan berdasarkan jarak. Konsep PSO mirip dengan algoritme genetika dimana sistem diinisialisasi dengan sebuah populasi dari beberapa solusi acak. Perbedaannya dengan GA, pada PSO setiap solusi acak tersebut memiliki sebuah kecepatan dan solusi potensial yang disebut partikel, dimana partikel ini akan "diterbangkan" melalu ruang pencarian (Eberhart & Shi, 2001).

Konsep PSO adalah, pada setiap langkah (iterasi), partikel mengubah kecepatannya, setiap partikel berjalan menuju lokasi pBest dan gBest-nya. Bobot percepatan partikelnya ditentukan secara acak. Setiap partikel akan melacak koordinatnya (berisi nilai *fitness*) di ruang pencarian yang terkait dengan solusi terbaik yang telah dicapai sejauh ini. Nilai *fitness* pada koordinat solusi terbaik disebut PBest. Nilai terbaik yang akan dilacak pada versi global PSO adalah nilai *fitness* terbaik secara keseluruhan dan lokasinya yang diperoleh sejauh ini oleh setiap partikel dalam populasi. Lokasi ini disebut gBest (Eberhart & Shi, 2001).

PSO memiliki dua fungsi utama yaitu *update* kecepatan partikel dan *update* posisi partikel. Update kecepatan dan posisi partikel akan terus dilakukan sehingga partikel akan terus dipercepat mendekati posisi partikel terbaik sebelumnya dan terbaik global sampai kondisi minimum yang *error* tercapai (Pei-you & Lu, 2013).

Berikut langkah-langkah proses untuk implementasi versi global dari PSO (Eberhart & Shi, 2001):

1. Inisialisasi populasi partikel dan kecepatannya secara acak pada dimensi d di ruang pencarian menggunakan Persamaan (2.40) berikut ini.

$$x = x_{min} + rand[0,1] \times (x_{max} - x_{min}) \quad \textbf{(2.40)}$$

   Keterangan

   x : posisi partikel

   x_{min} : batas minimal partikel dalam ruang pencarian

   x_{max} : batas maksimal partikel dalam ruang pencarian.

2. Untuk setiap partikel, evaluasi nilai *fitness*-nya.
3. Bandingkan nilai *fitness* partikel dengan nilai *PBest* partikel. Jika nilai *fitness* lebih baik daripada nilai *PBest*, maka atur nilai *PBest* partikel sama dengan nilai *fitness* dan lokasi PBest sama dengan lokasi partikel saat ini.
4. Bandingkan evaluasi *fitness* dengan nilai terbaik populasi sebelumnya. Jika nilai terbaik saat ini lebih baik dari nilai *GBest*, maka atur *GBest* sama dengan posisi dan nilai *fitness* saat ini.
5. Update kecepatan partikel menggunakan Persamaan (2.41) berikut ini.

$$\boldsymbol{v'_{id} = \omega . v_{id} + c_1 r_1 (p_{id} - x_{id}) + c_2 r_2 (p_{gd} - x_{id})} \quad \textbf{(2.41)}$$

   Keterangan

   v'_{id} : kecepatan partikel saat ini.

   c_1 dan c_2 : koefisien akselerasi.

   r_1 dan : : angka acak yang dipilih dari rentang [0,1].

   ω merupakan bobot inersia.

   x_{id}: posisi.

   p_{id} : posisi terbaik sebelumnya atau *Pbest*.

   v_{id} : kecepatan sebelumnya dari partikel.

   p_{gd} : posisi terbaik diantara seluruh partikel pada dimensi pencarian atau Gbest.

6. Update posisi partikel menggunakan Persamaan (2.42) berikut ini.

$$\boldsymbol{x'_{id} = x_{id} + v'_{id}} \quad \textbf{(2.42)}$$

   Keterangan

   x'_{id} ; posisi partikel saat ini.

   v'_{id} : kecepatan partikel saat ini.

x_{id} : posisi sebelumnya.

7. Ulangi langkah 2 sampai iterasi maksimal.

Untuk mengatasi masalah konvergensi dini dimana pergerakan partikel hanya terjadi pada area ruang pencarian solusi lokal, maka perlu adanya pengurangan secara linear bobot inersia ω dari nilai maksimum ke nilai minimum untuk setiap penambahan iterasi dengan Persamaan (2.43) (Lu & Geng, 2011).

$$\boldsymbol{\omega = \omega_{max} - iterasi \left(\frac{\omega_{min} - \omega_{max}}{iterasi\ max}\right)} \quad \textbf{(2.43)}$$

Keterangan :

ω_{max} dan ω_{min} biasanya diatur pada nilai 0,9 dan 0,1

Selain pengurangan bobot inersia ω, dilakukan pula perubahan pada nilai koefisien c_1 dan c_2 berdasarkan waktu atau disebut *Time-Varying Acceleration Coefficients* (TVAC). Tujuan dengan perubahan koefisien ini akan mempengaruhi komponen kognitif dan sosial sehingga pada awal PSO partikel akan bergerak lebih leluasa dalam ruang pencarian dibandingkan bergerak hanya menuju populasi terbaik, dan semakin menuju akhir optimasi dengan perubahan koefisien akan memungkinkan partikel akan berkumpul dalam global optimal (Ratnaweera, et al., 2004). Nilai c_{1i}, $c_{1f,2i}$, dan c_{2f} ditetapkan pada 2.5, 0.5, 0.5 dan 2.5 (Ratnaweera, et al., 2004). Berikut merupakan Persamaan (2.44) dan Persamaan (2.45) untuk menghitung perubahan nilai c_1 dan c_2 :

$$\boldsymbol{c_1 = (c_{1f} - c_{1i}) \frac{iterasi}{iterasi\ max} + c_{1i}} \quad \textbf{(2.44)}$$

$$\boldsymbol{c_2 = (c_{2f} - c_{2i}) \frac{iterasi}{iterasi\ max} + c_{2i}} \quad \textbf{(2.45)}$$

Keterangan

c_{1i}, c_{1f}, c_{2i}, dan c_{2f} adalah konstan

iterasi merupakan nilai iterasi sekarang

iterasi max adalah maksimum iterasi.

Pada *update* kecepatan untuk *real code* PSO digunakan teknik pembatasan atau disebut *velocity clamping* yang ditentukan berdasarkan nilai minimum dan maksimum setiap dimensi dari representasi solusi partikel. Batasan kecepatan atau *threshold* yang digunakan adalah seperti Persamaan (2.46) dan Persamaan (2.47) (Marini & Walczak, 2015).

$$Jika\ v_{ij}^{t+1} > vmax, maka\ v_{ij}^{t+1} = vmax_j \quad \textbf{(2.46)}$$

$$\boldsymbol{Jika\ v_{ij}^{t+1} < vmax, maka\ v_{ij}^{t+1} = -vmax_j} \quad \textbf{(2.47)}$$

Setiap partikel memiliki nilai *fitness* dimana semakin besar nilai *fitness* maka semakin baik partikel tersebut untuk dijadikan solusi. Nilai *fitness* untuk optimasi

permasalahan SVR didapatkan dari perhitungan evaluasi model SVR. Nilai *fitness* dapat diperoleh menggunakan tingkat *error* dari proses evaluasi yang dilakukan seperti pada Persamaan (2.48) berikut (Rodriguez & Ferreire, 2009).

$$\boldsymbol{fitness} = \frac{\mathbf{1}}{\mathbf{1}+\boldsymbol{MAPE}} \quad \textbf{(2.48)}$$

Keterangan:

MAPE adalah *Mean Absolute Percentage Error*

Secara umum, PSO sama halnya dengan algoritme evolusi lainnya bisa diaplikasikan untuk menyelesaikan permasalahan optimasi dan permasalahan yang bisa dioptimasi. Beberapa area implementasi algoritme PSO yang paling memiliki potensi adalah perancangan sistem, optimasi *multi-objective*, klasifikasi, pengenalan pola, pemodelan sistem biologi, penjadwalan, *signal processing*, pengembangan *game* dan aplikasi robotik, pengambilan keputusan.

2.8 Optimasi *Support Vector Regression* (SVR) dengan *Particle Swarm Optimization* (PSO)

Optimasi parameter SVR dengan menggunakan PSO bertujuan untuk menghasilkan tingkat *error* paling rendah untuk peramalan harga telur ayam. Parameter yang akan dioptimasi adalah parameter *lambda* (λ), parameter C, parameter cLR, dan parameter *epsilon* (ε).

Proses optimasi terdiri dari 4 tahapan utama, yaitu tahapan normalisasi, tahapan pelatihan SVR, tahapan PSO, dan tahapan pengujian SVR. Pada tahapan normalisasi, data runut waktu harga telur diubah ke dalam range data yang sama, untuk meminimalisasi nilai *error* hasil prediksi. Tahapan iterasi SVR dilakukan untuk mengetahui nilai MAPE hasil prediksi data latih. Tahapan selanjutnya adalah proses PSO, dimana pada tahapan ini dilakukan secara iteratif hingga iterasi maksimal. Pada tahapan PSO didapatkan satu partikel terbaik yang tersusun atas parameter-parameter SVR dengan nilai *fitness* tertinggi dan nilai MAPE terkecil. Tahapan pengujian SVR dilakukan untuk mendapatkan nilai MAPE hasil prediksi data uji dengan menggunakan parameter SVR hasil optimasi. *Update Gbest*, memperbarui partikel global yang didapatkan dari nilai partikel terbaik dari partikel terbaik lokal (*Pbest*).

2.9 Evaluasi Kinerja

Perkembangan teknik evaluasi akurasi metode peramalan dimulai pada tahun 1939 oleh Tinbergen untuk membuktikan pernyataan dari Keynes bahwa suatu teori harus dikonfimasikan apakah data dan metode statistik bekerja dengan baik (Woschnagg & Cipan, 2004). Evaluasi akurasi prediksi berfungsi untuk mengukur seberapa baik metode prediksi dalam melakukan prediksi data.

Teknik pengukuran akurasi prediksi ada 2 yaitu, *stand-alone accuracy* dan *relative accuracy*. Pengukuran *stand-alone accuracy* adalah pengukuran akurasi yang didapat tanpa referensi prediksi tambahan dan biasanya terkait dengan *loss function*. Misalnya pengukuran akurasi yang berdasarkan fungsi kuadrat dan

fungsi absolut. Pengukuran *relative accuracy* digagas untuk mengevaluasi kinerja dari *forecast relative* dalam prediksi *benchmark*. Berikut beberapa jenis pengukuran akurasi metode peramalan (Woschnagg & Cipan, 2004):

1. ***Root Mean Squared Error*** **(RMSE)**

   RMSE bergantung pada skala dari variabel dependen. RMSE digunakan sebagai pengukuran relatif untuk membandingkan prediksi pada series yang sama dalam model berbeda. Semakin kecil nilai *error*, maka semakin baik kemampuan prediksi metode. Nilai RMSE dihitung dengan menguadratkan nilai residual dibagi dengan jumlah data, lalu diakarkan. Nilai residual adalah selisih dari data aktual dengan data hasil prediksi. RMSE dihitung dengan Persamaan (2.49).

$$\boldsymbol{RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}} \qquad \textbf{(2.49)}$$

   Keterangan:

   y_i : data aktual

   $\hat{y}_i$: data target

   n : banyaknya data

2. ***Mean Square Error*** **(MSE)**

   MSE dapat dianalogikan sebagai nilai varian ditambah dengan kuadrat nilai bias suatu model. MSE memberikan bobot yang lebih besar karena menggunakan nilai kuadratik dari *error*. Sehingga nilai *error* yang kecil akan semakin kecil dan nilai *error* yang besar akan semakin besar. MSE sangat sensitif terhadap nilai *outlier*, karena dihitung dengan sistem kuadratik. Keuntungan penggunaan MSE adalah MSE sangat baik dalam membuat gambaran untuk mengukur konsistensi model. Model dengan nilai varian kecil lebih mampu memberikan hasil yang konsisten dibandingan dengan model dengan nilai varian besar. Rumus untuk menghitung nilai MSE prediksi dijabarkan pada Persamaan (2.50).

$$\boldsymbol{MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \qquad \textbf{(2.50)}$$

   Keterangan:

   y_i : data aktual

   $\hat{y}_i$: data target

   n : banyaknya data

3. ***Mean Absolute Error*** **(MAE)**

   MAE merepresentasikan nilai rata-rata dari nilai *error* absolut. MAE bergantung pada skala dari variabel dependen, tetapi MAE tidak terlalu terpengaruh dengan dengan deviasi daripada *squared loss*. Secara intuitif, MAE menghitung rata-rata nilai *error* dengan bobot yang sama

untuk keseluruhan data yang dievaluasi. MAE dihitung dengan Persamaan (2.51).

$$\boldsymbol{MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|^2} \quad \textbf{(2.51)}$$

Keterangan:

y_i : data aktual

$\hat{y}_i$: data target

n : banyaknya data

4. *Mean Absoulte Percentage Error* (MAPE)

Pengukuran akurasi yang populer lainnya adalah MAPE. MAPE adalah metode pengukuran nilai kesalahan (*error*) relatif dengan menggunakan pengukuran nilai absolut. Penggunaan nilai absolut pada perhitungan MAPE memiliki dua keuntungan. Pertama, nilai absolut tetap menjaga hasil perhitungan menjadi nilai positif. Kedua, metode MAPE memungkinkan untuk membandingkan hasil akurasi antara data time series beda skala karena MAPE tidak bergantung pada variabel dependen. Perhitungan nilai MAPE ditunjukkan oleh Persamaan (2.52).

$$\boldsymbol{MAPE = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right| \times 100\%} \quad \textbf{(2.52)}$$

Keterangan:

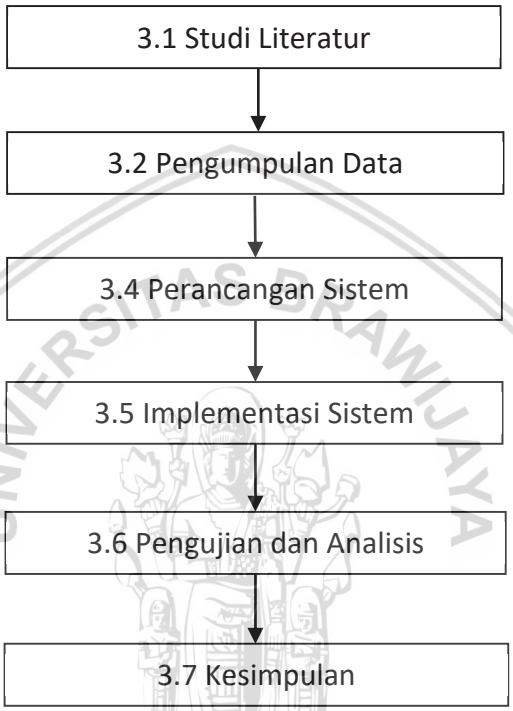
y_i : data aktual

$\hat{y}_i$: data target

n : banyaknya data

BAB 3 METODOLOGI

Bab ini menjelaskan tentang langkah-langkah yang ditempuh dalam penelitian yang diusulkan. Metodologi yang dilakukan dalam penelitian ini dilakukan melalui beberapa tahapan, yakni studi literatur, pengumpulan data, analisa dan perancangan sistem, implementasi sistem, pengujian sistem, dan pengambilan kesimpulan. Gambar 3.1 merupakan diagram alir yang berisi tahapan-tahapan yang dilakukan dalam penelitian yang diusulkan.



Gambar 3.1 Diagram alir tahapan penelitian

3.1 Studi literatur

Pada tahapan ini dilakukan pembelajaran literatur atau pustaka dari bidang-bidang ilmu yang berhubungan dengan penelitian ini, diantaranya:

1. Peramalan (*Forecasting*)
2. Support Vector Regression
3. Particle Swarm Optimization

Literatur diperoleh dari jurnal, *e-book,* buku, penelitian sebelumnya dan artikel-artikel dari internet yang dipandang layak dan berhubungan dengan tema penelitian.

3.2 Pengumpulan data

Data didapatkan dari situs web Sistem Informasi Ketersediaan dan Perkembangan Harga Bahan Pokok di Jawa Timur milik Dinas Perindustrian dan Perdagangan Jawa Timur. Data berupa data runtut waktu (*time series*) per hari

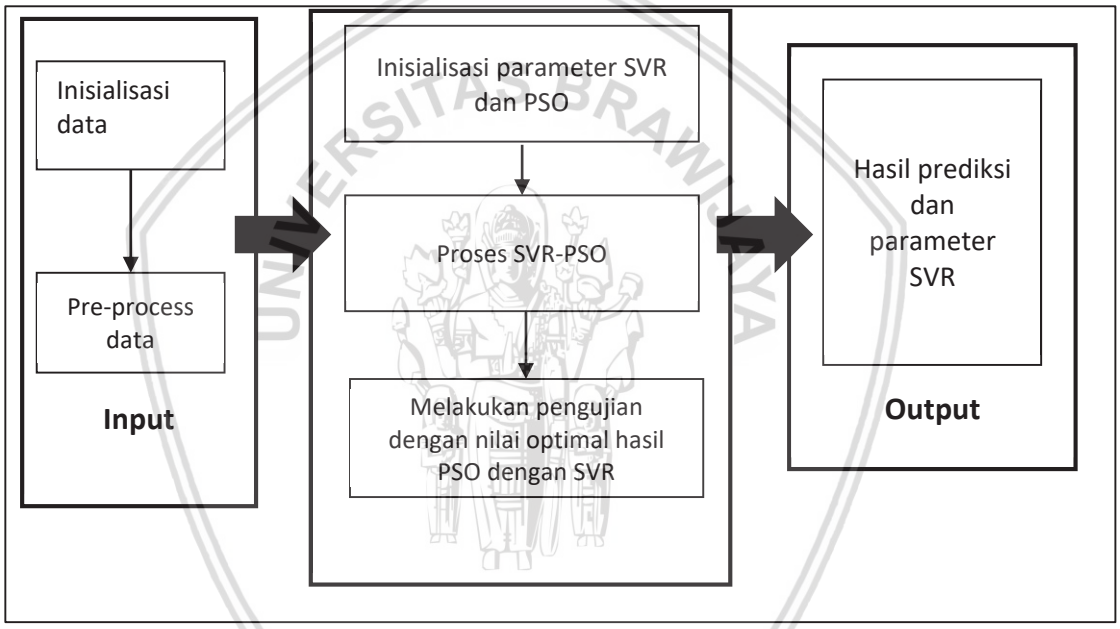
harga telur ayam ras per kilogram untuk wilayah Kota Malang dari 1 Januari 2012 sampai 31 Desember 2017.

3.3 Perancangan

Perancangan sistem merupakan tahapan yang dilakukan untuk merancang sistem baik dari segi model maupun arsitekturnya, hal apa saja yang dibutuhkan sistem dalam proses pembuatannya meliputi deskripsi sistem, perancangan algoritme program, dan perancangan pengujian. Langkah-langkah kerja sistem disesuaikan dengan arsitektur yang telah dirancang.

3.3.1 Model perancangan sistem

Pada tahap ini menjabarkan mengenai kinerja sistem secara terstruktur, dimulai dari *input* hingga *output* yang dihasilkan. Diagram model perancangan sistem dapat dilihat pada Gambar 3.2.



Gambar 3.2 Diagram model perancangan sistem

Proses yang berlangsung dalam Gambar 3.2 diatas yaitu:

1. *Input*

   Input dari sistem ini yaitu berupa data runtut waktu harga telur ayam ras.

2. Proses

   Proses penghitungan prediksi harga telur ayam ras menggunakan metode SVR-PSO adalah sebagai berikut :

   a. Inisialisasi data, parameter SVR, iterasi maksimal PSO, *range*, populasi, jumlah fitur
   b. Normalisasi data
   c. Inisialisasi partikel awal
   d. Proses SVR

e. Hitung *fitness*
f. Lakukan iterasi
    f.1 Lakukan perulangan
        i. *Update* kecepatan
        ii. *Update* posisi
        iii. SVR
        iv. Hitung *fitness*
        v. *Update* pBest
    f.2 Iterasi sama dengan jumlah populasi, perulangan berhenti
    f.3 *Update* nilai gBest
g. Mencapai iterasi maksimal, perulangan berhenti

3. Output

   Keluaran sistem akan berupa nilai-nilai parameter SVR yang optimal dan hasil dari prediksi menggunakan parameter optimal SVR.

3.4 Implementasi sistem

Penerapan sistem sesuai dengan perancangan yang telah dibuat yaitu berbasis web dan menggunakan bahasa pemrograman Java sebagai *platform*-nya. Adapun fase-fase dari implementasi sistem adalah sebagai berikut:

1. Implementasi antarmuka sistem dengan menggunakan bahasa pemrograman Java.
2. Implementasi metode *Support Vector Regression – Particle Swarm Optimization* (SVR-PSO) untuk sistem prediksi.

Keluaran sistem berupa informasi hasil prediksi harga telur ayam ras untuk waktu tertentu berdasarkan masukan yang diinput oleh pengguna.

3.5 Pengujian dan analisis

Pengujian sistem dilakukan untuk mengetahui apakah sistem berjalan dngan baik sesuai dengan perancangan yang telah dibuat sebelumnya dan sesuai dengan spesifikasi kebutuhan sistem. Berikut ini adalah tahapan pengujian yang akan dilakukan:

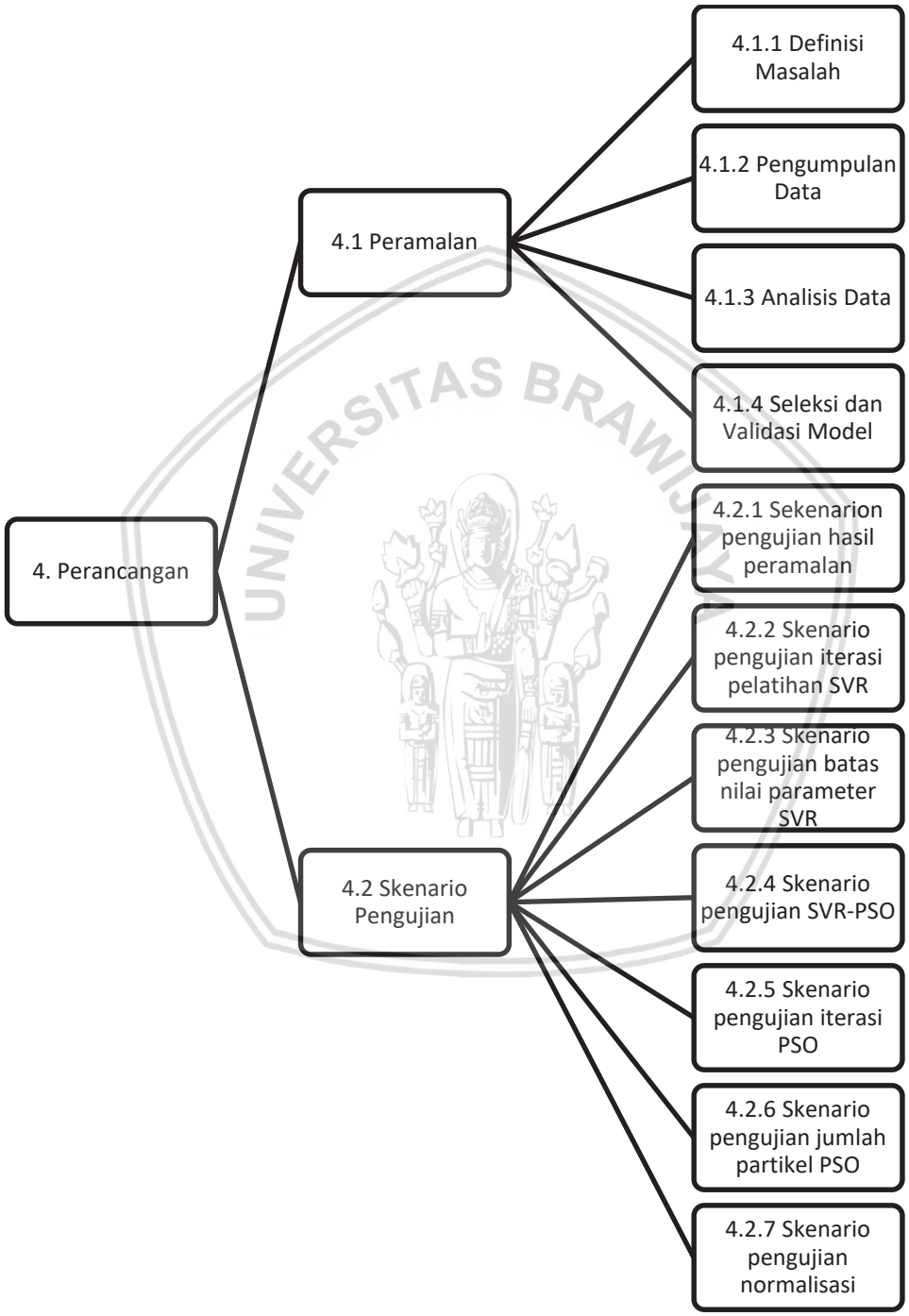
1. Pengujian perbandingan hasil peramalan
2. Pengujian iterasi pelatihan SVR
3. Pengujian batas nilai parameter SVR
4. Pengujian SVR-PSO
5. Pengujian iterasi PSO
6. Pengujian jumlah partikel PSO
7. Pengujian Normalisasi

3.6 Kesimpulan

Penarikan kesimpulan diperoleh setelah melalui semua tahapan penelitian dari studi literatur, perancangan sampai pengujian sistem. Setelah itu, dipaparkan saran yang dapat dijadikan motivasi untuk mengembangkan sistem yang lebih baik dan sempurna ke depannya.

BAB 4 PERANCANGAN

Bab ini berisi tentang perancangan sistem prediksi. Tahapan perancangan sistem prediksi ini meliputi perancangan sistem prediksi menggunakan metode SVR – PSO, perancangan antarmuka sistem, dan perancangan skenario pengujian sistem. Diagram alir tahapan perancangan sistem ditunjukkan pada Gambar 4.1.



Gambar 4.1 Diagram alir tahapan perancangan sistem

4.1 Peramalan

Perancangan sistem peramalan menjelaskan tentang tahapan yang harus dilalui untuk sistem peramalan, yaitu definisi masalah untuk menjelaskan permasalahan utama mengapa sistem peramalan ini diperlukan, pengumpulan data yang menjelaskan tentang sumber data yang digunakan, analisis data untuk menganalisis pola data yang digunakan untuk peramalan, seleksi model untuk memilih model sistem peramalan yang akan digunakan, dan validasi model untuk memvalidasi model sistem peramalan yang terseleksi.

4.1.1 Definisi masalah

Pada penelitian ini masalah yang akan dibahas adalah peramalan harga telur ayam ras di Kota Malang. Harga telur di Kota Malang memiliki pola yang fluktuatif dan tidak pasti. Faktor yang menyebabkan naik-turunnya harga telur adalah minat konsumen dan jumlah produksi telur, sedangkan jumlah produksi telur ini sangat ditentukan oleh cuaca. Pada musim kemarau biasanya ayam petelur bertelur lebih produktif dibandingkan pada musim hujan. Perubahan iklim global karena pemanasan global telah menyebabkan tidak menentunya cuaca di Indonesia. Untuk dapat menjaga kestabilan harga telur ayam sehingga dibutuhkan suatu sistem cerdas yang dapat meramalkan harga telur secara akurat dan terperinci.

4.1.2 Pengumpulan data

Pengumpulan data pada penelitian ini diperoleh dari situs web Sistem Informasi Ketersediaan dan Perkembangan Harga Bahan Pokok di Jawa Timur milik Dinas Perindustrian dan Perdagangan Jawa Timur. Data yang digunakan untuk penelitian ini adalah data runut waktu harga telur harian dan wilayah yang digunakan adalah Kota Malang dengan rentang waktu 2012 – 2017. Data harga telur rata-rata di Kota Malang sendiri adalah harga rata-rata dari harga telur di 5 pasar yang ada di Malang yaitu pasar Klojen, pasar Oro-Oro Dowo, pasar Tawangmangu, pasar Blimbing, dan pasar Madyopuro. Berikut ini data harian harga rata-rata telur ayam ras di Kota Malang tahun 2012 – 2017 yang ditunjukkan oleh Lampiran A. Berikut tampilan website siskaperbapo yang ditunjukkan pada Gambar 4.2

Harga Rata-Rata Provinsi Jawa Timur di Tingkat Konsumen Tanggal 2018-06-06 11:55:29

NO	NAMA BAHAN POKOK	SATUAN	HARGA KEMARIN	HARGA SEKARANG	PERUBAHAN (Rp)	PERUBAHAN (%)
01	BERAS					
	- Bengawan	kg	11.342	11.363	20	0,18% ▲
	- Mentik	kg	11.095	11.117	22	0,20% ▲
	- IR 64	kg	9.510	9.485	-25	-0,26% ▼
02	GULA PASIR					
	- Gula Pasir Dalam Negri	kg	11.212	11.229	17	0,16% ▲
03	MINYAK GORENG					
	- Bimoli Botol / Kemasan (Sps) 620 ml	620 ml	11.703	11.796	93	0,80% ▲
	- Bimoli botol/Kemasan (sps) 2 liter	2 Liter	26.417	26.313	-104	-0,39% ▼
	- Tanpa Merk / Minyak Curah	kg	11.198	11.172	-26	-0,24% ▼
04	DAGING					
	- Daging Sapi Murni	kg	107.530	108.119	589	0,55% ▲
	- Daging Ayam Broiler	kg	34.820	35.156	336	0,97% ▲
	- Daging Ayam Kampung	kg	59.019	59.537	519	0,88% ▲
05	TELUR AYAM					
	- Telur Ayam Ras / Petelur	kg	20.825	20.811	-14	-0,07% ▼
	- Telur Ayam Kampung	kg	36.904	36.644	-260	-0,70% ▼

Sumber : (Disperindag, 2011)

Gambar 4.2 Situs web siskaperbapo

Tabel 4.1 Data harga rata-rata telur ayam ras harian Kota Malang

Tanggal	Tahun						Rata	Min	Max
	2012	2013	2014	2015	2016	2017			
1-Jan	**14600**	15400	16100	19400	22600	**20800**	18150	14600	20800
2-Jan	**14600**	15400	16300	19400	22800	**21000**	18250	14600	21000
3-Jan	**14600**	15600	16300	19200	22800	**21000**	18250	14600	21000
4-Jan	**14600**	15800	16300	19000	22800	**21000**	18250	14600	21000
5-Jan	**14160**	15800	16300	19000	22800	**20400**	18076,67	14160	20400
6-Jan	**14160**	15900	16400	19000	**22800**	20400	18110	14160	22800
7-Jan	**14160**	16000	16400	19200	**22800**	20400	18160	14160	22800
8-Jan	**14360**	16300	16400	19600	**22800**	19700	18193,33	14360	22800
9-Jan	**14360**	16300	16600	19600	**22800**	18800	18076,67	14360	22800
10-Jan	**14160**	16600	16700	19700	**22800**	18800	18126,67	14160	22800
11-Jan	**14600**	16600	16800	20100	**22800**	18500	18233,33	14600	22800
12-Jan	**14900**	16600	16800	21200	**22800**	18800	18516,67	14900	22800
13-Jan	**14900**	16700	17000	21200	**22800**	18800	18566,67	14900	22800
14-Jan	**14900**	17000	17000	21200	**22400**	18800	18550	14900	22400
15-Jan	**15400**	17200	17000	21200	**22400**	18800	18666,67	15400	22400
16-Jan	**15400**	17200	17000	21200	**22400**	18800	18666,67	15400	22400
...	...	...	...	...	...	...	...	...	...
31-Dec	**15400**	16100	19400	22600	20700	23600	19633,33	15400	23600

(Sumber : Dinas Perdagangan Kota Malang)

4.1.3 Analisis data

Analisis data pada penelitian ini bertujuan untuk mengetahui pola dan karakteristik data *time series* yang digunakan serta sebagai tahapan *pre-processing* data sebelum masuk ke proses peramalan. Berikut spesifikasi data yang digunakan pada penelitian ini.

1. Data yang digunakan adalah data harga rata-rata telur ayam ras di Kota Malang pada tahun 2012 – 2017.
2. Hasil rekapitulasi data harga rata-rata telur ayam ditunjukkan pada Lampiran B.
3. Data harga rata-rata telur ayam dengan satuan Rupiah per kilogram.

4.1.3.1 Analisis data runut waktu

Hasil analisis data ini nantinya akan digunakan untuk acuan dalam pertimbangan awal sebelum masuk ke proses peramalan dengan melihat pola naik-turunnya harga telur. Berikut ini merupakan perhitungan harga rata-rata telur ayam pada tanggal 1 bulan Januari yang terdapat pada tabel 4.1.

$$rata - rata = \frac{14.600 + 15.400 + 16.100 + 19.400 + 22.600 + 20.800}{6}$$
$$= 18.150$$

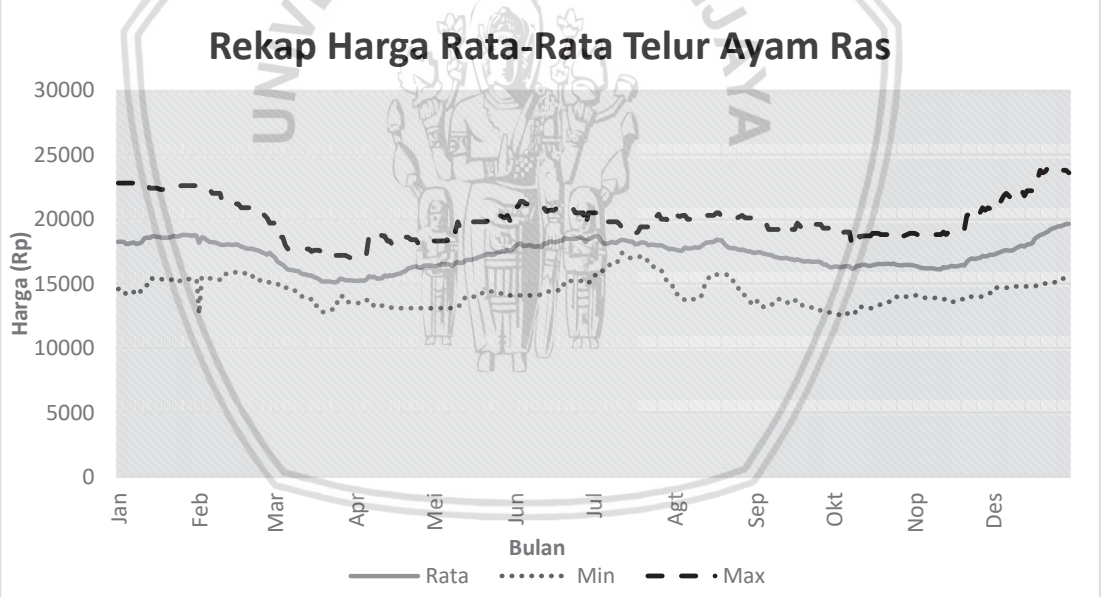
Cuplikan hasil rekapitulasi data dasarian hasil perhitungan rata-rata harga telur ayam ras di Kota Malang dapat dilihat pada kolom rata-rata Tabel 4.2.

Tabel 4.2 Cuplikan rekapitulasi data runut waktu harga rata-rata telur ayam ras di Kota Malang

TANGGAL	RATA	MIN	MAX
1-Jan	18150	14600	22600
2-Jan	18250	14600	22800
3-Jan	18250	14600	22800
4-Jan	18250	14600	22800
5-Jan	18076,67	14160	22800
6-Jan	18110	14160	22800
7-Jan	18160	14160	22800
8-Jan	18193,33	14360	22800
9-Jan	18076,67	14360	22800
10-Jan	18126,67	14160	22800
11-Jan	18233,33	14600	22800
12-Jan	18516,67	14900	22800
13-Jan	18566,67	14900	22800
14-Jan	18550	14900	22400
15-Jan	18666,67	15400	22400
16-Jan	18666,67	15400	22400
17-Jan	18633,33	15400	22400
18-Jan	18616,67	15400	22300
19-Jan	18550	15300	22300

20-Jan	18566,67	15300	22300
21-Jan	18558,33	15250	22300
22-Jan	18600	15300	22300
23-Jan	18700	15300	22300
24-Jan	18683,33	15300	22300
25-Jan	18650	15100	22400
26-Jan	18750	15300	22600
27-Jan	18800	15300	22600
28-Jan	18783,33	15300	22600
29-Jan	18776,67	15360	22600
30-Jan	18693,33	15360	22600
31-Jan	18760	15360	22600
...	...	...	...
31-Dec	19633,33	15400	23600

Dari perhitungan rata-rata yang ditunjukkan pada Tabel 4.3 data harga rata rata telur ayam ras di Kota Malang per dasarian, lalu kita mendapatkan hasil penggambaran dengan grafik pola naik turunnya harga telur ayam yang ditunjukkan pada Gambar 4.3 berikut.



Gambar 4.3 Grafik rekap harga rata-rata telur ayam ras

Berdasarkan Gambar 4.3 grafik dasarian rata-rata harga telur ayam ras di Kota Malang tahun 2012 – 2017 memiliki pola musiman dengan kenaikan harga diatas rata-rata yang terjadi pada bulan januari dan desember, serta pertengahan tahun yaitu bulan mei – agustus. Garis linier yang berada pada harga 17.230 adalah nilai rata-rata dari rata-rata harga dasarian telur ayam ras tahun 2012 – 2017. Garis inilah yang memisahkan trend harga naik dan trend harga turun. Dari Gambar 4.3 Trend harga naik terjadi pada awal tahun yaitu awal bulan Januari sampai pertengahan bulan Februari dan pada akhir tahun yaitu awal bulan Desember sampai akhir bulan Desember, serta pertengahan tahun mulai akhir bulan Mei

sampai akhir bulan Agustus. Sedangkan trend harga turun terjadi pada dua periode yaitu periode pertengahan bulan Februari sampai pertengahan bulan Mei dan periode awal bulan September sampai akhir bulan November.

4.1.4 Seleksi dan validasi model

Seleksi model dilakukan untuk menentukan model atau metode prediksi yang cocok untuk data yang telah dianalisis sebelumnya. Pada penelitian ini metode prediksi yang digunakan adalah SVR – PSO dengan jenis data yang digunakan adalah data runut waktu (*time series*). Setelah itu, dilakukan validasi model untuk mengevaluasi model atau metode prediksi yang digunakan pada penelitian.

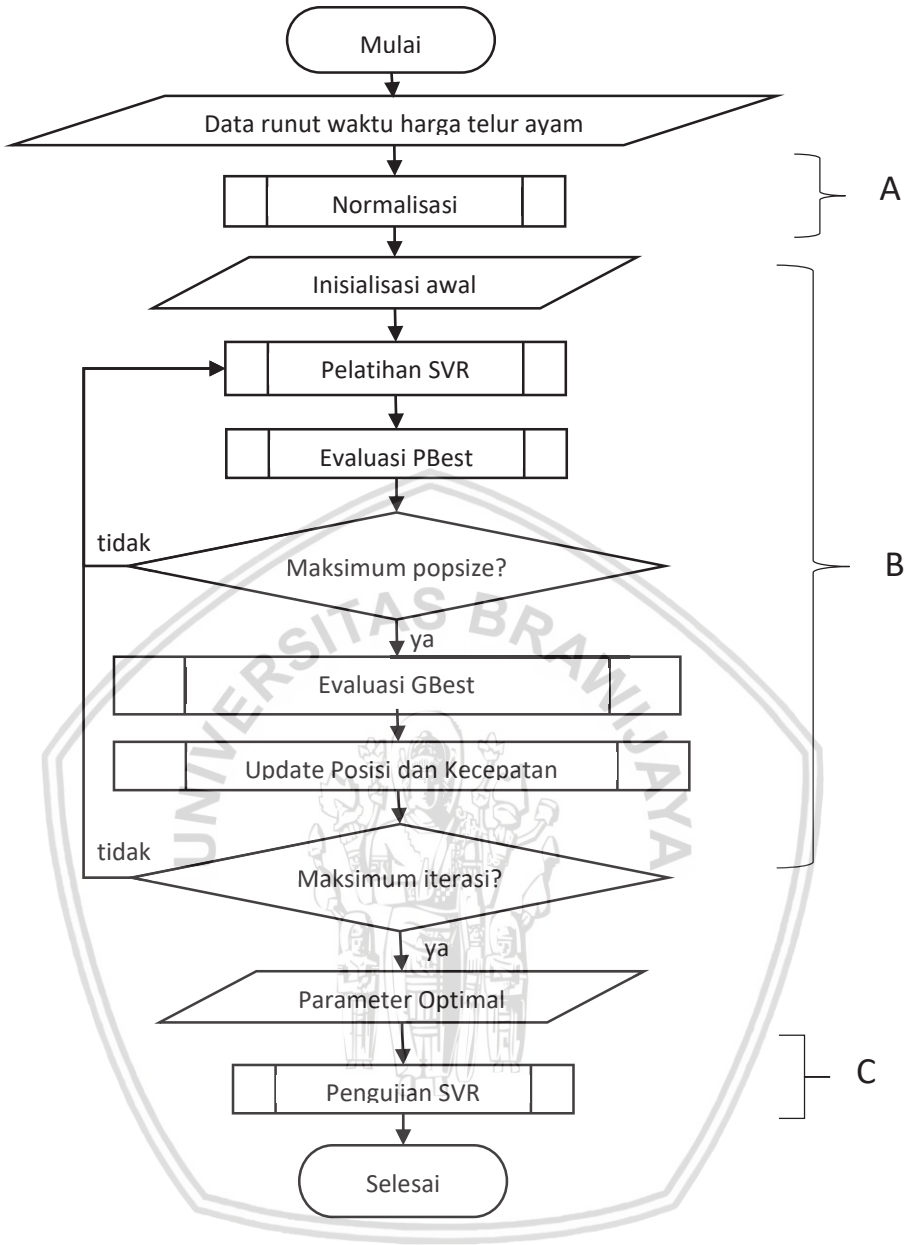
4.1.4.1 Seleksi model

Perancangan untuk peramalan data runut waktu dapat dilakukan melalui dua macam pendekatan yaitu pendekatan *artificial intelligence* (AI) dan pendekatan statistik. Pada pendekatan statistik dengan data time series pada umumnya merupakan model linier. Sedangkan pendekatan AI biasanya digunakan untuk kasus non-linier.

Pada penelitian ini kasus yang akan diselesaikan adalah peramalan kasus non-linier dengan menggunakan pendekatan AI. Salah satu metode peramalan dalam AI yang akan digunakan adalah *support vector regression* (SVR). Pada penelitian ini metode SVR akan digunakan untuk prediksi dengan menggunakan metode *particle swarm optimization* (PSO) untuk optimasi parameter-parameter SVR. Pengunaan metode SVR pada penelitian ini didasarkan pada penelitian sebelumnya yang membuktikan bahwa SVR mampu menyelesaikan kasus non-linier data runut waktu dengan baik. Optimasi parameter SVR diharapkan mampu menemukan parameter SVR optimal dengan akurasi terbaik. Pada penelitian mengenai prediksi volume penjualan mobil dengan SVR-PSO menyatakan bahwa PSO-SVR lebih baik dibandingkan dengan metode GA-SVR dalam tingkat efisiensi dan akurasi prediksi (Lu & Geng, 2011).

4.1.4.2 Validasi model

Validasi model bertujuan untuk mengevaluasi kelayakan model yang telah ditentukan pada tahapan seleksi model untuk diimplementasikan. Pada penelitian ini menyelesaikan kasus peramalan data runut waktu harga telur ayam ras di Kota Malang dengan metode *Support Vector Regression* (SVR) yang parameter-parameternya dioptimasi dengan menggunakan metode *Particle Swarm Optimization* (PSO). Berikut ini diagram alir proses peramalan runut waktu dengan metode SVR-PSO yang ditunjukkan pada Gambar 4.4



Gambar 4.4 Diagram alir metode SVR – PSO

Metode SVR-PSO memiliki inputan berupa data runut waktu harga telur ayam ras dan hasil akhir sistem berupa prediksi harga telur ayam ras. Dari Gambar 4.4 dapat dilihat bahwa metode SVR – PSO memiliki 4 tahapan proses utama, yaitu:

1. Tahapan normalisasi, data runut waktu harga telur diubah ke dalam *range* data yang sama. Normalisasi ini dilakukan dengan tujuan untuk meminimalisasi nilai *error* hasil peramalan.
2. Tahapan PSO, proses ini dilakukan secara iteratif hingga iterasi maksimal. Pada tahapan PSO didapatkan satu partikel terbaik yang tersusun atas parameter-parameter SVR dengan nilai *fitness* tertinggi

dan nilai MAPE terkecil. Perhitungan nilai *fitness* ini didapatkan dari nilai MAPE hasil pelatihan SVR data latih.

3. Tahapan pengujian SVR dilakukan untuk mendapatkan nilai MAPE hasil peramalan data uji dengan menggunakan parameter SVR hasil optimasi. Update Gbest, memperbarui partikel global yang didapatkan dari nilai partikel terbaik dari partikel terbaik lokal (Pbest).

Setiap proses memiliki sub proses yang akan dijelaskan lebih detail pada Tabel 4.3 berikut.

Tabel 4.3 Tahapan proses SVR – PSO

Kebutuhan	Proses	Metode/Teknik	Luaran
A. Normalisasi			
A.1 Menetukan Nilai Data Maksimal			
Data runut waktu harga telur ayam	Menentukan nilai terbesar dari dataset	Sorting	Nilai max
A.2 Menentukan Nilai Data Minimal			
Data runut waktu harga telur ayam	Menentukan nilai terkecil dari dataset	Sorting	Nilai min
A.3 Menentukan Range Data			
Data runut waktu harga telur ayam	Menentukan nilai range data dengan nilai min dan nilai max	Range = nilai max – nilai min	Nilai range
A.4 Menghitung Nilai Normalisasi			
Data runut waktu harga telur ayam	- Menentukan nilai max dan min - Menentukan nilai range - Menentukan nilai normalisasi dengan menggunakan nilai max, min, dan range	Normalisasi min-max	Nilai Normalisasi data uji dan data latih
B. Particle Swarm Optimization			
B.1 Pelatihan SVR			
B.1.1 Menghitung Matriks Hessian Data Latih			
B.1.1.1 Menghitung Jarak Euclidean Data Latih			
Data latih dan data uji normalisasi	Menentukan nilai jarak antar data latih mengunakan rumus Euclidean	Jarak euclidean	Matriks jarak antar data latih
B.1.1.2 Menghitung nilai kernel RBF			
- Matriks jarak data latih - Parameter σ	Menghitung kernel RBF	Kernel RBF	Nilai kernel RBF
B.1.1.3 Menghitung Nilai Matriks Hessian Data Latih			
- Matriks data jarak data latih - Nilai kernel RBF	Menghitung nilai matriks hessian data latih	Matriks hessian	Nilai matriks hessian

- Parameter λ			
B.1.2 Menghitung Nilai ϒ			
- Matriks hessian - Parameter cLR	Menghitung nilai ϒ	ϒ = cLR/ max (hessian)	Nilai ϒ
B.1.3 Sequential Learning			
B.1.3.1 Menghitung Nilai Error			
- Data latih normalisasi - Nilai αi* dan αi - Matriks hessian	Menghitung nilai error sebanyak data latih	Sequential learning	Nilai error
B.1.3.2 Menghitung Nilai δα*			
- Nilai error - Nilai α* - Parameter C dan ε - Nilai ϒ	Menghitung nilai δα*	Sequential learning	Nilai δα*
B.1.3.3 Menghitung Nilai δα			
- Nilai error - Nilai α - Parameter C dan ε - Nilai ϒ	Menghitung nilai δα	Sequential learning	Nilai δα
B.1.3.4 Menghitung Nilai α*			
- Nilai α* - Nilai δα*	Menghitung Nilai α*	Sequential learning	Nilai α* baru
B.1.3.5 Menghitung Nilai α			
- Nilai α - Nilai δα	Menghitung Nilai α	Sequential learning	Nilai α baru
Tahapan B.1.3 dilakukan sebanyak data latih dan diulangi hingga iterasi mencapai batas iterasi SVR			
B.1.4 Menghitung Nilai f(x) Data Latih			
- Nilai Lagrange Multiplier - Matriks hessian	Menghitung nilai f(x) data latih	Regresi	Nilai f(x) data latih
B.1.5 Denormalisasi Data Latih			
- Data latih normalisasi - Nilai f(x) data latih - Nilai min - Nilai range	Mendenormalisasi hasil prediksi f(x)	Denormalisasi	Nilai denormalisasi f(x)
B.1.6 Menghitung Nilai MAPE data latih			
- Data latih - Nilai denormalisasi f(x)	Menghitung Nilai MAPE hasil prediksi	MAPE	Persentase nilai MAPE prediksi
B.1.7 Menghitung Nilai *Fitness*			
- Nilai MAPE	Menghitung Nilai *Fitness*	*Fitness* = MAPE	Nilai *Fitness*
B.2 Evaluasi Nilai PBest			

- Nilai MAPE	Menghitung nilai PBest terkini	*Fitness*	Nilai pBest terkini tiap partikel
Proses B.1 dan B.2 diulangi sebanyak popsize			
B.3 Evaluasi Nilai GBest			
Nilai PBest tiap partikel	Menghitung nilai GBest iterasi terkini	GBest = max (Pbest)	Nilai GBest iterasi terkini
B.4 Update Kecepatan dan Posisi Partikel			
B.4.1 Update Kecepatan Partikel			
- Parameter w, C1, dan C2 - Nilai random - Kecepatan iterasi sebelumnya - Posisi partikel iterasi sebelumnya - Nilai GBest	Menghitung kecepatan baru partikel	Kecepatan partikel PSO	Nilai kecepatan baru
B.4.2 Update Posisi Partikel			
- Posisi partikel iterasi sebelumnya - Kecepatan partikel baru	Menghitung posisi baru partikel	Posisi(i) = Posisi(i-1) + Kecepatan(i)	Nilai posisi baru partikel
Proses B dilakukan sebanyak maksimal iterasi PSO dan simpan partikel GBest terkhir sebagai Parameter Optimal SVR			
C. Pengujian SVR			
C.1 Menghitung Matriks Hessian Data Uji			
C.1.1 Menghitung Jarak Euclidean Data Latih dan Data Uji			
Data latih dan data uji normalisasi	Menentukan nilai jarak antar data latih/data uji mengunakan rumus Euclidean	Jarak euclidean	Matriks jarak antar data latih/data uji
C.1.2 Menghitung nilai kernel RBF Data Uji			
- Matriks jarak data uji - Parameter σ optimal	Menghitung kernel RBF	Kernel RBF	Nilai kernel RBF
C.1.3 Menghitung Nilai Matriks Hessian Data Uji			
- Matriks data jarak data uji - Nilai kernel RBF - Parameter λ optimal	Menghitung nilai matriks hessian data uji	Matriks hessian	Nilai matriks hessian data uji
C.2 Menghitung Nilai f(x) Data Uji			
- Nilai Lagrange Multiplier	Menghitung nilai f(x) data uji	Regresi	Nilai f(x) data uji

- Matriks Hessian data uji - Data uji normalisasi			
C.3 Denormalisasi Data Uji			
- Data uji normalisasi - Nilai f(x) data uji - Nilai min - Nilai range	Mendenormalisasi hasil prediksi f(x)	Denormalisasi	Nilai denormalisasi f(x)
C.4 Menghitung Nilai MAPE Data Uji			
- Data Uji - Nilai denormalisasi f(x) data uji	Menghitung Nilai MAPE hasil prediksi data uji	MAPE	Persentase nilai MAPE prediksi data uji

A. Normalisasi

Pada studi kasus ini, data yang akan digunakan adalah data harga rata-rata telur ayam di Kota Malang pada bulan Januari Tabel 4.4.

Tabel 4.4 Data harga telur ayam ras bulan januari

No	Tanggal	Harga Rata-Rata
1	1-Jan	18150
2	2-Jan	18250
3	3-Jan	18250
4	4-Jan	18250
5	5-Jan	18076,67
6	6-Jan	18110
7	7-Jan	18160
8	8-Jan	18193,33
9	9-Jan	18076,67
10	10-Jan	18126,67
11	11-Jan	18233,33
12	12-Jan	18516,67
13	13-Jan	18566,67
14	14-Jan	18550

Langkah 1 : Memilih jumlah variable lag yang digunakan

Setelah memilih data yang akan digunakan untuk perhitungan yaitu data pada Tabel, selanjutnya adalah memilih lag. Pada perhitungan ini, lag yang akan digunakan adalah 4 lag time yaitu X_1, X_2, X_3, X_4 dan Y yang merupakan data sebenarnya/aktual.

Langkah 2 : Menentukan data latih dan data uji

Dari Tabel diatas terdapat 14 data runut waktu dan 4 lag time, sehingga data yang akan masuk ke perhitungan sebanyak 10 data yang akan dibagi menjadi data latih dan data uji. Pada penelitian ini, rasio data latih dan data uji adalah 7 : 3

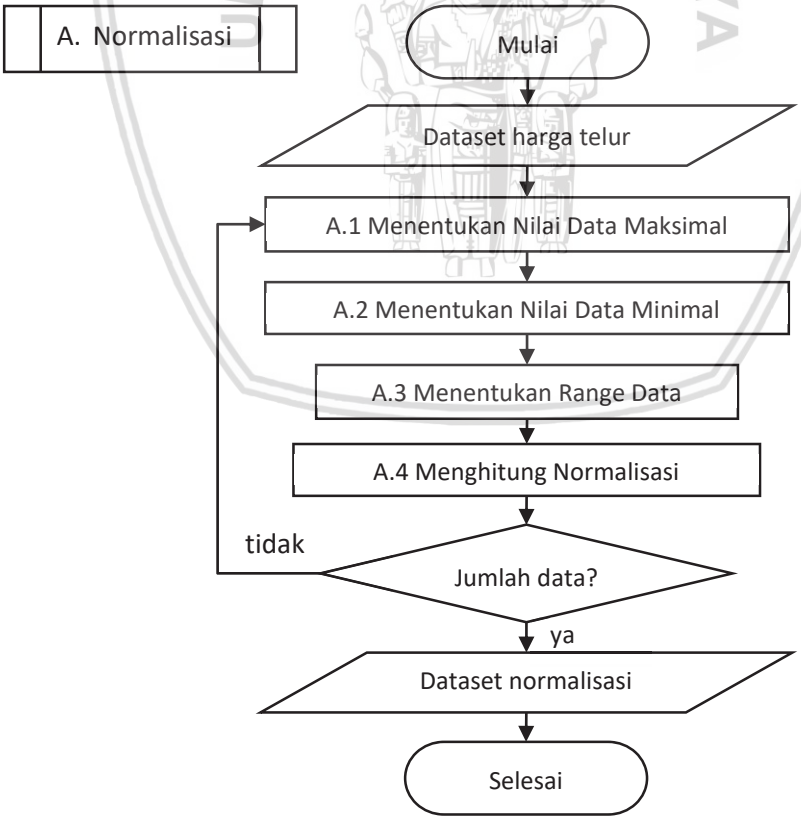
dari semua data, sehingga data latih berjumlah sebanyak 7 data dan data uji sebanyak 3 data. Pembagian data latih dan data uji dapat dilihat pada Tabel 4.5

Tabel 4.5 Pembagian data latih dan data uji

Data Ke -	X_1	X_2	X_3	X_4	Y	Jenis Data
1	18150	18250	18250	18250	18076,67	Data Latih
2	18250	18250	18250	18076,67	18110	
3	18250	18250	18076,67	18110	18160	
4	18250	18076,67	18110	18160	18193,33	
5	18076,67	18110	18160	18193,33	18076,67	
6	18110	18160	18193,33	18076,67	18126,67	
7	18160	18193,33	18076,67	18126,67	18233,33	
8	18193,33	18076,67	18126,67	18233,33	18516,67	Data Uji
9	18076,67	18126,67	18233,33	18516,67	18566,67	
10	18126,67	18233,33	18516,67	18566,67	18550	

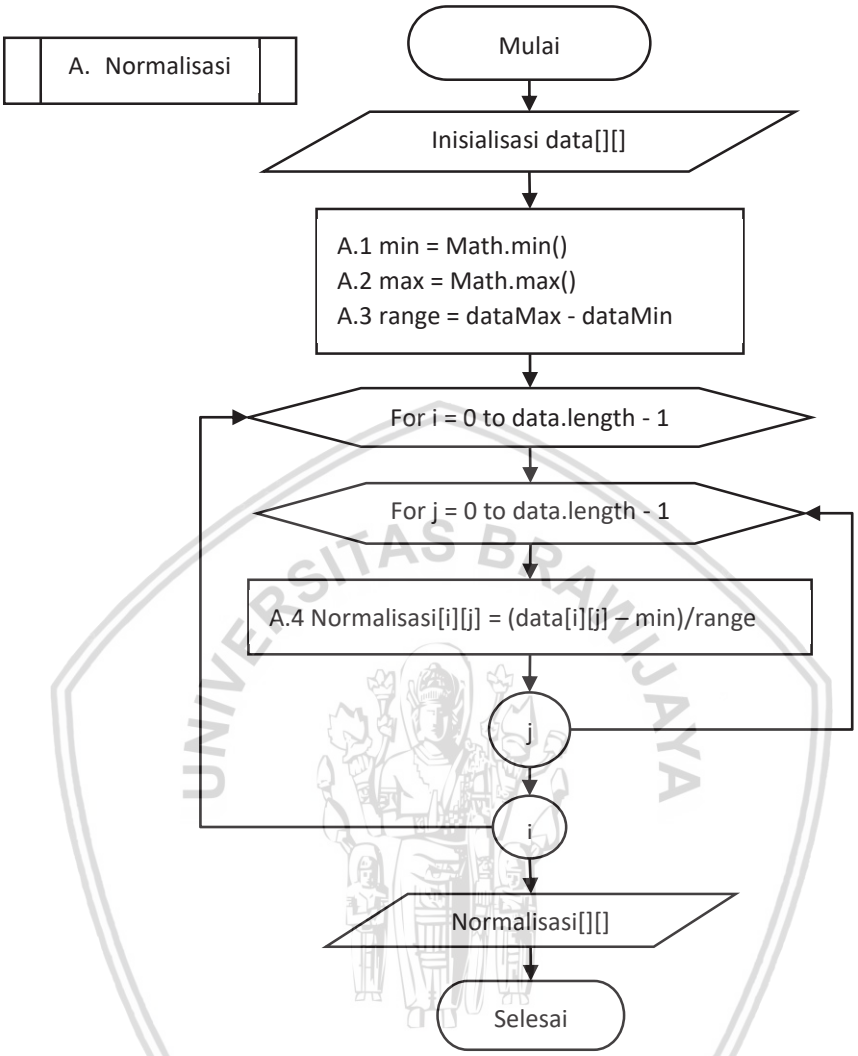
Langkah 3 : Proses Normalisasi

Normalisasi adalah salah satu tahap penting untuk meminimalkan variasi dari data dan nilai *error* prediksi. Sebelum melakukan normalisasi, diperlukan nilai range yang didapat dari selisih nilai maksimal dan nilai minimal. Tahapan proses normalisasi dapat dilihat pada Gambar 4.5



Gambar 4.5 Diagram alir proses normalisasi

Tahapan proses algoritme normalisasi data dapat dilihat pada Gambar 4.6 berikut ini.



Gambar 4.6 Diagram alir algoritme proses normalisasi

Penjelasan dari diagram alir algoritme proses normalisasi data akan dijabarkan pada pseudocode Gambar 4.7 berikut ini.

```
Nama algoritme : normalisasi
Deklarasi :
Double data[][], max, min, range, normalisasi[][]
Deskripsi :
 4. Input : data[][]
 5. Proses :
     A.1 Menentukan nilai variabel min
     A.2 Menentukan nilai variabel max
     A.3 Menghitung range = max – min
     A.4 Menghitung normalisasi[i][j] = (data[i][j] – min)/range
 6. Output : normalisasi[][]
```

Gambar 4.7 Pseudocode algoritme normalisasi

Berdasarkan pseudocode Gambar 4.7 berikut penjelasan detail tahapan normalisasi data :

A.1 Menentukan nilai data maksimal

Nilai maksimal digunakan saat perhitungan nilai *range.* Data yang digunakan untuk pencarian nilai maksimal adalah nilai X_1, X_2, X_3, X_4, dan Y dari data uji dan data latih pada Tabel 4.6. Nilai maksimal data Tabel 4.5 yang digunakan untuk perhitungan normalisasi adalah 18066,67.

A.2 Menentukan nilai data minimal

Nilai minimal digunakan saat perhitungan nilai range, dimana nilai range dan nilai minimal ini akan digunakan saat perhitungan normalisasi. Nilai minimal data pada Tabel 4.5 yang digunakan untuk perhitungan normalisasi adalah 18076,67.

A.3 Menghitung range

Nilai range diperoleh dari selisih nilai maksimal dan nilai minimal. Berikut penjelasan contoh perhitungan nilai range untuk data Tabel 4.6

$$range = max - min$$

$$range = 18076{,}67 - 18566{,}67 = 490$$

Tabel 4.6 Range data normalisasi

Nilai Minimal	18076,67
Nilai Maksimal	18566,67
Nilai Range	490

A.4 Menghitung normalisasi

Selanjutnya adalah tahapan menghitung nilai normalisasi menggunakan rumus normalisasi min-max pada persamaan 2.1. Pada contoh kasus perhitungan disini batasan normalisasi nya adalah [0,1]. Berikut penjelasan contoh perhitungan normalisasi data X_1 data ke -1

$$x_{1,1} = \frac{18150 - 18076{,}67}{490} \times (1 - 0) + 0 = 0{,}14966$$

Hasil perhitungan normalisasi untuk data pada Tabel 4.5 dijabarkan pada Tabel 4.7 berikut.

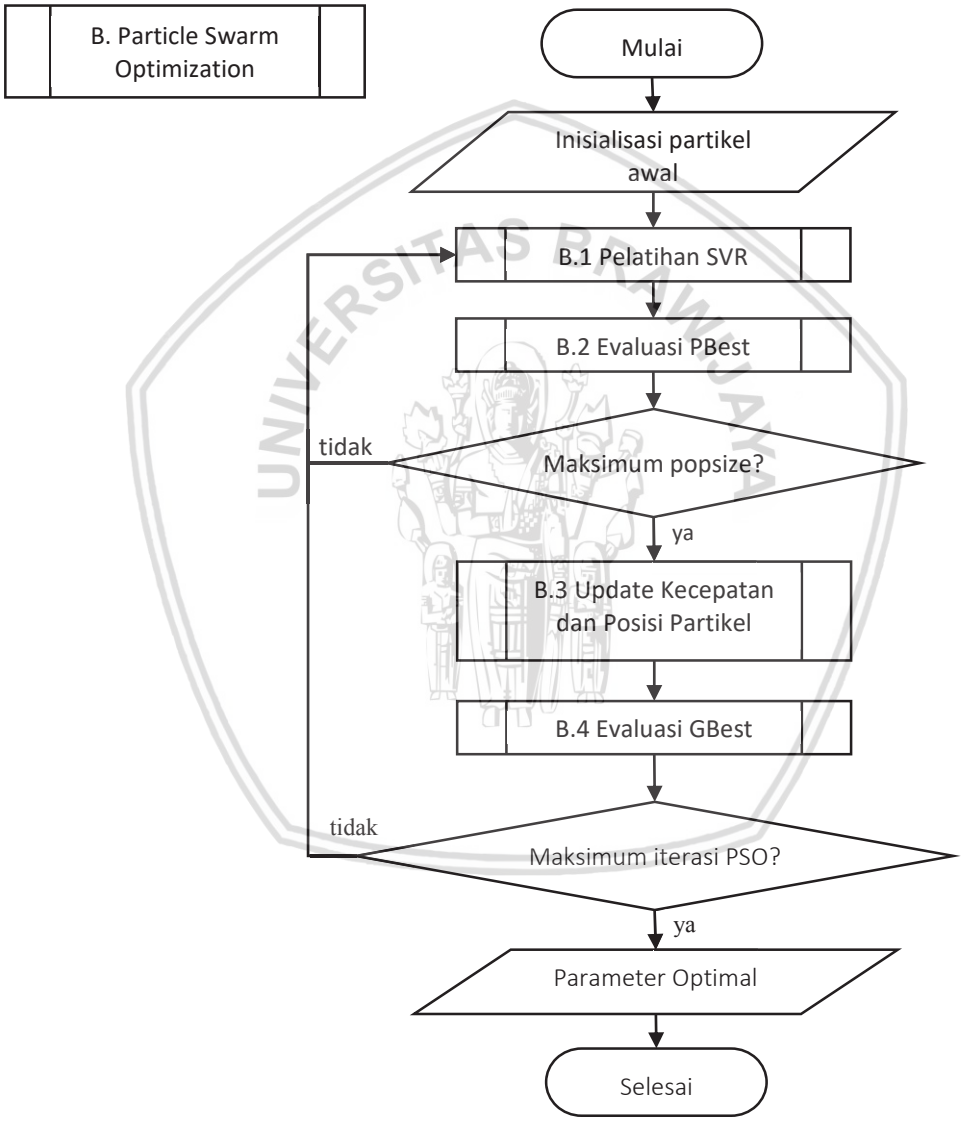
Tabel 4.7 Hasil normalisasi data

Data Ke -	X_1	X_2	X_3	X_4	Y	Jenis Data
1	**0,14966**	0,353741	0,353741	0,353741	0	Data Latih
2	0,353741	0,353741	0,353741	0	0,068027	
3	0,353741	0,353741	0	0,068027	0,170068	
4	0,353741	0	0,068027	0,170068	0,238095	
5	0	0,068027	0,170068	0,238095	0	
6	0,068027	0,170068	0,238095	0	0,102041	
7	0,170068	0,238095	0	0,102041	0,319728	
8	0,238095	0	0,102041	0,319728	0,897959	Data Uji

9	0	0,102041	0,319728	0,897959	1	
10	0,102041	0,319728	0,897959	1	0,965986	

B. Particle Swarm Optimization

Pada studi kasus ini PSO adalah metode yang digunakan untuk mengoptimasi parameter SVR. Parameter SVR yang akan dioptimasi adalah λ, C, ε, dan cLR. Dalam kasus ini, *fitness* yang digunakan untuk mengukur kualitas partikel sebanding dengan nilai MAPE hasil peramalan dengan menggunakan metode SVR. Diagram alir metode PSO dapat dilihat pada Gambar 4.8 berikut.



Gambar 4.8 Diagram alir metode PSO

PSO Iterasi ke – 0

Langkah 1 : Inisialisasi Posisi Partikel Awal (X_i)

Pada inisialisasi partikel awal (t = 0) dibangkitkan secara acak sebanyak popSize yang telah ditentukan sebelumnya. Penentuan batas minimal dan batas maksimal parameter SVR dilakukan berdasarkan asusmsi batas parameter pada penelitian metode SVR yang telah dilakukan sebelumnya yang ditunjukkan pada Tabel 4.8 dan Tabel 4.9 berikut ini.

Tabel 4.8 Batas parameter SVR penelitian sebelumnya

Parameter	Penelitian	Batas Minimal	Batas Maksimal
Lambda (λ)	(Vijayakumar & Wu, 1999)	0	5
Complexity (C)	(Wang, et al., 2003)	0,001	1000
Epsilon (ε)	(Wang, et al., 2003)	0,001	100

Tabel 4.9 Inisialisasi batas parameter SVR

Parameter	Batas Minimal	Batas Maksimal
λ	0,01	1
C	10	1000
ε	0,0001	0,01
cLR	0,01	1

Setelah menentukan batasan parameter, selanjutnya adalah membangkitkan partikel secara random berdasarkan Persamaan (2.18) yang dapat dilihat pada Tabel 4.10 berikut ini.

Misalkan :

Rand[0,1] = 0,68828

$X_{1,\lambda}$ = 0,01 + (0,688283) x (1 – 0,01) = 0,691401

Tabel 4.10 Hasil inisialisasi posisi partikel awal

Pop Size = 3				
Partikel (X_i)	**λ**	**C**	**ε**	**cLR**
X_1	**0,691401**	570,5107	0,005988	0,5925
X_2	0,540425	136,5756	0,005018	0,2539
X_3	0,772748	74,43793	0,007109	0,390051

Langkah 2 : Inisialisasi Kecepatan Partikel Awal

Pada iterasi ke – 0 kecepatan partikel semua partikel adalah 0. Berikut hasil inisialisasi kecepatan partikel dapat dilihat pada Tabel 4.11 berikut ini.

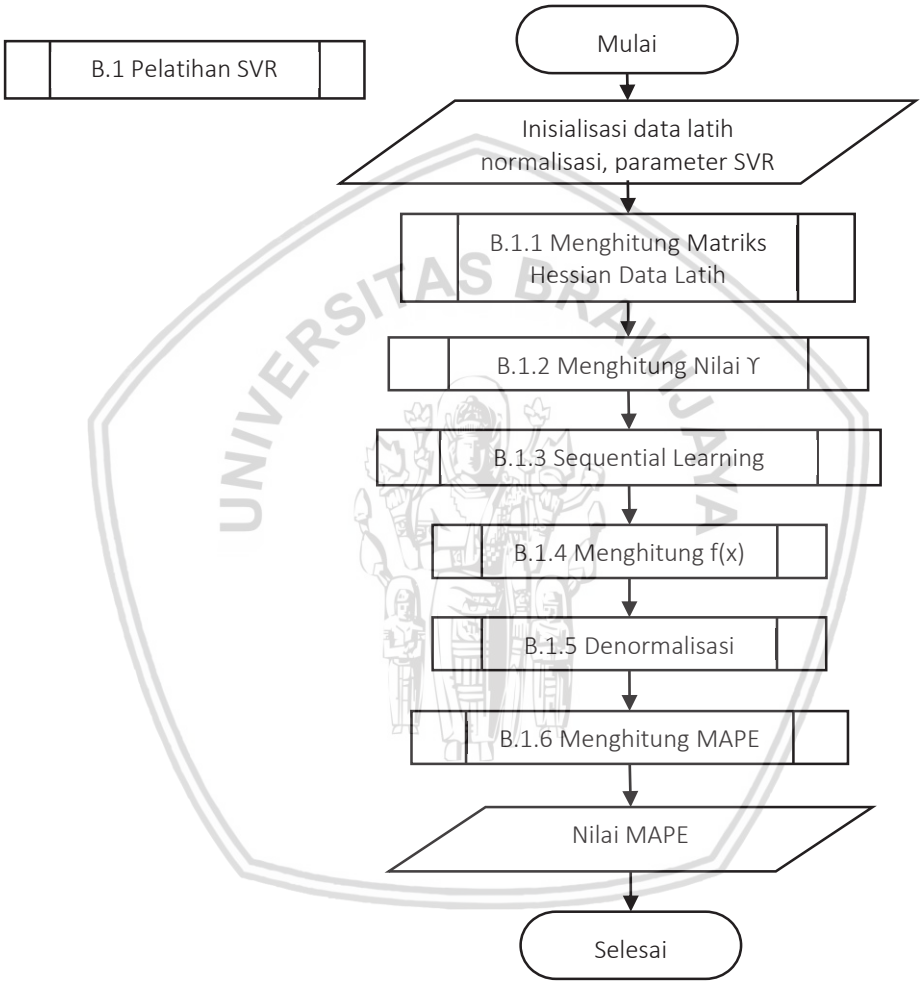
Tabel 4.11 Inisialisasi kecepatan partikel awal

Kecepatan (V_i)	λ	C	ε	cLR
V_1	0	0	0	0

V_2	0	0	0	0
V_3	0	0	0	0

B.1 Pelatihan SVR

Untuk mengukur kualitas partikel PSO, dilakukan perhitungan nilai *fitness* untuk masing-masing partikel PSO. Perhitungan nilai *fitness* didapatkan berdasarkan nilai MAPE yang didapatkan dari pelatihan SVR untuk setiap partikel. Berikut diagram alir pelatihan SVR yang ditunjukkan oleh Gambar 4.9.



Gambar 4.9 Diagram alir pelatihan SVR

Inisialisasi data latih

Data latih adalah data yang telah dinormalisasi sebelumnya. Data latih yang digunakan sebanyak 7 data. Berikut inisialisasi data latih dapat dilihat pada Tabel 4.12 berikut ini.

Tabel 4.12 Inisialisasi data latih

Data Ke -	X_1	X_2	X_3	X_4	Y
1	0,14966	0,353741	0,353741	0,353741	0

2	0,353741	0,353741	0,353741	0	0,068027
3	0,353741	0,353741	0	0,068027	0,170068
4	0,353741	0	0,068027	0,170068	0,238095
5	0	0,068027	0,170068	0,238095	0
6	0,068027	0,170068	0,238095	0	0,102041
7	0,170068	0,238095	0	0,102041	0,319728

Inisialisasi parameter SVR

Parameter SVR yang akan dilatih adalah partikel X_1 hasil inisialisasi awal PSO. Berikut ini tabel inisialisasi parameter SVR yang ditunjukkan pada Tabel 4.13.

Tabel 4.13 Inisialisasi parameter SVR

λ	C	ε	cLR	Σ
0,691401	570,5107	0,005988	0,5925	0,1

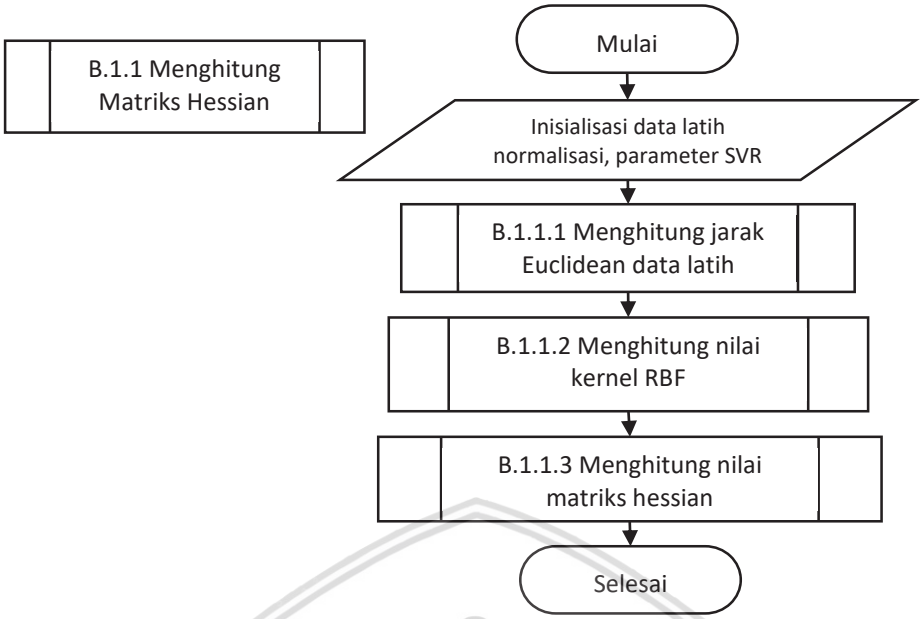
Selain parameter SVR, nilai α dan α* juga perlu diinisialisasi dengan nilai 0 sebanyak data latih yang digunakan dalam pelatihan. Inisialisasi nilai α dan α* ditujukkan pada Tabel 4.14 berikut.

Tabel 4.14 Inisialisasi nilai α dan α*

Data ke – i	α*	α
1	0	0
2	0	0
3	0	0
4	0	0
5	0	0
6	0	0
7	0	0

B.1.1 Menghitung Matriks Hessian Data Latih

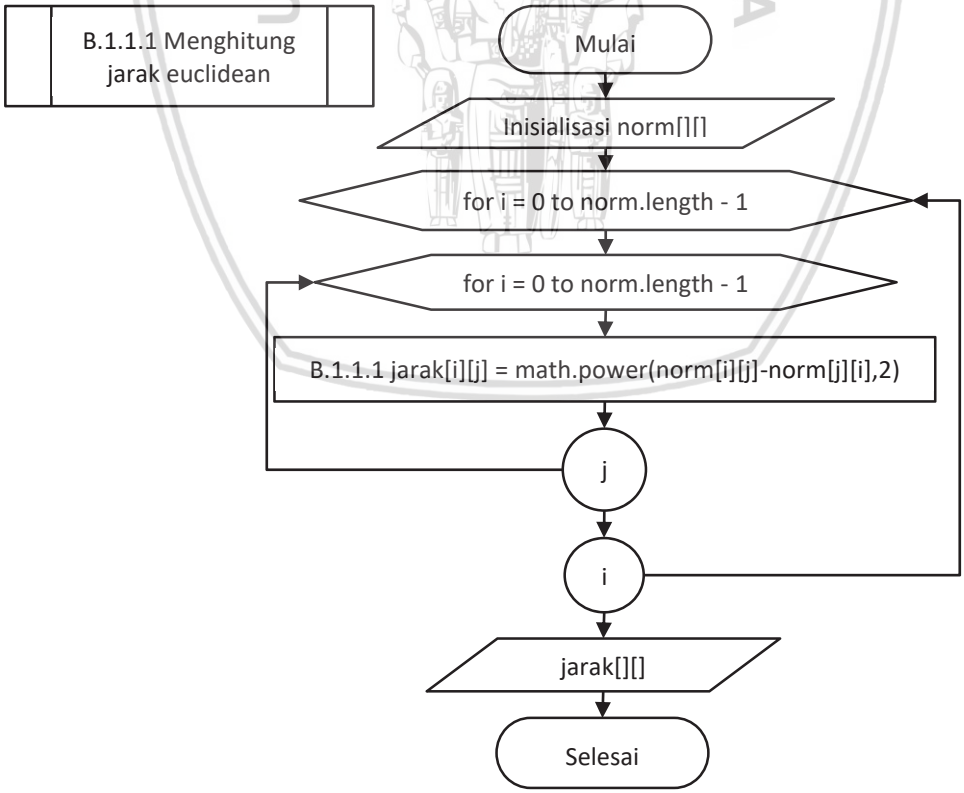
Dalam menghitung matriks hessian dengan menggunakan Persamaan (2.32) tidak perlu melibatkan data aktual. Pada Persamaan (2.32) nilai matriks hessian Rij didapatkan dari penjumlahan fungsi kernel $K(x_1,x_2)$ dengan nilai kuadrat parameter lambda (λ). Fungsi kernel digunakan dalam regresi untuk memetakan data inputan ke dalam ruang vektor. Pada penelitan ini fungsi kernel yang digunakan adalah Kernel RBF. Berikut diagram alir perhitungan matriks hessian yang ditunjukkan oleh Gambar 4.10



Gambar 4.10 Diagram alir menghitung matriks hessian

Tahap 1 : Menghitung jarak data latih

Menghitung jarak antar data latih di perlukan untuk perhitungan kernel RBF. Perhitungan jarak pada penelitian ini menggunakan rumus jarak Euclidean. Diagram alir perhitungan jarak Euclidean dapat dilihat pada Gambar 4.11 berikut.



Gambar 4.11 Diagram alir algoritme menghitung jarak euclidean

Penjelasan dari diagram alir algoritme menghitung jarak Euclidean akan dijabarkan pada pseudocode Gambar 4.12 berikut ini.

```
Nama algoritme : menghitung jarak euclidean
Deklarasi :
Double jarak[][]
Deskripsi :
  1. Input : norm[][]
  2. Proses :
       a. Perulangan i
       b. Perulangan j
          B.1.1.1 Menghitung jarak[i][j] += sum +
          Math.power(norm[i]-norm[j],2)
       c. End loop j
       d. End loop i
  3. Output : jarak[][]
```

Gambar 4.12 Pseudocode menghitung jarak euclidean

Berikut contoh perhitungan manual mencari jarak Euclidean data ke -1 dan data ke -2 pada data latih Tabel 4.12 dan hasil perhitungan jarak Euclidean seluruh data latih dapat dilihat pada Tabel 4.15.

$D_{1,2} = (0{,}14966 - 0{,}35374)^2 + (0{,}35374 - 0{,}35374)^2 + (0{,}35374 - 0{,}35374)^2 + (0 - 0{,}06803)^2$

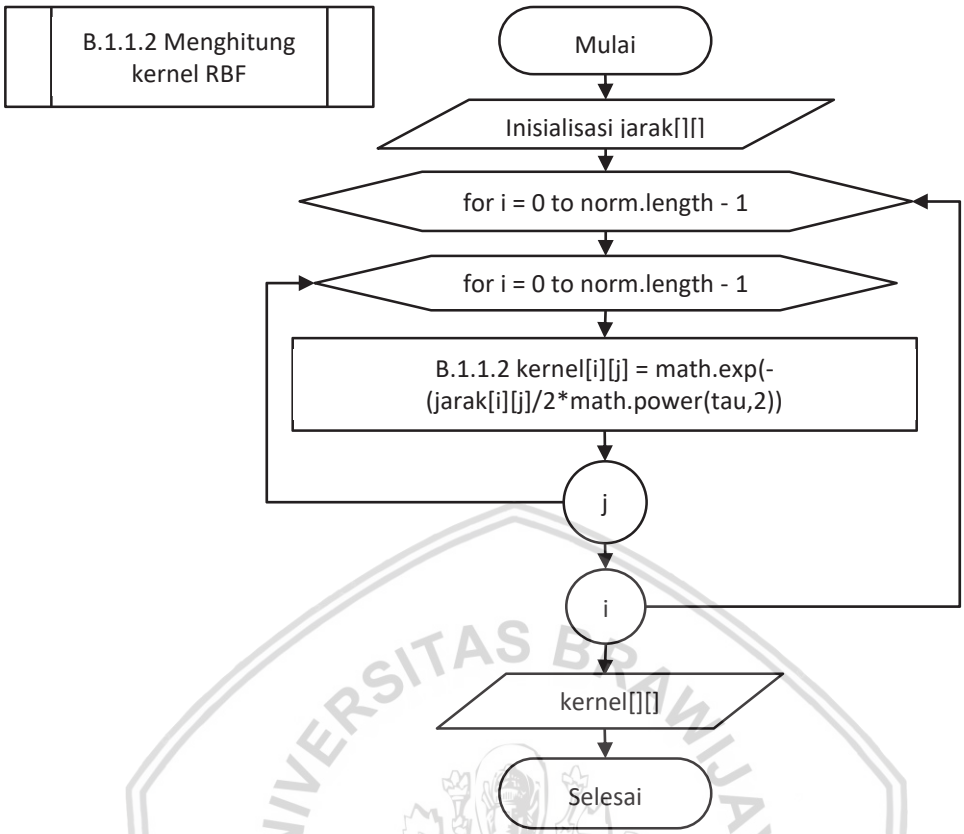
$= 0{,}16678$

Tabel 4.15 Hasil perhitungan jarak euclidean data latih

D_{ij}	1	2	3	4	5	6	7
1	0	0,166782	0,248415	0,282151	0,151141	0,178907	0,202277
2	0,166782	0	0,129761	0,235689	0,297191	0,128743	0,182655
3	0,248415	0,232967	0	0,140173	0,264612	0,176686	0,048267
4	0,282151	0,362008	0,219123	0	0,144801	0,168402	0,099681
5	0,151141	0,680767	0,363014	0,233381	0	0,076357	0,10528
6	0,178907	1,006499	0,708759	0,423958	0,246099	0	0,082142
7	0,202277	1,372588	0,974154	0,601488	0,288129	0,041218	0

Tahap 2 : Menghitung nilai kernel RBF

Setelah menghitung jarak data, selanjutnya adalah menghitung nilai kernel RBF dengan menggunakan rumus pada Persamaan (2.8). Diagram alir perhitungan kernel RBF dapat dilihat pada Gambar 4.13 berikut.



Gambar 4.13 Diagram alir algoritme menghitung kernel RBF

Penjabaran diagram alir algoritme perhitungan kernel RBF diatas dapat dilihat pada pseudocode Gambar 4.14 berikut.

```
Nama Algoritme : Menghitung Kernel RBF
Deklarasi :
Double kernel[][]
Deskripsi
 1. Input : jarak[][]
 2. Proses :
    B.1.1.2 Kernel[i][j] = math.exp(-
    (jarak[i][j]/2*math.power(tau,2))
 3. Output : kernel[i][j]
```

Gambar 4.14 Pseudocode menghitung kernel RBF

Berikut contoh manualisasi perhitungan kernel $K(X_1,X_2)$ inisialisasi jarak euclidean data latih Tabel 4.15 dan hasil perhitungan kernel semua data latih dapat dilihat pada Tabel 4.16 berikut.

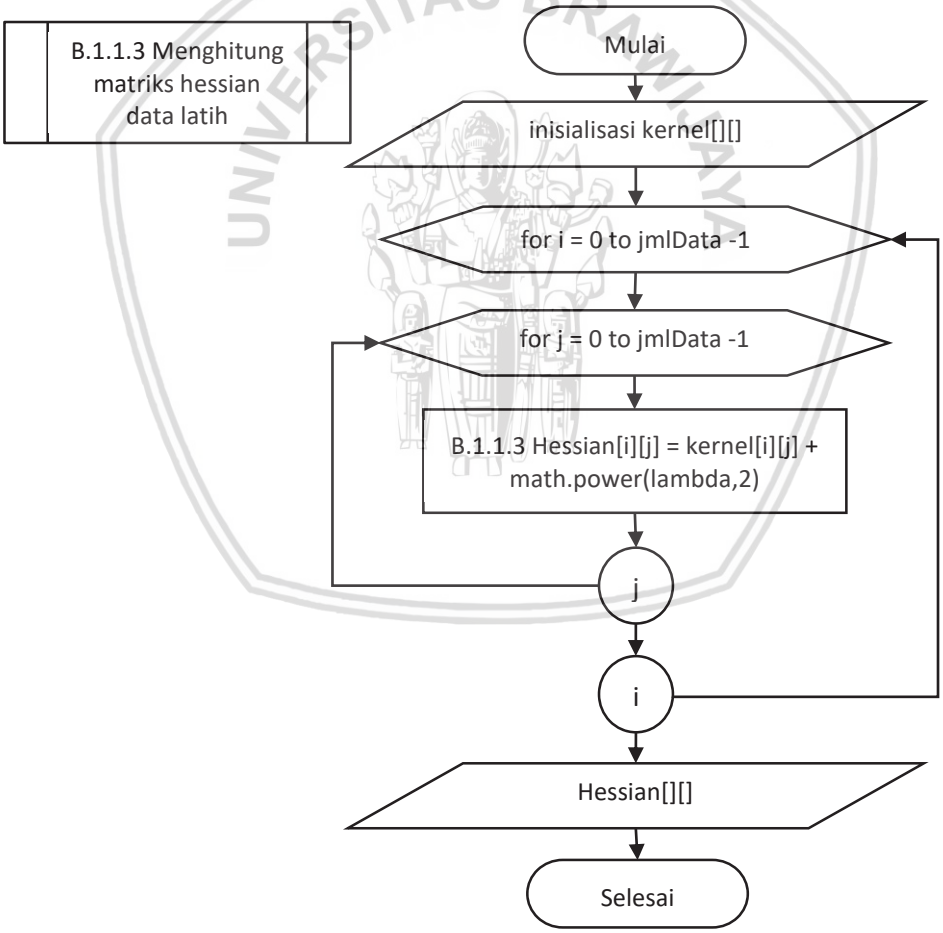
$$\mathrm{K}(x_1, x_2) = EXP\left(-\frac{0{,}16678}{(2 \times 0{,}1)^2}\right) = 0{,}015459$$

Tabel 4.16 Hasil perhitungan nilai kernel RBF

Kernel	1	2	3	4	5	6	7
1	1	0,01546	0,00201	0,00086	0,02286	0,01142	0,00637
2	0,01546	1	0,03901	0,00276	0,00059	0,04001	0,0104
3	0,00201	0,00296	1	0,03007	0,00134	0,01207	0,29919
4	0,00086	0,00012	0,00418	1	0,02678	0,01485	0,08274
5	0,02286	$4,1x10^{-8}$	0,00011	0,00293	1	0,14824	0,07193
6	0,01142	$1,2x10^{-11}$	$2,02x10^{-8}$	$2,5 x10^{-5}$	0,002128	1	0,12828
7	0,00637	$1,2x10^{-15}$	$2,7x10^{-11}$	$2,95x10^{-7}$	0,000744	0,356842	1

Tahap 3 : Menghitung nilai matriks hessian data latih

Selanjutnya adalah menghitung matriks hessian data latih. Dalam menghitung matriks hessian diperlukan nilai kernel dan nilai lambda. Berikut diagram alir perhitungan matriks hessian yang dapat dilihat pada Gambar 4.15.



Gambar 4.15 Diagram alir algoritme menghitung matriks hessian

Penjelasan dari diagram alir diatas dapat dilihat pada pseudocode Gambar 4.16 berikut.

```
Nama Algoritme : Menghitung Matriks Hessian
Deklarasi :
Double hessian[][]
Deskripsi
 1. Input : kernel[][]
 2. Proses :
    B.1.1.3 Hessian[i][j] = kernel[i][j] + math.power(lambda,2)
 3. Output : hessian[i][j]
```

Gambar 4.16 Pseudocode menghitung matriks hessian

Berikut contoh manualisasi perhitungan matriks hessian $R_{1,2}$ dan $R_{2,1}$ dan hasil perhitungan matriks *hessian* seluruh data latih dapat dilihat pada Tabel 4.17.

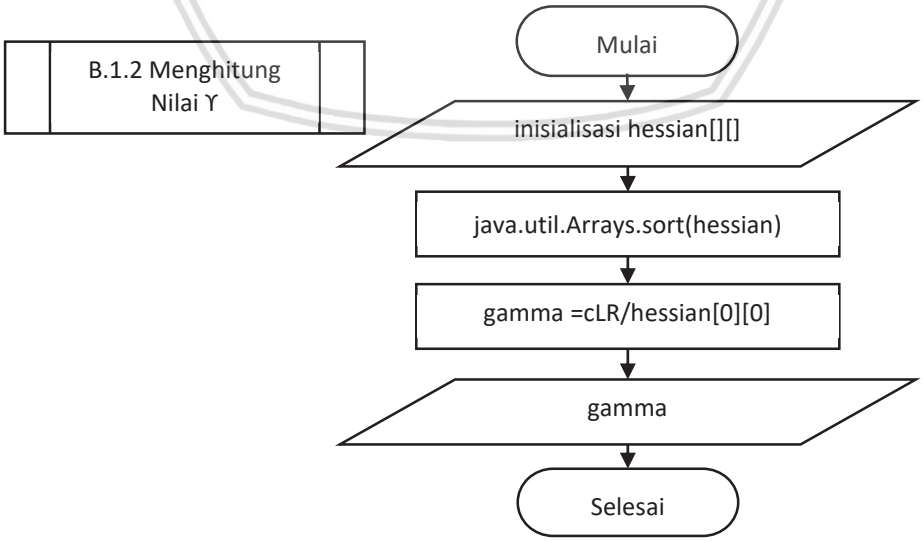
$R_{1,2} = R_{2,1} = 0{,}01546 + (0{,}6914)^2 = 0{,}49349$

Tabel 4.17 Hasil perhitungan matriks hessian data latih

R_{ij}	1	2	3	4	5	6	7
1	1,47803	**0,49349**	0,48004	0,47890	0,50089	0,48945	0,48440
2	**0,49349**	1,47803	0,51704	0,48080	0,47863	0,51805	0,48843
3	0,48004	0,48099	1,47803	0,50810	0,47937	0,49010	0,77723
4	0,47890	0,47815	0,48221	1,47803	0,50482	0,49288	0,56078
5	0,50089	0,47803	0,47815	0,48096	1,47803	0,62627	0,54997
6	0,48945	0,47803	0,47803	0,47806	0,48016	1,47803	0,60631
7	0,48440	0,47803	0,47803	0,47804	0,47878	0,83488	1,47803

B.1.2 Menghitung Nilai ϒ

Nilai gamma didapatkan dari pembagian nilai parameter cLR dan nilai maksimal matriks hessian. Rumus penghitungan nilai gamma dapat dilihat pada Persamaan (2.33). Berikut diagram alir proses perhitungan nilai gamma ditunjukkan oleh Gambar 4.17.



Gambar 4.17 Diagram alir algoritme menghitung nilai ϒ

Penjabaran diagram alir menghitung nilai ϒ diatas dapat dilihat pada pseudocode Gambar 4.18 berikut.

```
Nama Algoritme : Menghitung Matriks Hessian
Deklarasi :
Double gamma
Deskripsi
 1. Input : hessian[][]
 2. Proses :
    a. sort(hessian)
    b. gamma = cLR/hessian[0][0]
 3. Output : hessian[i][j]
```

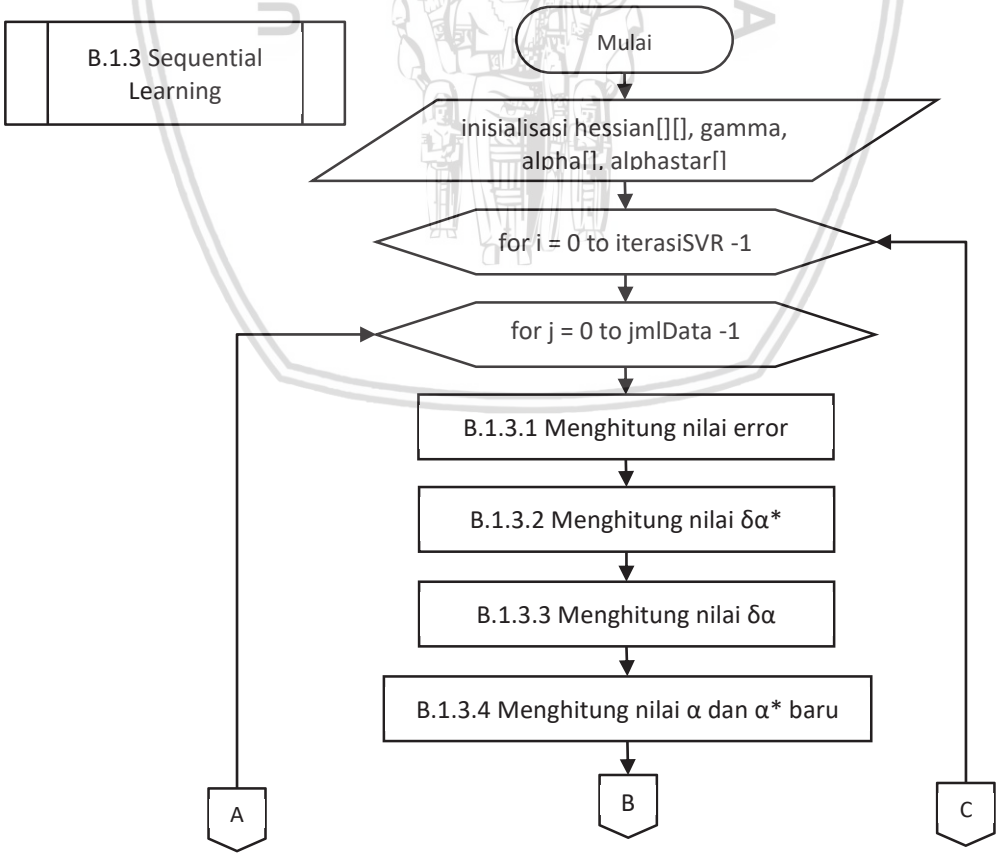
Gambar 4.18 Pseudocode menghitung nilai ϒ

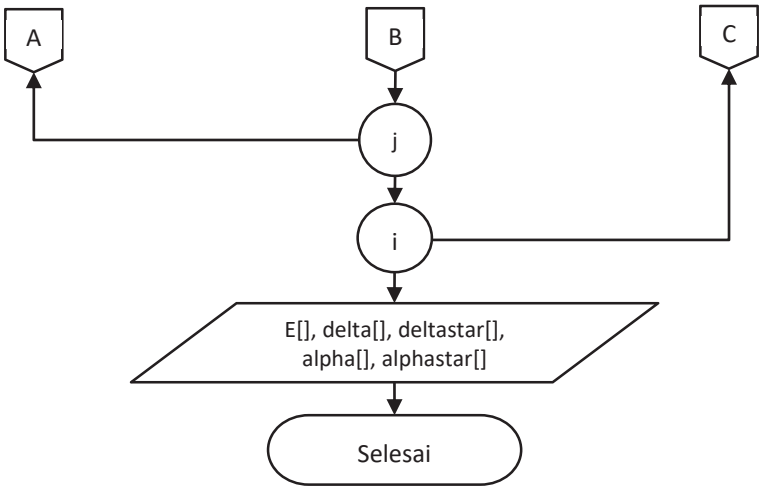
Berikut manualisasi perhitungan nilai gamma studi kasus matriks hessian Tabel 4.17 dan inisialisasi parameter Tabel 4.13

$$\gamma = \frac{0{,}5925}{MAX(HESSIAN)} = \frac{0{,}5925}{1{,}47803} = 0{,}40087$$

B.1.3 Sequential Learning

Sequential learning terdiri dari 4 tahap yaitu, menghitung nilai *error*, menghitung nilai δα*, menghitung nilai δα, dan menghitung nilai α* dan α. Hasil akhir dari sequential adalah nilai *error*, δα*, δα, α* dan α yang nantinya akan dikonversi menjadi nilai regresi hasil peramalan SVR. Berikut diagram alir sequential learning dapat dilihat pada Gambar 4.19 berikut.

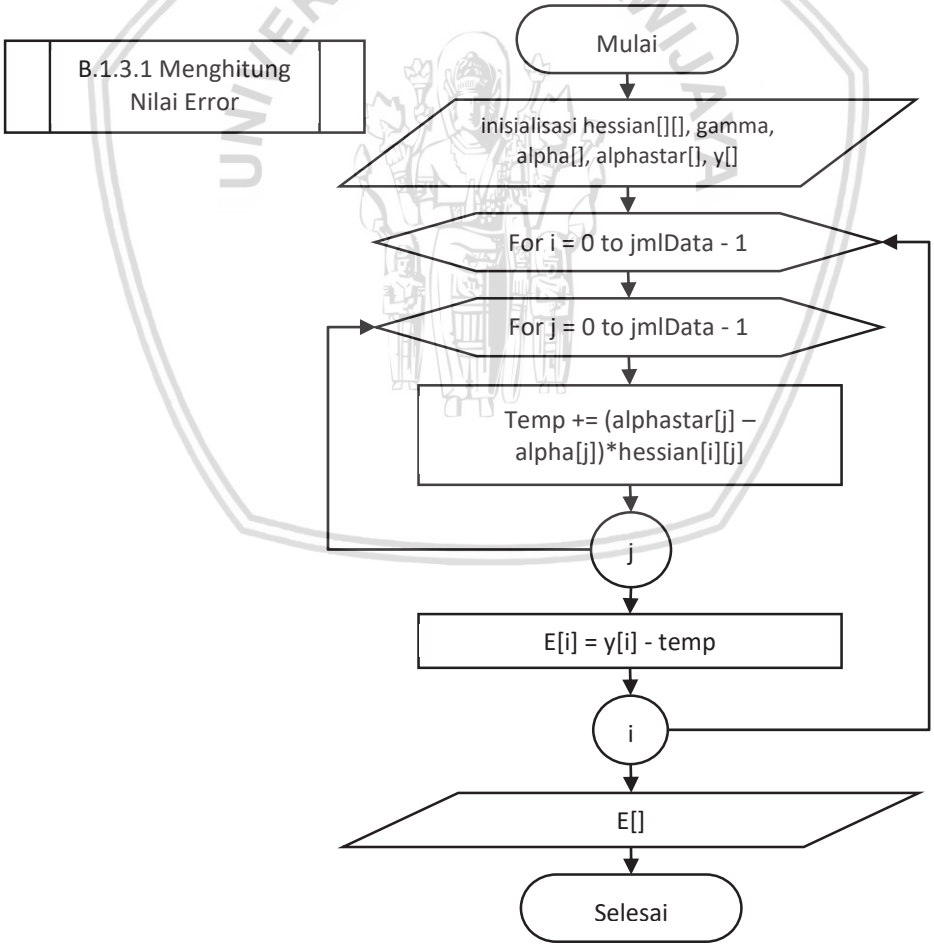




Gambar 4.19 Diagram alir sequential learning

B.1.3.1 Menghitung Nilai Error

Menghitung nilai error untuk setiap data latih menggunakan Persamaan (2.34). Berikut diagram alir menghitung nilai error ditunjukkan oleh Gambar 4.20 berikut.



Gambar 4.20 Diagram alir algoritme menghitung nilai error

Penjabaran dari diagram alir algoritme menghitung nilai error ditunjukkan oleh pseudocode Gambar 4.21 berikut.

```
Nama Algoritme : Menghitung Nilai Error
Deklarasi :
Double temp,E[]
Deskripsi
  1. Input : hessian[][], y[], alpha[], alphastar[]
  2. Proses :
     temp = (alphastar[] – alpha[])*hessian[][]
     E[] = y[] – temp
     temp = 0
  3. Output : E[]
```

Gambar 4.21 Pseudocode menghitung nilai error

Berikut contoh manualisasi penghitungan nilai error iterasi ke - 1 untuk data latih ke – 1 dengan inisialisasi matriks hessian Tabel 4.17, inisialisasi data latih Tabel 4.13 dan inisialisasi nilai α dan α* Tabel 4.14.

E(1) = 0 – ((0-0)*1,47803 + (0-0)*0,49349 + (0-0)*0,48004 +

(0-0)*0,47890 + (0-0)*0,50089 + (0-0)*0,48945 + (0-0)*0,48440

= **0**

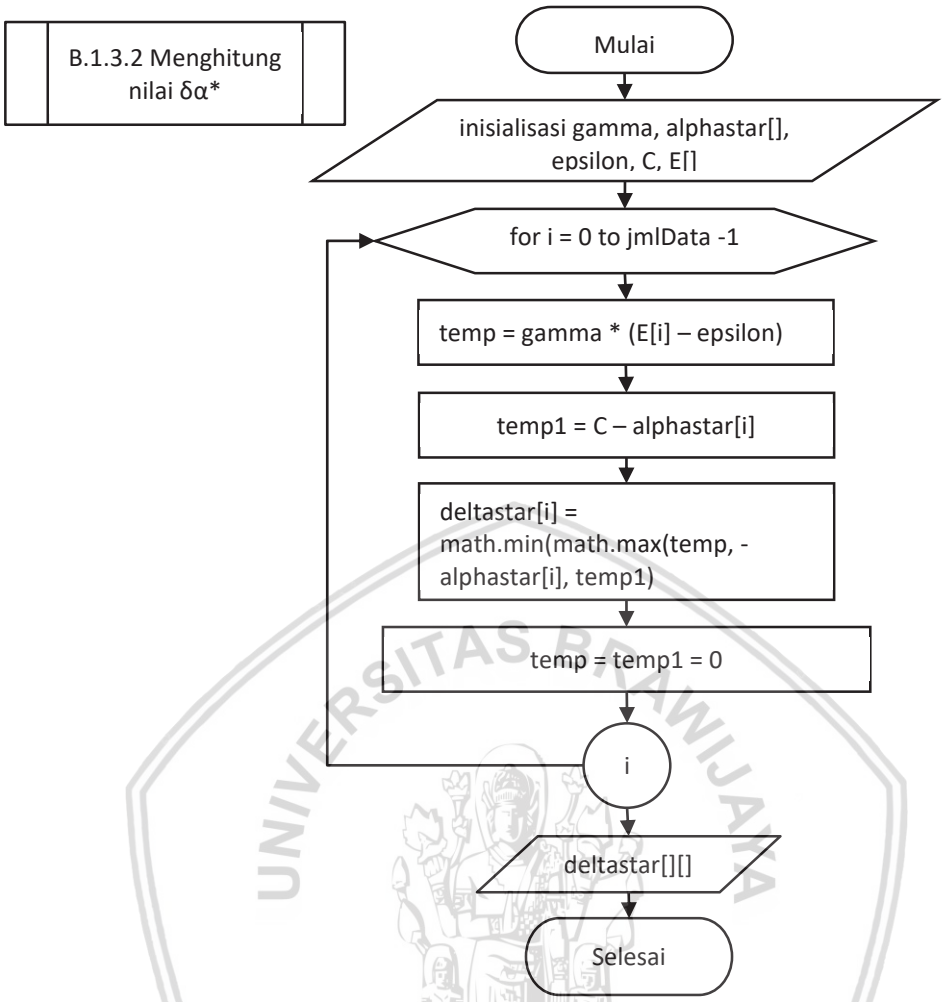
Berikut tabel hasil perhitungan nilai error Ei untuk seluruh data latih iterasi ke – 1 yang ditunjukkan oleh Tabel 4.18 berikut

Tabel 4.18 Hasil perhitungan nilai error iterasi ke – 1

Data ke -	Ei
1	0
2	0,068027
3	0,170068
4	0,238095
5	0
6	0,102041
7	0,319728

B.1.3.2 Menghitung Nilai δα*

Untuk menghitung nilai δα* (batas atas nilai *lagrange multiplier*) untuk setiap data latih menggunakan Persamaan (2.35). Berikut diagram alir menghitung nilai δα* yang ditunjukkan oleh Gambar 4.22 berikut.



Gambar 4.22 Diagram alir algoritme menghitung nilai δα*

Penjabaran dari diagram alir menghitung nilai δα* diatas dapat dilihat pada pseudocode Gambar 4.23 berikut.

```
Nama Algoritme : Menghitung Nilai δα*
Deklarasi :
Double temp, temp1, deltastar[]
Deskripsi
  1. Input : alphastar[], E[], gamma, epsilon, C
  2. Proses :
     temp = gamma*(E[i] - epsilon)
     temp1 = C - alphastar[i]
     deltastar[i] = math.min(math.max(temp, -
     alphastar[i]),temp1)
     temp = temp1 = 0
  3. Output : deltastar[i]
```

Gambar 4.23 Pseudocode menghitung nilai δα*

Berikut contoh manualisasi perhitungan nilai δα* iterasi ke – 1 untuk data latih ke -1 dengan inisialisasi nilai α* Tabel 4.14, inisialisasi nilai error Tabel 4.18, dan iterasi parameter SVR tabel 4.13.

δα[1]* = MIN(MAX((0.40087(0 – 0.00599)), -0), 570.511 – 0) = 0

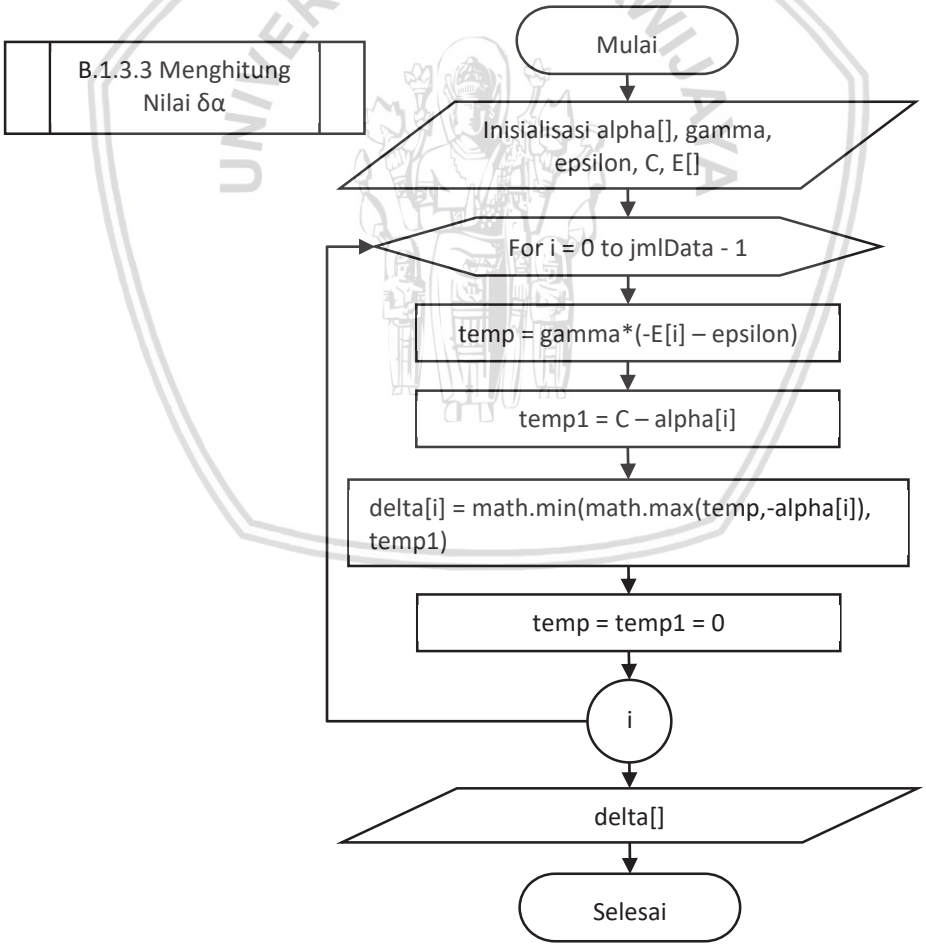
Berikut tabel hasil perhitungan nilai δα* untuk seluruh data latih iterasi ke – 1 yang ditunjukkan oleh Tabel 4.19 berikut.

Tabel 4.19 Hasil perhitungan nilai δα* iterasi ke – 1

Data ke -	$\delta_{\alpha i}$*
1	0
2	0,02487
3	0,065775
4	0,093045
5	0
6	0,038505
7	0,125769

B.1.3.3 Menghitung Nilai δα

Untuk menghitung nilai δα (batas bawah nilai *lagrange multiplier*) untuk setiap data latih menggunakan Persamaan (2.36). Berikut diagram alir menghitung nilai δα yang ditunjukkan oleh Gambar 4.24 berikut



Gambar 4.24 Diagram alir menghitung nilai δα

Penjabaran dari diagram alir menghitung nilai δα diatas dapat dilihat pada pseudocode Gambar 4.25 berikut.

```
Nama Algoritme : Menghitung Nilai δα*
Deklarasi :
Double temp, temp1, deltastar[]
Deskripsi
  1. Input : alpha[], E[], gamma, epsilon, C
  2. Proses :
     temp = gamma*(-E[i] – epsilon)
     temp1 = C – alpha[i]
     delta[i] = math.min(math.max(temp, -alpha[i]),temp1)
     temp = temp1 = 0
  3. Output : delta[i]
```

Gambar 4.25 Pseudocode menghitung nilai δα*

Berikut contoh manualisasi perhitungan nilai δα iterasi ke – 1 untuk data latih ke - 1 dengan inisialisasi nilai α Tabel 4.14, inisialisasi nilai error Tabel 4.18, dan inisialisasi parameter SVR tabel 4.13.

δα[1] = MIN(MAX((0.40087(-0 – 0.00599)), -0), 570.5111 – 0) = 0

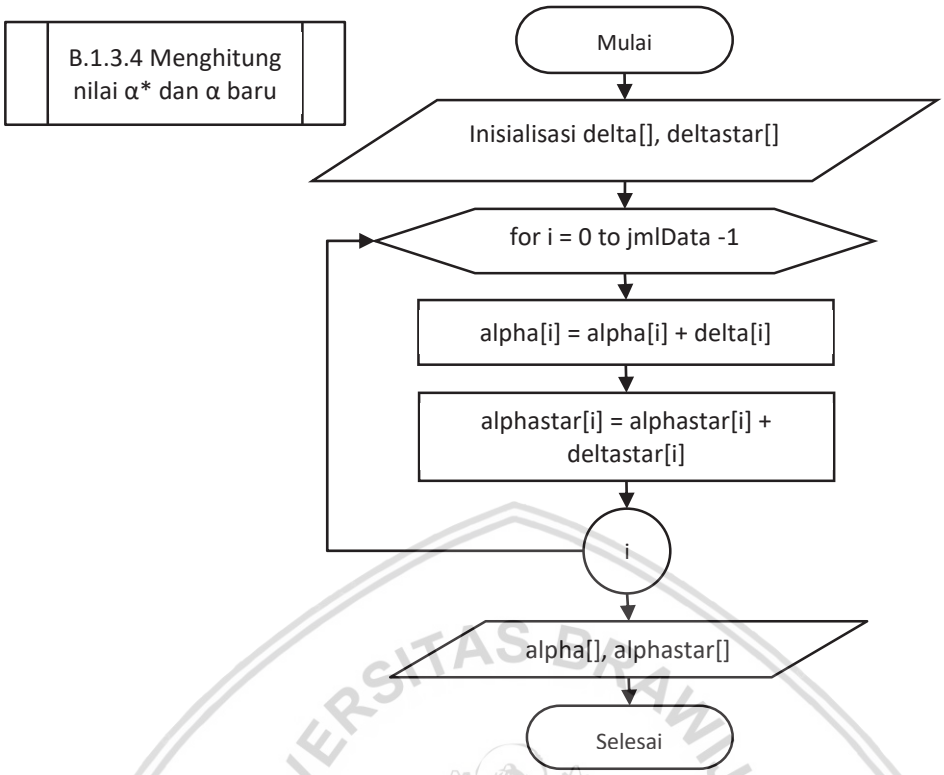
Berikut tabel hasil perhitungan nilai δα untuk seluruh data latih iterasi ke – 1 yang ditunjukkan oleh Tabel 4.20 berikut.

Tabel 4.20 Hasil perhitungan nilai δα iterasi ke – 1

Data ke -	$\delta\alpha_i$
1	0
2	0
3	0
4	0
5	0
6	0
7	0

B.1.3.4Menghitung Nilai α* dan α baru

Untuk menghitung nilai α* dan α baru untuk setiap data latih menggunakan Persamaan (2.37) dan Persamaan (2.38). Berikut diagram alir menghitung nilai α* dan α baru yang ditunjukkan oleh Gambar 4.26 berikut.



Gambar 4.26 Diagram alir algoritme menghitung nilai α* dan α baru

Penjabaran dari diagram alir menghitung nilai α* dan α baru diatas dapat dilihat pada pseudocode Gambar 4.27 berikut.

```
Nama Algoritme : Menghitung Nilai α* dan α baru
Deklarasi :
alpha[], alphastar[]
Deskripsi
  1. Input : delta[], deltastar[]
  2. Proses :
     alpha[i] = alpha[i] + delta[i]
     alphastar[i] = alphastar[i] + deltastar[i]
  3. Output : alpha[], alphastar[]
```

Gambar 4.27 Pseudocode algoritme menghitung nilai α* dan α

Berikut contoh manualisasi perhitungan nilai α* dan α iterasi ke – 1 untuk data latih ke - 2 dengan inisialisasi nilai α* dan α Tabel 4.14, inisialisasi nilai δα* Tabel 4.19, dan inisialisasi nilai δα Tabel 4.20.

α[2]* = 0 + 0,02487 = 0,02487

α[2] = 0 + 0 = 0

Berikut tabel hasil perhitungan nilai α* dan α baruuntuk seluruh data latih iterasi ke – 1 yang ditunjukkan oleh Tabel 4.21 berikut.

Tabel 4.21 Hasil perhitungan nilai α* dan α iterasi ke – 1

Data ke -	αi*	αi*
1	0	0
2	**0,02487**	**0**
3	0,065775	0
4	0,093045	0
5	0	0
6	0,038505	0
7	0,125769	0

Tahap menghitung nilai error, tahap menghitung nilai δα*, tahap menghitung nilai δα, dan tahap menghitung nilai α dan α* dilakukan secara sekuensial sampai iterasi maksimal yang telah ditentukan sebelumnya yaitu sebanyak 10 iterasi. Berikut tabel hasil sequential learning pada iterasi ke – 1 dan iterasi terakhir yang di tunjukkan oleh Tabel 4.22 dan Tabel 4.23

Tabel 4.22 Hasil sequential learning iterasi ke – 1

Iterasi 1					
Data ke	**Ei**	**δαi***	**δαi**	**αi***	**αi**
1	0	0	0	0	0
2	0,068027	0,02487	0	0,02487	0
3	0,170068	0,065775	0	0,065775	0
4	0,238095	0,093045	0	0,093045	0
5	0	0	0	0	0
6	0,102041	0,038505	0	0,038505	0
7	0,319728	0,125769	0	0,125769	0

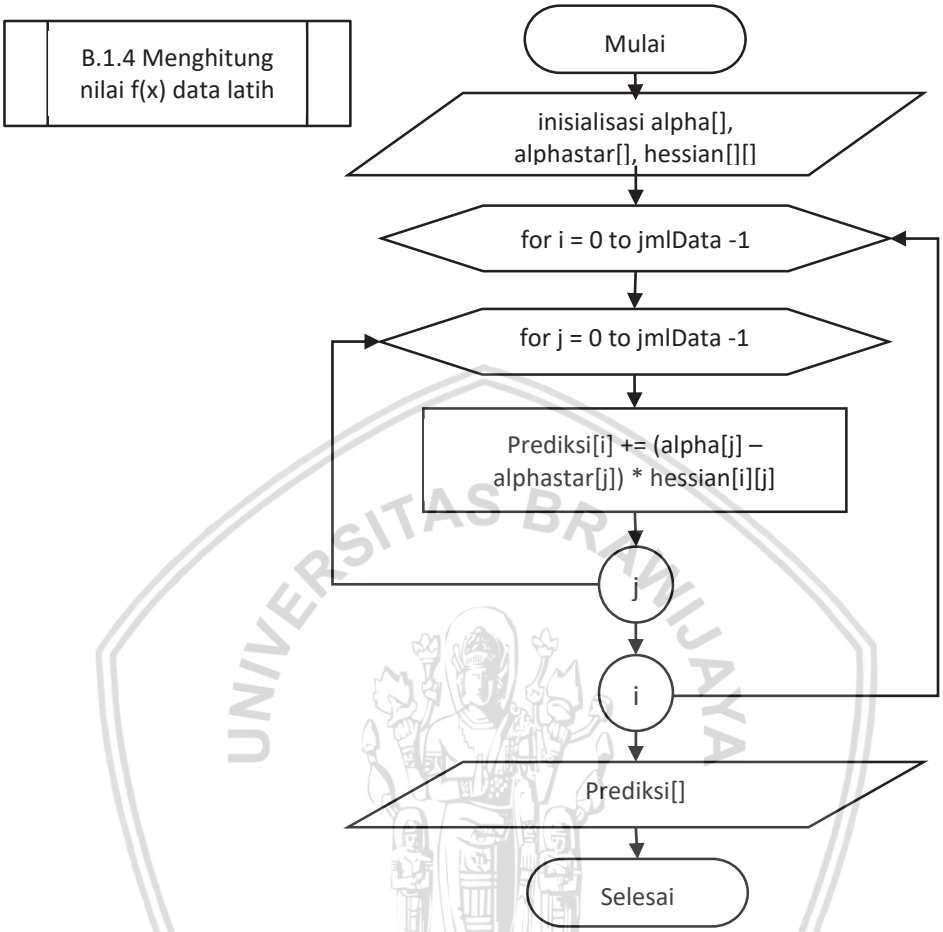
Tabel 4.23 Hasil sequential learning iterasi terakhir

Iterasi 10					
Data ke -	**Ei**	**δαi***	**δαi**	**αi***	**αi**
1	-9,835455	-2,13521	3,940341	0	3,940341
2	-9,932215	-2,18578	3,979129	0	3,979129
3	-10,59924	-2,36993	4,24652	0	4,24652
4	-9,885311	-2,25048	3,960327	0	3,960327
5	-10,37815	-2,2521	4,157891	0	4,157891
6	-10,02342	-2,21843	4,015692	0	4,015692
7	-10,47332	-2,4186	4,196042	0	4,196042

B.1.4 Menghitung Nilai f(x) Data Latih

Setelah sequential learning mencapai iterasi maksimalnya, selanjutnya adalah menghitung nilai regresi f(x) atau nilai hasil prediksi dengan menggunakan rumus

pada Persamaan (2.40). Nilai f(x) didapatkan dari nilai α dan α* iterasi terakhir dan matriks hessian. Berikut diagram alir menghitung nilai f(x) dapat dilihat pada Gambar 4.28.



Gambar 4.28 Diagram alir algoritme menghitung nilai f(x) data latih

Penjabaran dari diagram alir menghitung nilai f(x) data latih ditunjukkan oleh pseudocode Gambar 4.29 berikut.

```
Nama Algoritme : Menghitung Nilai f(x) data latih
Deklarasi :
Prediksi[]
Deskripsi
  1. Input : alpha[], alphastar[], hessian
  2. Proses :
     Prediksi[] = (alphastar[] – alpha[])*hessian[][]
  3. Output : prediksi[]
```

Gambar 4.29 Pseudocode menghitung f(x) data latih

Berikut manualisasi menghitung nilai f(x) data latih ke – 1 dengan inisialisasi nilai matriks hessian Tabel 4.17 dan inisialisasi nilai α dan α* Tabel 4.21.

$$f(x1) = ((0-0) \times 3{,}94034) + ((0-3{,}97913) \times 0{,}49349) + ((0-4{,}24652) \times 0{,}48004) + ((0-3{,}96033) \times 0{,}47890) + ((0-4{,}15789) \times 0{,}50089) + ((0-4{,}01569) \times 0{,}48945) + ((0-4{,}19604) \times 0{,}48440) = -17{,}8034$$

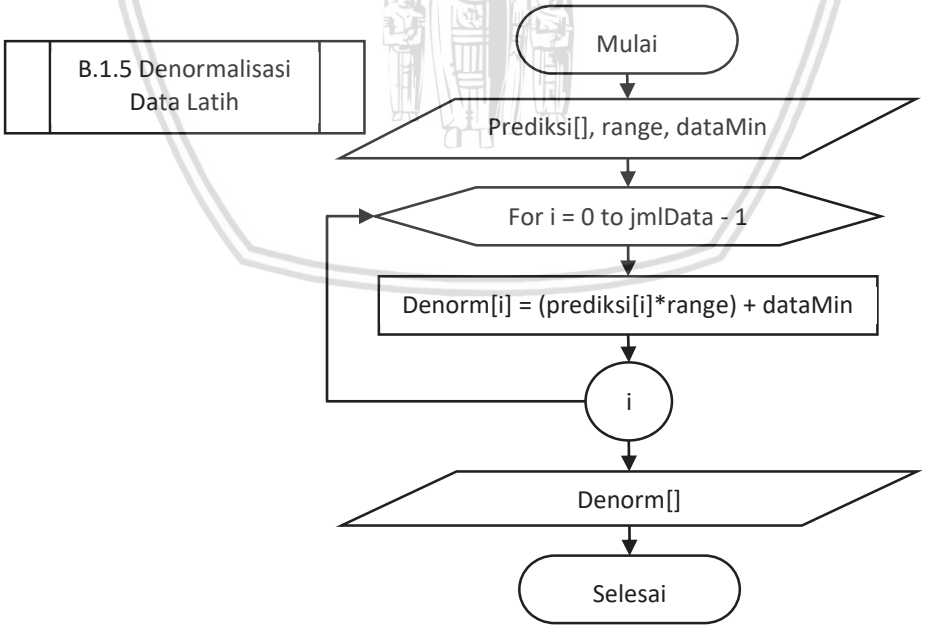
Berikut hasil perhitungan nilai f(x) untuk semua data latih yang ditunjukkan oleh Tabel 4.24

Tabel 4.24 Hasil perhitungan nilai f(x) data latih

Data latih ke-i	f(x)
1	-17,8034
2	-18,0454
3	-19,3168
4	-18,1222
5	-18,7792
6	-18,2299
7	-19,2792

B.1.5 Denormalisasi Data Latih

Denormalisasi mengubah data hasil prediksi f(x) ke dalam *range* data aktual, bukan *range* data normalisasi. Proses denormalisasi ini bertujuan untuk membandingkan hasil prediksi dengan data aktual sehingga nilai *error* dapat diketahui secara pasti. Diagram alir denormalisasi data latih ditunjukkan oleh Gambar 4.30 berikut.



Gambar 4.30 Diagram alir algoritme denormalisasi data latih

Penjabaran dari diagram alir denormalisasi data latih dapat dilihat pada pseudocode Gambar 4.31 berikut.

```
Nama Algoritme : Denormalisasi data latih
Deklarasi :
Denorm[]
Deskripsi
  1. Input : prediksi[], range, dataMin
  2. Proses :
     Denorm[] = (prediksi[]*range)+dataMin
  3. Output : denorm[]
```

Gambar 4.31 Pseudocode denormalisasi data latih

Berikut contoh manualisasi denormalisasi data latih ke – 1 dengan inisialisasi nilai range dan data minimal Tabel 4.6 dan inisialisasi nilai f(x) prediksi Tabel 4.24 serta hasil perhitungan seluruh data latih yang ditunjukkan oleh Tabel 4.25.

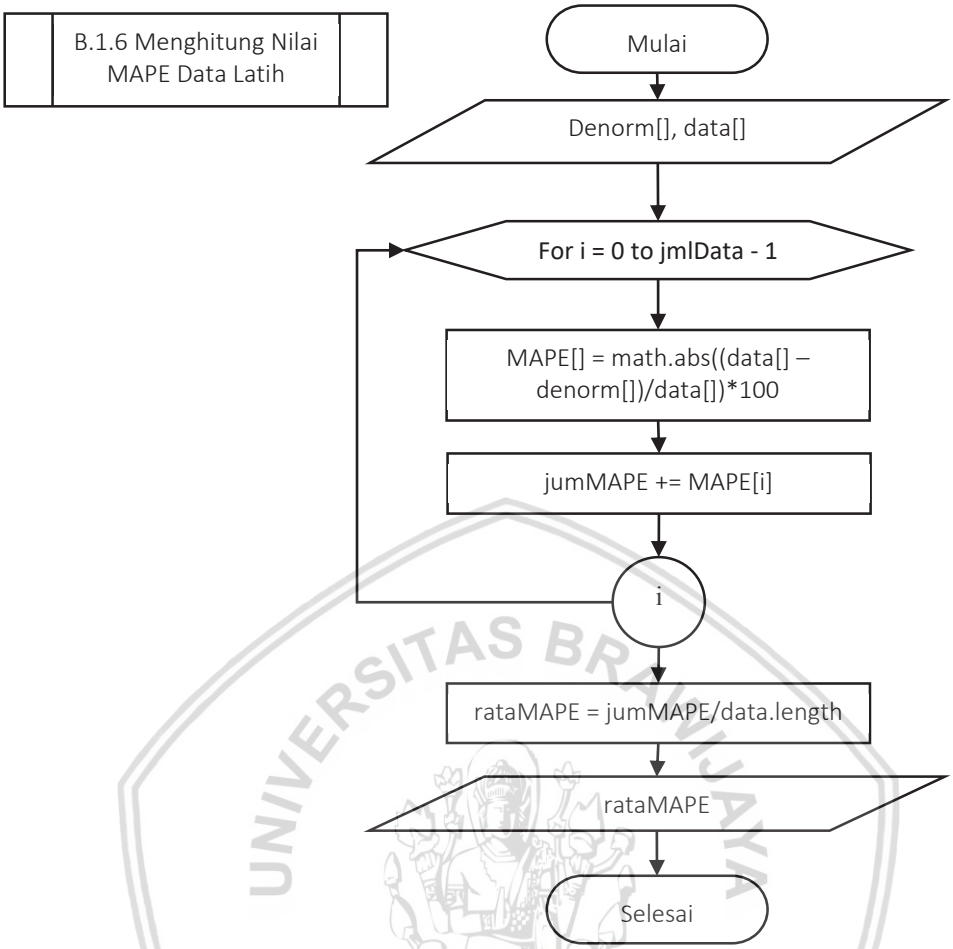
Denormalisasi[1] = (-17,803 x 490) + 18076,7 = 9352,978

Tabel 4.25 Hasil denormalisasi data latih

Data latih ke-i	f(x) Denormalisasi
1	9352,978
2	9234,403
3	8611,449
4	9196,811
5	8874,86
6	9143,994
7	8629,84

B.1.6 Menghitung Nilai MAPE data latih

Setelah seluruh hasil prediksi data latih didenormalisasi, selanjutnya adalah menghitung nilai MAPE menggunakan Persamaan (2.53) dan merata-rata nilai MAPE-nya. Nilai MAPE didapatkan dari nilai mutlak selisih data hasil prediksi dengan data aktual lalu dipersentasekan. Diagram alir menghitung nilai MAPE ditunjukkan oleh Gambar 4.32 berikut.



Gambar 4.32 Diagram alir algoritme menghitung nilai mape data latih

Penjabaran dari diagram alir menghitung nilai MAPE data latih dapat dilihat pada pseudocode Gambar 4.33 berikut

```
Nama Algoritme : Menghitung Nilai MAPE data latih
Deklarasi :
MAPE[], rataMAPE, jumMAPE
Deskripsi
  1. Input : denorm[], data[]
  2. Proses :
     MAPE[] = math.abs((denorm[]-data[])/data[])
     jumMAPE += MAPE[]
     rataMAPE = jumMAPE/data.length
  3. Output : rataMAPE
```

Gambar 4.33 Pseudocode menghitung nilai MAPE data latih

Berikut contoh perhitungan nilai MAPE data latih ke – 1 dengan inisialisasi data denormalisasi Tabel 4.25 beserta tabel hasil perhitungan Nilai MAPE data latih untuk partikel X_1 PSO yang ditunjukkan oleh Tabel 4.26.

$$MAPE(1) = \left|\frac{9352{,}98 - 18076{,}7}{9352{,}98}\right| \times 100 = 48{,}259$$

Tabel 4.26 Hasil perhitungan nilai MAPE data latih partikel X_1

Data latih ke-i	MAPE %	Rata Rata
1	48,25938846	50,34682375%
2	49,00936903	
3	52,58012922	
4	49,44955422	
5	50,90433066	
6	49,55501511	
7	52,66997955	

B.1.7 Menghitung Nilai *Fitness*

Nilai *fitness* adalah parameter pengukuran kualitas suatu partikel. Pada kasus ini nilai *fitness* berbanding terbalik dengan nilai MAPE yang dihasilkan satu partikel dari pelatihan SVR. Dengan menggunakan Persamaan (2.49), berikut tabel nilai *fitness* hasil pelatihan SVR untuk semua partikel inisialisasi pada Tabel 4.10 yang ditunjukkan oleh Tabel 4.27.

Tabel 4.27 Nilai *fitness* PSO iterasi ke – 0

Partikel (X_i)	Λ	C	ε	cLR	MAPE	*Fitness*
X_1	0,691401	570,5107	0,005988	0,5925	50,34682	0,019475
X_2	0,540425	136,5756	0,005018	0,2539	0,035947	0,965301
X_3	0,772748	74,43793	0,007109	0,390051	0,029417	0,971424

B.2 Evaluasi Nilai PBest

Pada evaluasi nilai PBest ini, PBest iterasi terkini diambil dari nilai *fitness* terbaik setiap partikel yang dibandingkan dengan nilai *fitness* yang dihasilkan pada iterasi sebelumnya.

Berikut contoh evaluasi Nilai PBest pada PSO Iterasi ke – 2, Nilai PBest iterasi ke – 1 ditunjukkan oleh Tabel 4.28, hasil nilai *fitness* iterasi ke – 2 ditunjukkan oleh Tabel 4.29, dan hasil evaluasi PBest iterasi ke – 2 ditunjukkan oleh Tabel 4.30.

Tabel 4.28 PBest iterasi ke – 1

PBest[1]	λ	C	ε	cLR	*Fitness*
$PBest_1$	0,719872	225,5704	0,006813	0,42039	0,978661
$PBest_2$	0,540425	136,5756	0,005018	0,2539	0,965301
$PBest_3$	0,772748	74,43793	0,007109	0,390051	0,971424

Tabel 4.29 Hasil perhitungan nilai *fitness* iterasi ke – 2

Partikel	λ	C	ε	cLR	MAPE	*Fitness*
X_1	1	338,3556	0,01	0,630585	2451,209	0,000407796
X_2	1	226,3027	0,01	0,576722	915,7445	0,001090816
X_3	1	130,0692	0,01	0,592986	1245,956	0,000801953

Tabel 4.30 Hasil evaluasi pbest iterasi ke – 2

PBest[1]	λ	C	ε	cLR	*Fitness*
$PBest_1$	0,719872	225,5704	0,006813	0,42039	0,978661
$PBest_2$	0,540425	136,5756	0,005018	0,2539	0,965301
$PBest_3$	0,772748	74,43793	0,007109	0,390051	0,971424

B.3 Evaluasi Nilai GBest

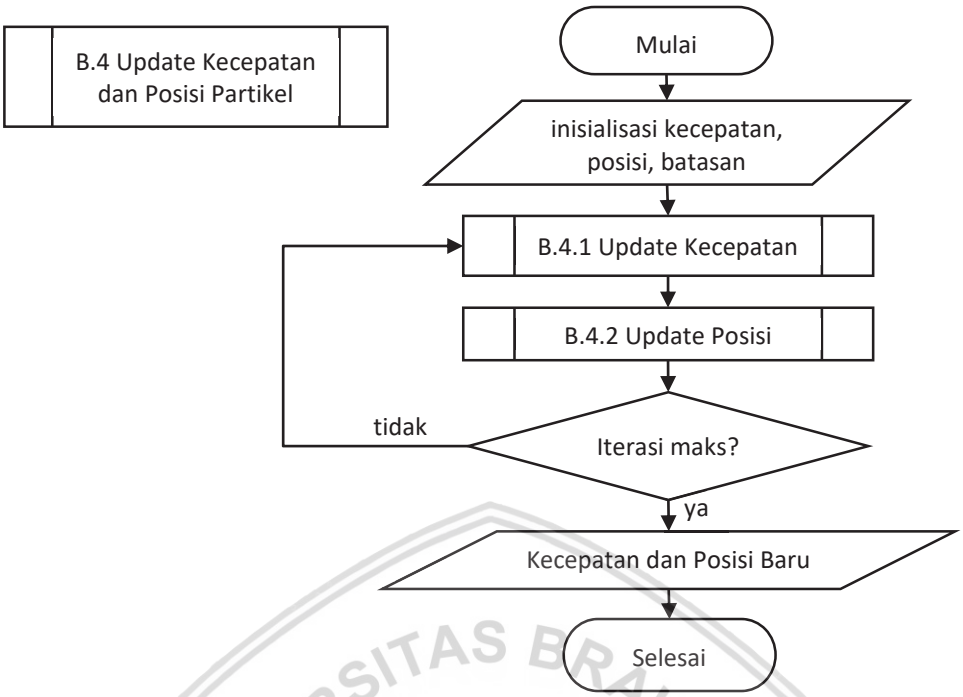
Setelah evaluasi PBest, selanjutnya menentukan Global Best (GBest). Nilai GBest didapatkan dari partikel dengan nilai PBest terbaik. Berikut ini tabel hasil evaluasi GBest untuk iterasi ke – 2 yang ditunjukkan oleh Tabel 4.31.

Tabel 4.31 Hasil evaluasi GBest iterasi ke – 2

PBest[1]	λ	C	ε	cLR	*Fitness*
$PBest_1$	0,719872	225,5704	0,006813	0,42039	0,978661
$PBest_2$	0,540425	136,5756	0,005018	0,2539	0,965301
$PBest_3$	0,772748	74,43793	0,007109	0,390051	0,971424
GBest(1)	**0,719872**	**225,5704**	**0,006813**	**0,42039**	**0,978661**

B.4 Update Kecepatan dan Posisi Partikel

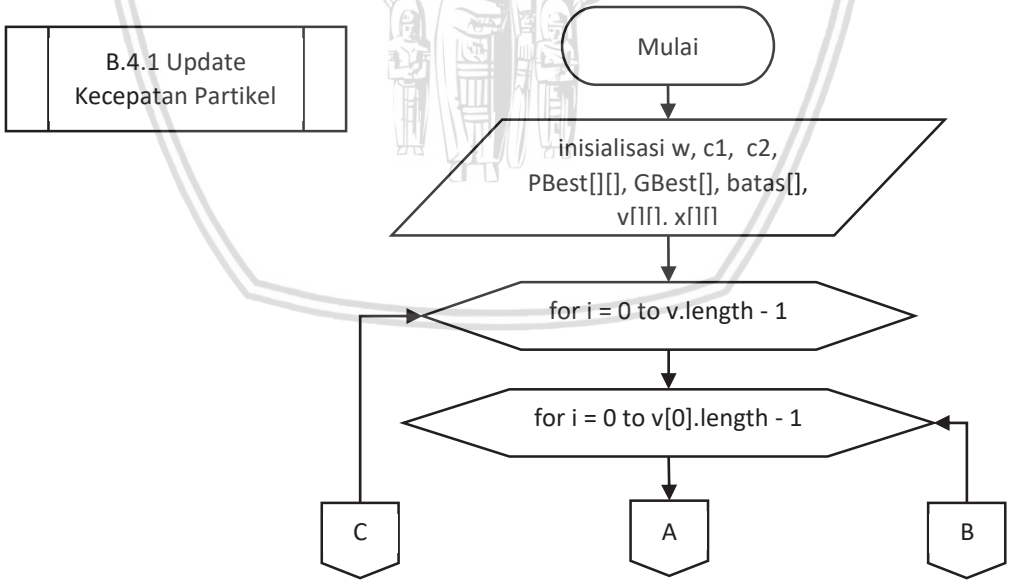
Pada tahapan ini kecepatan dan posisi setiap partikel diperbaharui secara iteratif, hingga iterasi maksimal dan menemukan partikel yang mendekati optimal. Pada PSO, posisi partikel memiliki batasan tersendiri tergantung objek optimasinya, pembatasan posisi partikel ini dimaksudkan agar dalam proses pencarian solusi terdesentralisasi dan tidak keluar dari ruang pencarian solusi. Kecepatan partikel juga memiliki *threshold* (batasan), batasan kecepatan partikel tergantung batasan posisi partikelnya. Berikut diagram alir update kecepatan dan posisi partikel ditunjukkan oleh Gambar 4.34

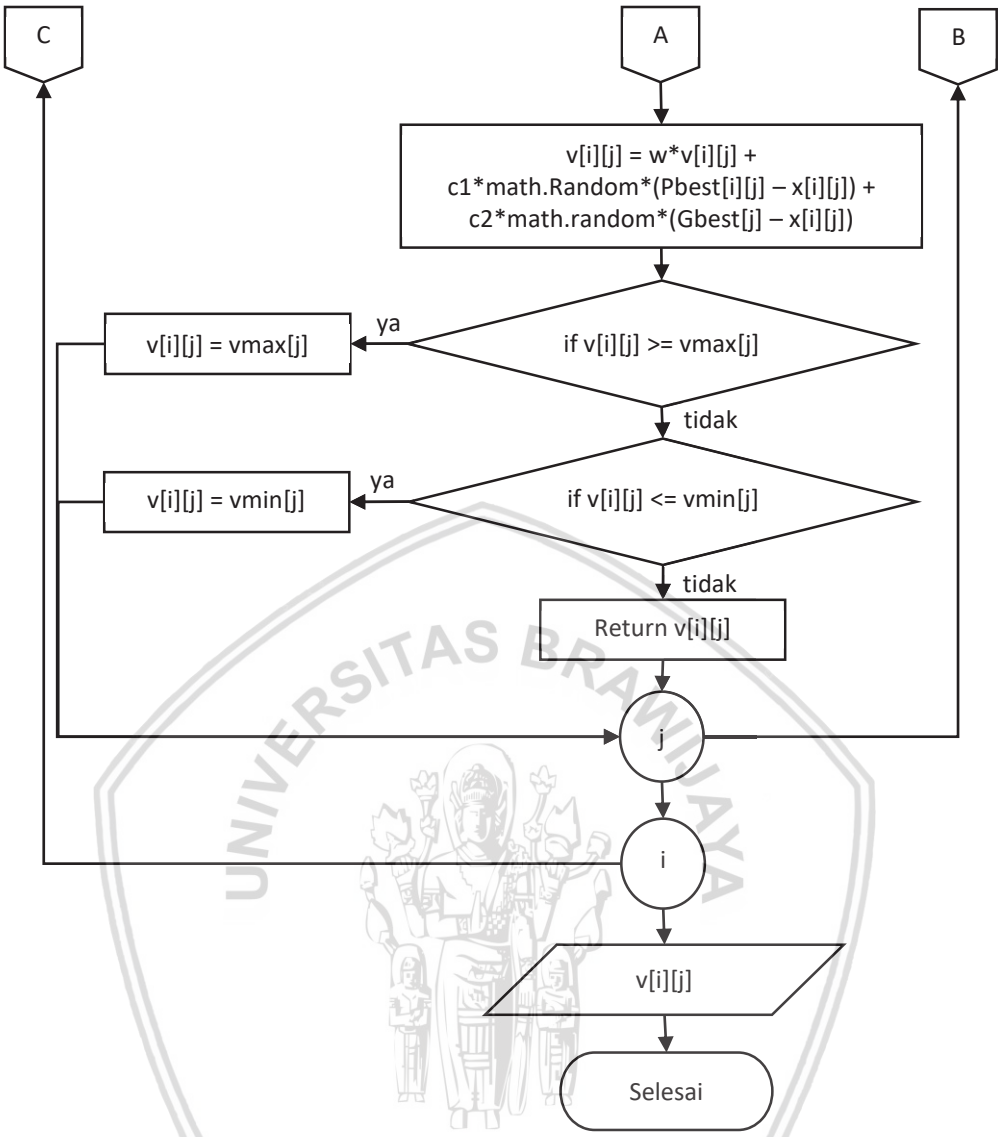


Gambar 4.34 Diagram alir update kecepatan dan posisi partikel

B.4.1 Update Kecepatan Partikel

Update kecepatan dilakukan pada tiap dimensi di setiap partikel dengan menggunakan Persamaan (2.42). Berikut diagram alir menghitung kecepatan partikel PSO yang ditunjukkan oleh Gambar 4.35.





Gambar 4.35 Diagram alir algoritme update kecepatan partikel

Penjabaran diagram alir update kecepatan partikel dapat dilihat pada pseudocode Gambar 4.36 berikut.

```
Nama Algoritme : Update Kecepatan Partikel
Deskripsi
  1. Input : w, c1,  c2, PBest[][], GBest[], vmax[],vmin[]
     v[][], x[][]
  2. Proses :
     • v[i][j] = w*v[i][j] + c1*math.Random*(Pbest[i][j] –
       x[i][j]) + c2*math.random*(Gbest[j] – x[i][j])
     • if (v[i][j] >= vmax[j]) → v[i][j] = vmax
     • else if (v[i][j] <= vmin[j]→ v[i][j]= vmin
     • else return v[i][j]
  3. Output : v[][]
```

Gambar 4.36 Pseudocode update kecepatan partikel

Berikut contoh manualisasi update kecepatan partikel ke – 1 dimensi λ pada iterasi ke – 1 dengan inisialisasi posisi awal Tabel 4.10 dan hasil update kecepatan semua partikel iterasi ke – 1 yang ditunjukkan oleh Tabel 4.32.

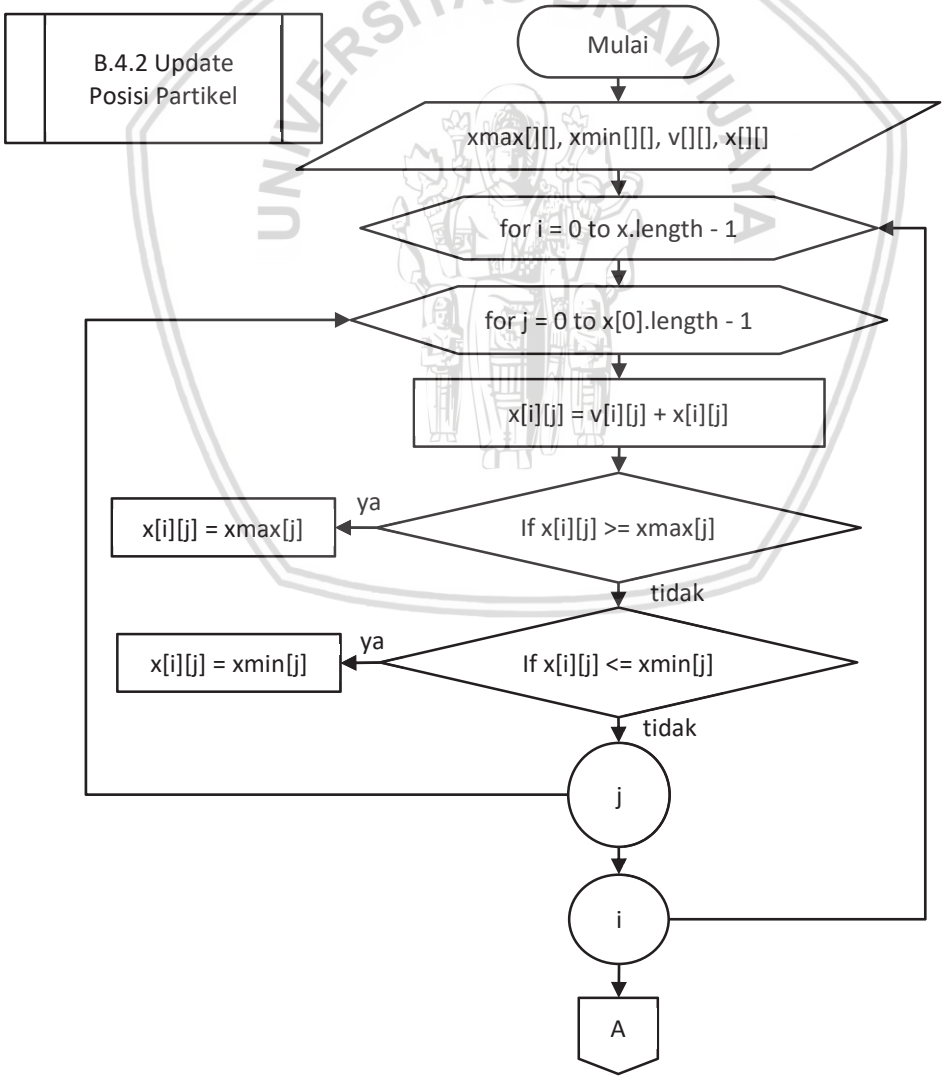
$$v_{1,\lambda}^{1} = (0{,}5 * 0) + (1 * 0{,}85)(0{,}691401 - 0{,}691401) + (1 * 0{,}35)(0{,}772748 - 0{,}691401) = 0{,}028471$$

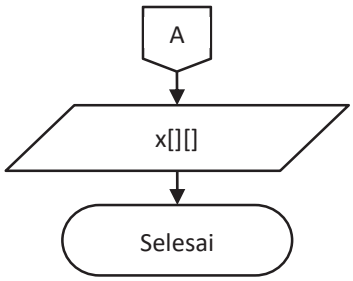
Tabel 4.32 Hasil update kecepatan iterasi ke – 1

v(1)	λ	C	ε	cLR
v_1	0,028471	-344,94	0,000824	-0,17211
v_2	0,227298	-55,0898	0,001732	0,067027
v_3	0	0	0	0

B.4.2 Update Posisi Partikel

Sama halnya dengan update kecepatan, update posisi partikel dilakukan pada tiap dimensi di setiap partikel dengan menggunakan Persamaan (2.43). Berikut diagram alir update posisi partikel PSO yang ditunjukkan oleh Gambar 4.37.





Gambar 4.37 Diagram alir algoritme update posisi partikel

Penjabaran diagram alir update posisi partikel dapat dilihat pada pseudocode Gambar 4.38 berikut.

```
Nama Algoritme : Update Posisi Partikel
Deskripsi
  1. Input : xmin[], xmax[], v[][], x[][]
  2. Proses :
      • x[i][j] = x[i][j] + v[i][j]
      • if (x[i][j] >= xmax[j]) → x[i][j] = xmax
      • else if (x[i][j] <= xmin[j]→ x[i][j] = xmin
      • else return x[i][j]
  3. Output : x][]
```

Gambar 4.38 Pseudocode update posisi partikel

Berikut contoh manualisasi update kecepatan partikel ke – 1 dimensi λ pada iterasi ke – 1 dengan inisialisasi posisi awal Tabel 4.10 dan hasil update posisi semua partikel iterasi ke – 1 yang ditunjukkan oleh Tabel 4.33.

$$x_{1,\lambda}^{1} = 0{,}691400578 + 0{,}028471425 = 0{,}719872004$$

Tabel 4.33 Hasil update posisi partikel iterasi ke – 1

x(1)	λ	C	ε	cLR
x_1	0,719872	225,5704	0,006813	0,42039
x_2	0,767723	81,48577	0,006751	0,320926
x_3	0,772748	74,43793	0,007109	0,390051

PSO Iterasi Ke – i

Pada iterasi-iterasi selanjutnya lakukan pelatihan SVR untuk setiap partikel yang telah di-*update*. Iterasi PSO dilakukan sampai iterasi maksimal yang telah ditentukan sebelumnya. Hasil akhir dari proses PSO ini berupa partikel dengan nilai *fitness* terbaik (GBest). Berikut hasil evaluasi PBest Iterasi Ke – 1 yang ditunjukkan oleh Tabel 4.34, hasil evaluasi PBest Iterasi Ke – 5 yang ditunjukkan oleh Tabel 4.35, dan hasil evaluasi PBest Iterasi Ke – 10 yang ditunjukkan oleh Tabel 4.36.

Tabel 4.34 Hasil evaluasi PBest PSO iterasi ke - 1

Partikel	PBEST				*Fitness*
	λ	C	ε	cLR	
1	0,719872	225,5704	0,006813	0,42039	0,978661
2	0,540425	136,5756	0,005018	0,2539	0,965301

3	0,772748	74,43793	0,007109	0,390051	0,971424
GBest	**0,719872**	**225,5704**	**0,006813**	**0,42039**	**0,978661**

Tabel 4.35 Hasil evaluasi PBest PSO iterasi ke – 5

Partikel	PBEST				*Fitness*
	λ	C	ε	cLR	
1	0,719872	225,5704	0,006813	0,42039	0,978661
2	0,540425	136,5756	0,005018	0,2539	0,965301
3	0,772748	74,43793	0,007109	0,390051	0,971424
GBest	**0,719872**	**225,5704**	**0,006813**	**0,42039**	**0,978661**

Tabel 4.36 Hasil evaluasi PSO iterasi maksimal (10 iterasi)

Partikel	PBEST				*Fitness*
	λ	C	ε	cLR	
1	0,719872	225,5704	0,006813	0,42039	0,978661
2	0,540425	136,5756	0,005018	0,2539	0,965301
3	0,772748	74,43793	0,007109	0,390051	0,971424
GBest	**0,719872**	**225,5704**	**0,006813**	**0,42039**	**0,978661**

Tabel 4.37 Parameter terbaik hasil optimasi PSO

GBEST				*Fitness*
Λ	C	ε	cLR	
0,719872	**225,5704**	**0,006813**	**0,42039**	**0,978661**

C. Pengujian SVR

Pada tahapan pengujian dengan SVR ini dilakukan untuk menguji parameter optimal hasil optimasi PSO. Parameter optimal ini adalah parameter dengan nilai *fitness* tertinggi dan nilai MAPE terendah pada saat pelatihan SVR. Parameter optimal diuji dengan data uji menggunakan metode SVR. Data yang digunakan untuk pengujian SVR sebanyak 3 data. Berikut data uji yang digunakan untuk pengujian SVR ditunjukkan oleh Tabel 4.38 dan data uji normalisasi yang ditunjukkan oleh Tabel 4.39

Tabel 4.38 Data uji

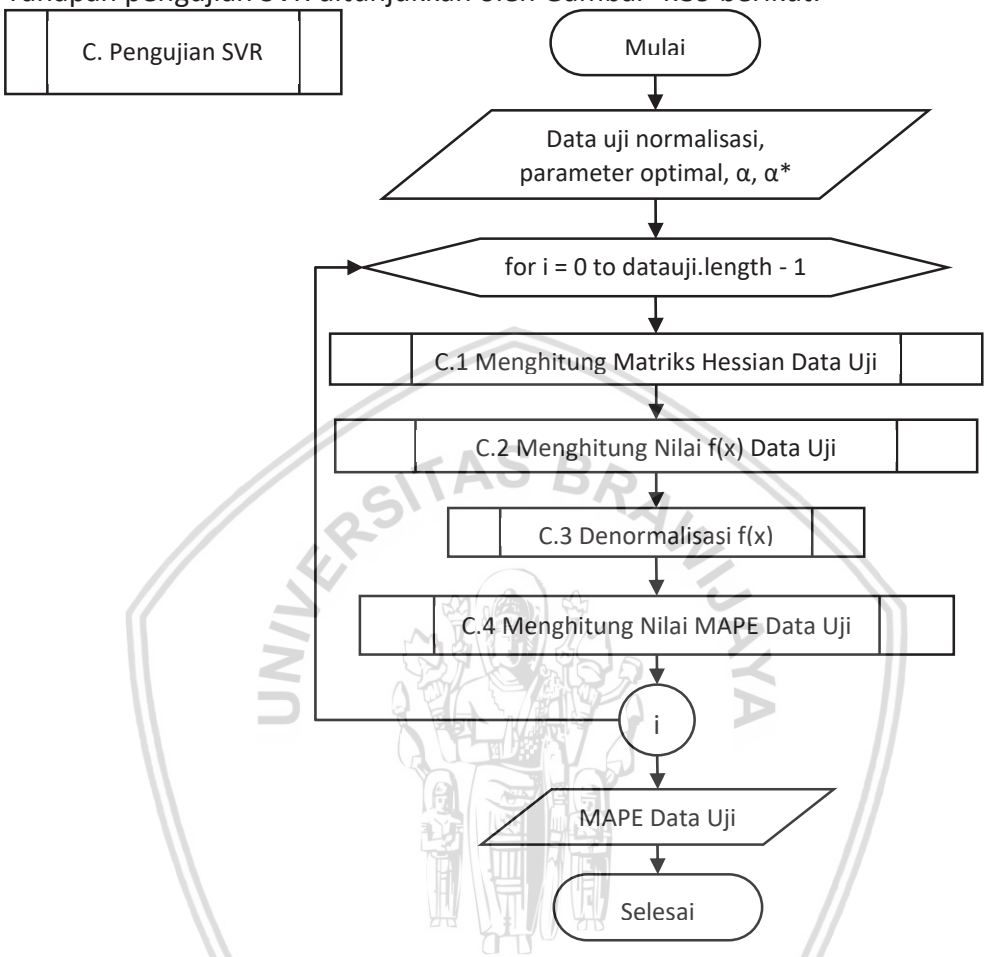
Data Ke -	X_1	X_2	X_3	X_4	Y
1	18193,33	18076,67	18126,67	18233,33	18516,67
2	18076,67	18126,67	18233,33	18516,67	18566,67
3	18126,67	18233,33	18516,67	18566,67	18550

Tabel 4.39 Data uji normalisasi

Data Ke -	X_1	X_2	X_3	X_4	Y
1	0,238095	0	0,1020408	0,319728	0,897959

2	0	0,102041	0,3197279	0,897959	1
3	0,102041	0,319728	0,8979592	1	0,965986

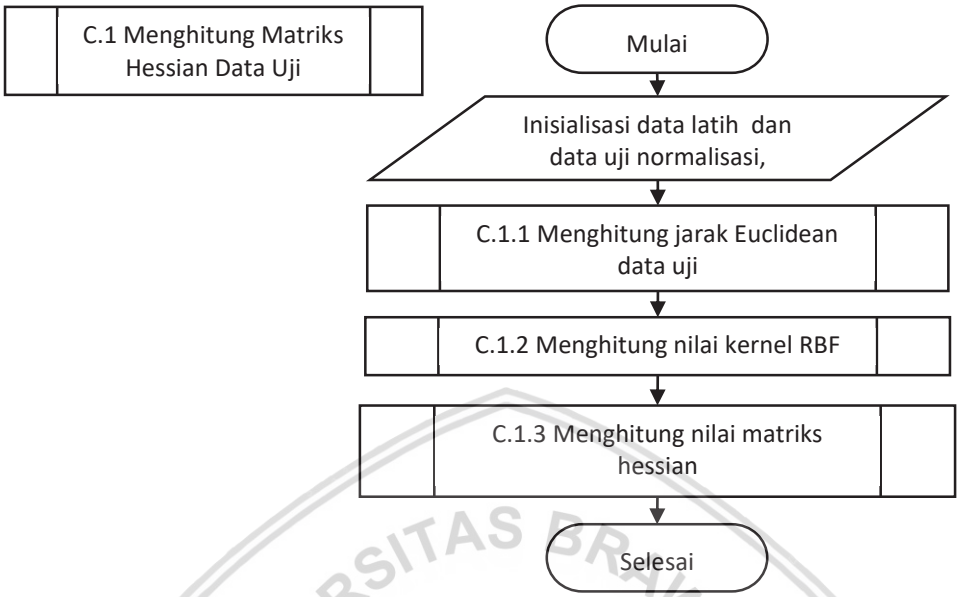
Tahapan pengujian SVR ditunjukkan oleh Gambar 4.39 berikut.



Gambar 4.39 Diagram alir pengujian SVR

C.1 Menghitung Matriks Hessian Data Uji

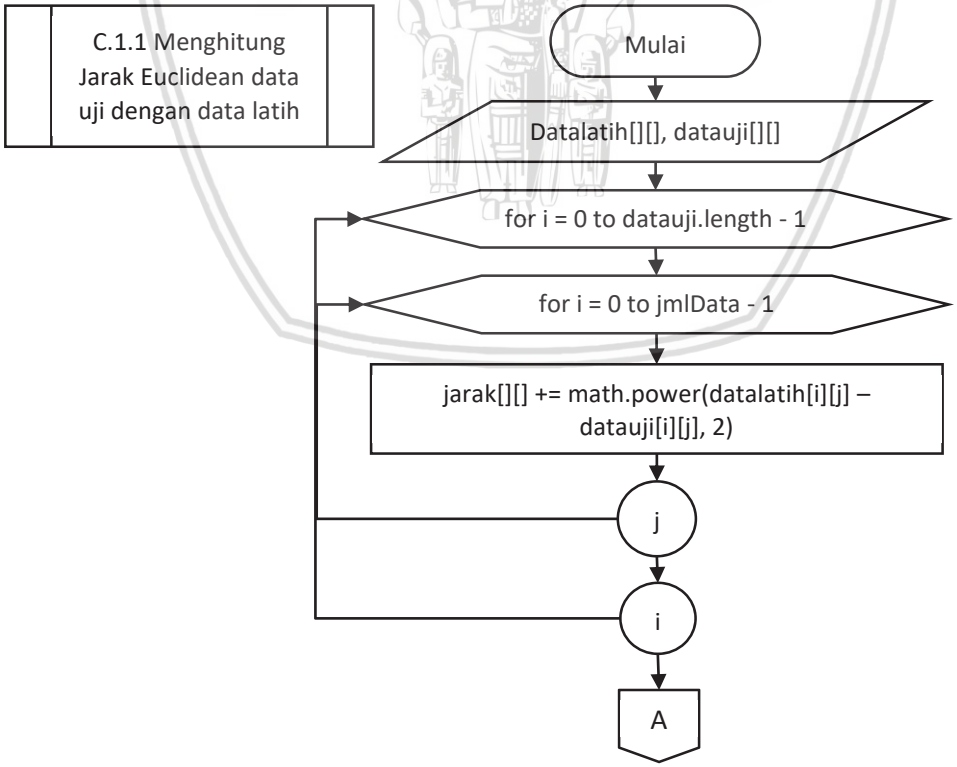
Diagram alir menghitung matriks hessian data uji ditunjukkan oleh Gambar 4.40.

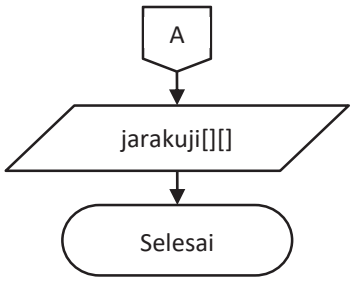


Gambar 4.40 Diagram alir menghitung matriks hessian data uji

C.1.1 Menghitung jarak data latih dengan data uji

Diagram alir menghitung jarak data uji dengan data latih ditunjukkan oleh Gambar 4.41 berikut.





Gambar 4.41 Diagram alir algoritme menghitung jarak euclidean data uji

Penjelasan dari diagram alir algoritme menghitung jarak Euclidean data uji diatas dijabarkan pada pseudocode Gambar 4.42.

```
Nama algoritme : menghitung jarak Euclidean data latih dengan
data uji
Deklarasi :
Double jarak[][], sum
Deskripsi :
 1. Input : datalatih[][], datauji[][]
 2. Proses :
      a. Perulangan i
      b. Perulangan j
      c. Menghitung jarakuji[i][j] +=
         Math.power(datalatih[i][j]-datauji[i][j],2)
      d. End loop j
      e. End loop i
 3. Output : jarakuji[][]
```

Gambar 4.42 Pseudocode menghitung jarak Euclidean data uji

Berikut contoh perhitungan jarak Euclidean data uji ke – 1 dengan data latih ke – 1 yang hasil perhitungannya ditunjukkan oleh Tabel 4.40

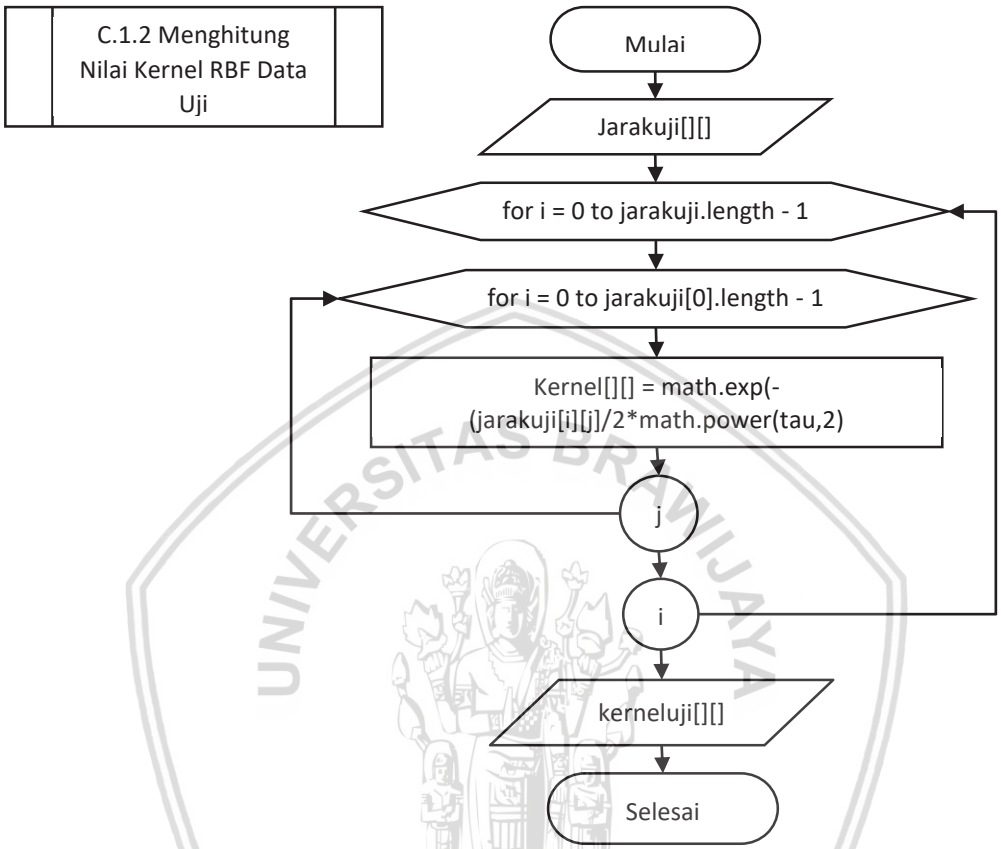
$$D_{1,1} = (0{,}14966 - 0{,}2381)^2 + (0{,}35374 - 0)^2 + (0{,}35374 - 0{,}10204)^2 + (0{,}35374 - 0{,}31973)^2 = 0{,}197464$$

Tabel 4.40 Hasil perhitungan jarak euclidean data uji

Data Uji	Data Latih						
	1	2	3	4	5	6	7
1	0,197464	0,304086	0,2122727	0,036929	0,072609	0,178583	0,119117
2	0,383081	0,995974	0,9794993	0,728724	0,458975	0,82225	0,783146
3	0,717247	1,360683	1,7394141	1,543153	1,18409	1,458975	1,623953

C.1.2 Menghitung nilai kernel RBF data uji

Diagram alir menghitung nilai kernel RBF data uji ditunjukkan oleh Gambar 4.43 berikut.



Gambar 4.43 Diagram alir algoritme menghitung nilai kernel RBF data uji

Penjelasan dari diagram alir algoritme menghitung nilai kernel RBF data uji diatas dijabarkan pada pseudocode Gambar 4.44.

```
Nama Algoritme : Menghitung Kernel RBF Data Uji
Deklarasi :
Double kernel[][]
Deskripsi
 1. Input : jarakuji[][]
 2. Proses :
      Kernel[i][j] = math.exp(-
      (jarak[i][j]/2*math.power(tau,2))
 3. Output : kerneluji[i][j]
```

Gambar 4.44 Pseudocode menghitung nilai kernel RBF data uji

Berikut contoh perhitungan nilai kernel RBF data uji ke – 1 dengan data latih ke – 1 yang hasil perhitungannya ditunjukkan oleh Tabel 4.41

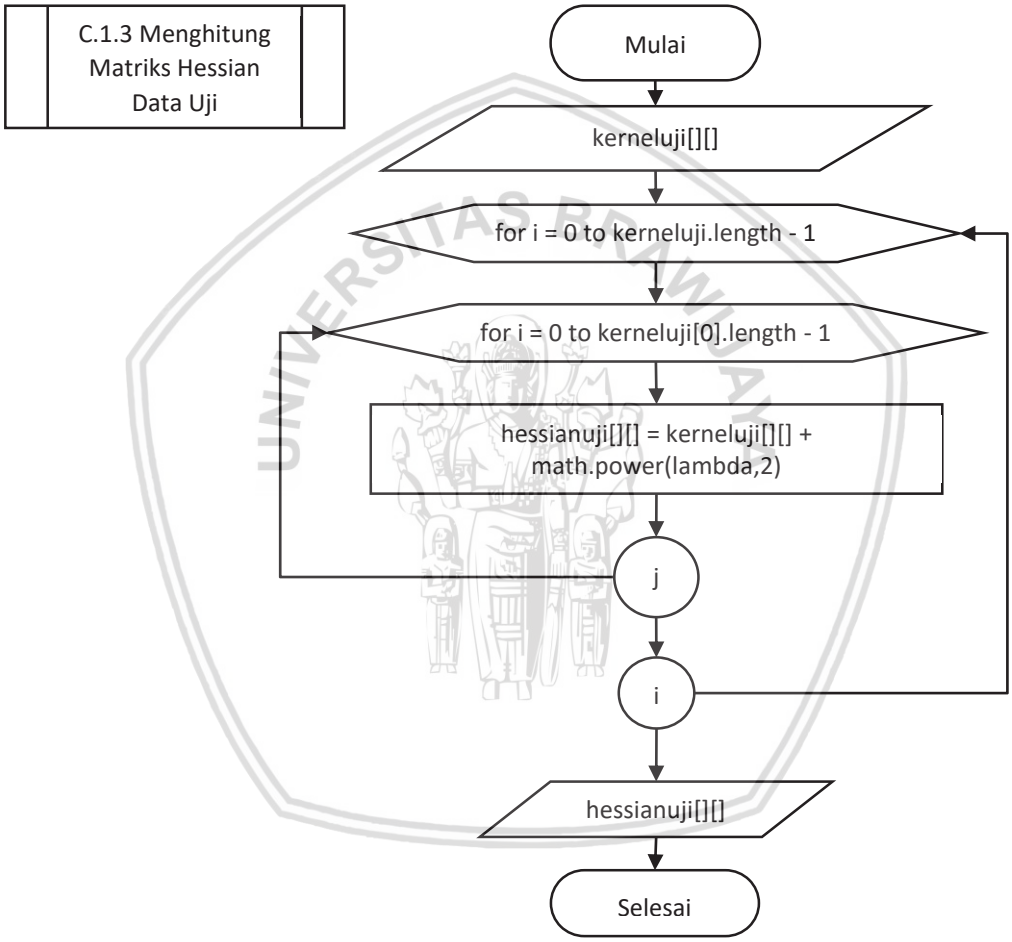
$$K_{1,1} = EXP\left(-\frac{0{,}19746}{(2 \times 0{,}1)^2}\right) = EXP\left(-\frac{0{,}19746}{0{,}04}\right) = 0{,}00718$$

Tabel 4.41 Hasil perhitungan kernel RBF data uji

Data Uji	Data Latih						
	1	2	3	4	5	6	7
1	0,007179	0,000499	0,004958	0,397235	0,162803	0,01151	0,050898
2	6,93E-05	1,54E-11	2,32E-11	1,22E-08	1,04E-05	1,18E-09	3,14E-09
3	1,63E-08	1,68E-15	1,3E-19	1,76E-17	1,39E-13	1,44E-16	2,33E-18

C.1.3 Menghitung matriks hessian data uji

Diagram alir menghitung nilai kernel RBF data uji ditunjukkan oleh Gambar 4.45 berikut.



Gambar 4.45 Diagram alir algoritme menghitung matriks hessian data uji

Penjelasan dari diagram alir algoritme menghitung nilai matriks hessian data uji diatas dijabarkan pada pseudocode Gambar 4.46.

```
Nama Algoritme : Menghitung Matriks Hessian Data Uji
Deklarasi :
Double hessianuji[][]
Deskripsi
 1. Input : kerneluji[][]
 2. Proses :
    Hessian[i][j] = kernel[i][j] + math.power(lambda,2)
 3. Output : hessian[i][j]
```

Gambar 4.46 Pseudocode menghitung matriks hessian data uji

Berikut contoh perhitungan matriks hessian data uji ke – 1 dengan data latih ke – 1 yang hasil perhitungannya ditunjukkan oleh Tabel 4.42

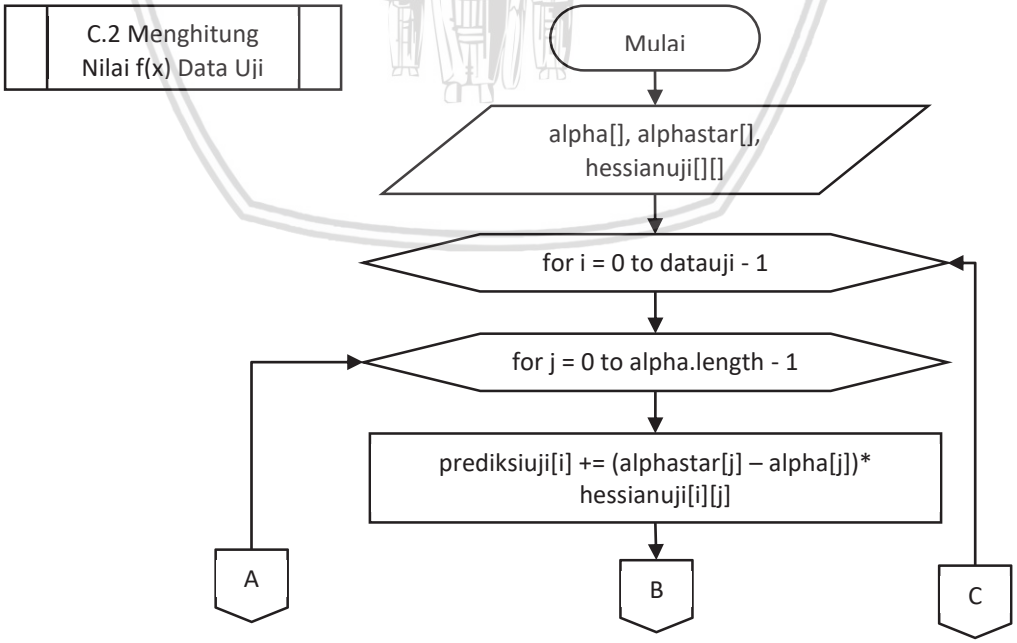
$$R_{1,1} = 0{,}007179 + 0{,}719872^2 = 0{,}525395$$

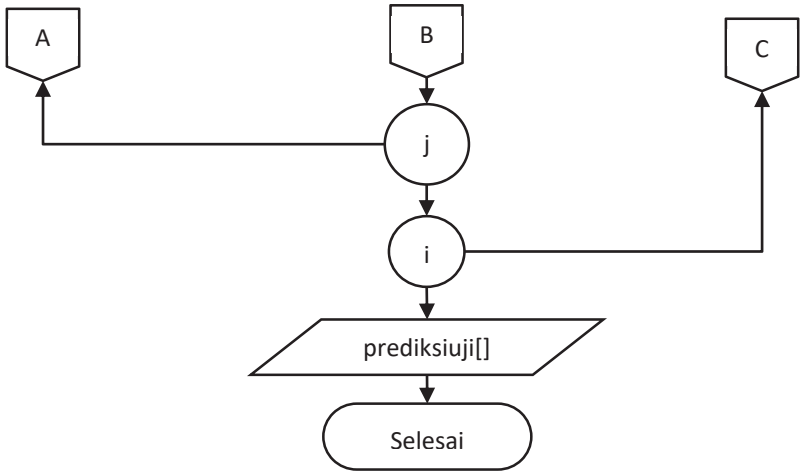
Tabel 4.42 Hasil perhitungan matriks hessian data uji

Data Uji	Data Latih						
	1	2	3	4	5	6	7
1	0,525395	0,518715	0,523173	0,915451	0,681018	0,529725	0,569114
2	0,518285	0,518216	0,518216	0,518216	0,518226	0,518216	0,518216
3	0,518216	0,518216	0,518216	0,518216	0,518216	0,518216	0,518216

C.2 Menghitung Nilai f(x) Data Uji

Tahapan pembentukan nilai regresi f(x) data uji ditunjukkan oleh Gambar 4.47 berikut.





Gambar 4.47 Diagram alir algoritme menghitung nilai f(x) data uji

Penjelasan dari diagram alir algoritme menghitung nilai f(x) data uji diatas dijabarkan pada pseudocode Gambar 4.48.

```
Nama Algoritme : Menghitung Nilai f(x) data uji
Deklarasi :
Double prediksiuji[]
Deskripsi
  1. Input : alpha[], alphastar[], hessianuji[][]
  2. Proses :
     Prediksi[] = (alphastar[] - alpha[])*hessianuji[][]
  3. Output : prediksiuji[]
```

Gambar 4.48 Pseudocode menghitung nilai f(x) data uji

Berikut contoh perhitungan nilai f(x) data uji ke − 1 menggunakan Persamaan (2.40) dengan hasil penghitungan matriks hessian data uji pada Tabel 4.41 dan perolehan nilai α dan α* hasil pelatihan SVR pada Tabel 4.23 beserta tabel hasil perhitungan nilai f(x) seluruh data uji yang ditunjukkan oleh Tabel 4.43.

$$f(x_1) = (0 - 0{,}0719872) * 0{,}525395 + (0 - 0{,}016493) * 0{,}518715 + (0{,}016662 - 0{,}00215) * 0{,}523173 + (0{,}124252 - 0) * 0{,}915451 + (0 - 0{,}089688) * 0{,}681018 + (0 - 0{,}0067) * 0{,}529725 + (0{,}218212 - 0) * 0{,}569114$$

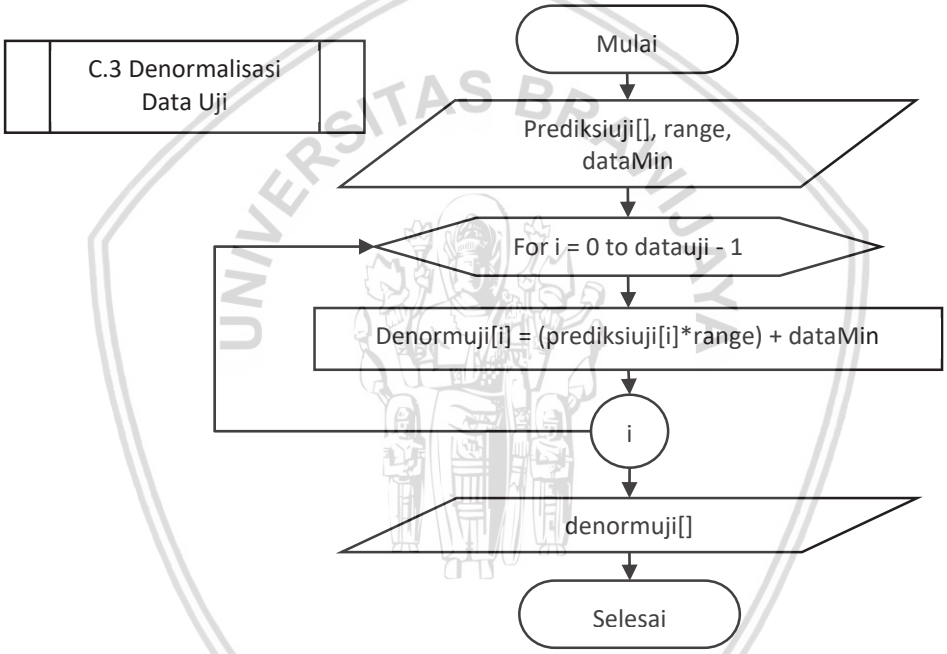
$$f(x_1) = (-0{,}0719872 * 0{,}525395) + (-0{,}016493 * 0{,}518715) + (0{,}014512 * 0{,}523173) + (0{,}124252 * 0{,}915451) + (-0{,}08969 * 0{,}681018) + (-0{,}0067 * 1{,}529725) + (0{,}218212 * 1{,}569114) = 0{,}13161$$

Tabel 4.43 Hasil perhitungan nilai f(x) data uji

Data Uji	F(x)	Y (data aktual)
1	0,13161	0,57393
2	0,086311	0,468872
3	0,086318	0,249027

C.3 Denormalisasi f(x) Data Uji

Proses denormalisasi ini dilakukan untuk membandingkan hasil prediksi data uji dengan data aktual. Diagram alir denormalisasi data uji ditunjukkan oleh Gambar 4.49 berikut.



Gambar 4.49 Diagram alir algoritme denormalisasi data uji

Penjelasan dari diagram alir algoritme menghitung nilai f(x) data uji diatas dijabarkan pada pseudocode Gambar 4.50.

```
Nama Algoritme : Denormalisasi data uji
Deklarasi :
Double denormuji[]
Deskripsi
  1. Input : prediksiuji[], range, dataMin
  2. Proses :
     denormuji[] = (prediksiuji[]*range)+dataMin
  3. Output : denormuji[]
```

Gambar 4.50 Pseudocode denormalisasi data uji

Berikut contoh perhitungan denormalisasi nilai f(x) pada Tabel 4.45 data uji ke – 1 dengan inisialisasi range, data minimal, dan data maksimal pada Tabel 4.6 beserta tabel hasil perhitungan nilai f(x) seluruh data uji yang ditunjukkan oleh Tabel 4.44.

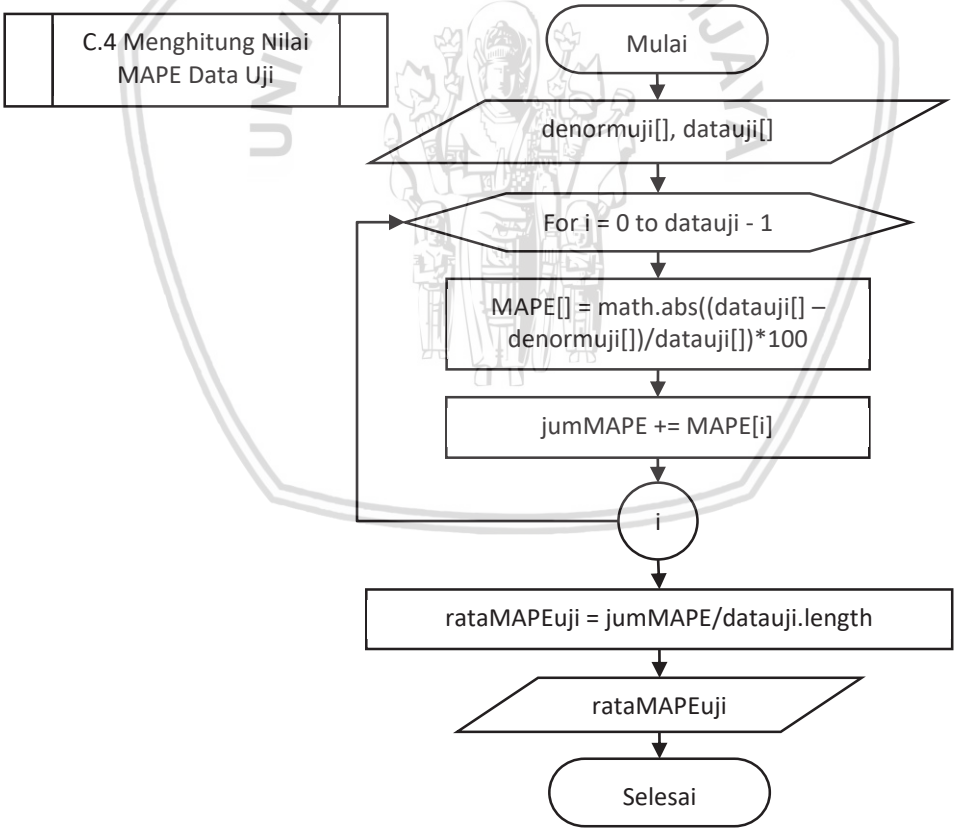
$$Denormalisasi\ f(x_1) = \frac{0{,}13161 * 490}{18076{,}67} = 18141{,}156$$

Tabel 4.44 Hasil denormalisasi data uji

Data Uji	Denormalisasi f(x)	Y denormalisasi
1	18141,156	18516,67
2	18118,959	18566,67
3	18118,962	18550

C.4 Menghitung Nilai MAPE Data Uji

Perhitungan nilai MAPE dilakukan untuk mengetahui persen rata-rata nilai error hasil prediksi. Berikut diagram alir menghitung nilai MAPE data uji yang ditunjukkan oleh Gambar 4.51



Gambar 4.51 Diagram alir algoritme menghitung nilai MAPE data uji

Penjelasan dari diagram alir algoritme menghitung nilai MAPE data uji diatas dijabarkan pada pseudocode Gambar 4.52.

```
Nama Algoritme : Menghitung Nilai MAPE data uji
Deklarasi :
Double MAPE[], rataMAPE, jumMAPE
Deskripsi
  1. Input : denormuji[], datauji[]
  2. Proses :
     MAPE[] = math.abs((denormuji[]-datauji[])/datauji[])*100
     jumMAPE += MAPE[]
     rataMAPE = jumMAPE/datauji.length
  3. Output : rataMAPE
```

Gambar 4.52 Pseudocode menghitung nilai MAPE data uji

Berikut contoh perhitungan nilai MAPE data uji ke – 1 dengan inisialisasi data denormalisasi Tabel 4.44 beserta tabel hasil perhitungan Nilai MAPE data latih yang ditunjukkan oleh Tabel 4.45.

$$MAPE(1) = \left|\frac{18516{,}67 - 18141{,}156}{18516{,}67}\right| \times 100\% = 2{,}027961$$

Tabel 4.45 Hasil Perhitungan Nilai MAPE Data Uji

Data Uji	MAPE %
1	2,027962
2	2,41135
3	2,323653
Rata-Rata	**2,254322**

4.2 Perancangan pengujian

Untuk mengetahui parameter terbaik dari metode SVR maupun metode PSO, maka perlu dilakukan suatu skenario pengujian. Berikut daftar perancangan skenario pengujian yang akan dilakukan, antara lain:

1. Pengujian perbandingan hasil peramalan
2. Pengujian iterasi pelatihan SVR
3. Pengujian batas nilai parameter SVR
4. Pengujian SVR-PSO
5. Pengujian iterasi PSO
6. Pengujian jumlah partikel PSO
7. Pengujian normalisasi

4.2.1 Perancangan skenario pengujian perbandingan hasil peramalan

Pengujian SVR dilakukan untuk membandingkan hasil peramalan metode SVR dan metode SVR-PSO. Rancangan pengujian SVR ditunjukkan oleh Tabel 4.46 berikut.

Tabel 4.46 Rancangan pengujian perbandingan hasil peramalan

Tanggal	Nilai Target	SVR		SVR-PSO	
		Hasil Peramalan	MAPE	Hasil Peramalan	MAPE

4.2.2 Perancangan skenario pengujian iterasi pelatihan SVR

Pengujian iterasi pelatihan SVR dilakukan untuk menentukan jumlah iterasi pelatihan SVR yang optimal. Pengujian ini dilakukan dengan parameter awal SVR selalu konstan. Rancangan pengujian iterasi pelatihan SVR ditunjukkan oleh Tabel 4.47 berikut.

Tabel 4.47 Rancangan pengujian iterasi pelatihan SVR

Jumlah Iterasi	MAPE Percobaan ke -					Rata-rata MAPE
	1	2	3	4	5	
10						
50						
100						
500						
1000						
5000						

4.2.3 Perancangan skenario pengujian batas nilai parameter SVR

Pengujian batas nilai parameter SVR yang akan di uji coba adalah parameter λ, C, ε, cLR dan σ. Pengujian ini dilakukan untuk menentukan batas ruang pencarian dimensi yang optimal sehingga menghasilkan kombinasi parameter SVR yang optimal.

4.2.3.1 Rancangan pengujian batas nilai parameter λ

Rancangan pengujian batas nilai parameter λ ditunjukkan oleh Tabel 4.48 berikut.

Tabel 4.48 Rancangan pengujian batas nilai parameter λ

Batas Min	Batas Maks	MAPE Percobaan ke -					Rata-rata MAPE
		1	2	3	4	5	
0,0001	0,001						
	0,01						
	0,1						
0,001	0,01						
	0,1						

	1						
0,01	0,1						
	1						
	10						

4.2.3.2 Rancangan pengujian batas nilai parameter C

Rancangan pengujian batas nilai parameter C ditunjukkan oleh Tabel 4.49 berikut.

Tabel 4.49 Rancangan pengujian batas nilai parameter C

Batas Min	Batas Maks	MAPE Percobaan ke -					Rata-rata MAPE
		1	2	3	4	5	
1	10						
	100						
	1000						
10	100						
	1000						
100	1000						

4.2.3.3 Rancangan pengujian batas nilai parameter ε

Rancangan pengujian batas nilai parameter ε ditunjukkan oleh Tabel 4.50 berikut.

Tabel 4.50 Rancangan pengujian batas nilai parameter ε

Batas Min	Batas Maks	MAPE Percobaan ke -					Rata-rata MAPE
		1	2	3	4	5	
0,0001	0,001						
	0,01						
	0,1						
0,001	0,01						
	0,1						
	1						
0,01	0,1						
	1						
	10						

4.2.4 Perancangan skenario pengujian SVR – PSO

Pengujian SVR – PSO dilakukan untuk membandingkan hasil prediksi metode SVR setelah dioptimasi dengan sebelum dioptimasi dengan metode PSO.. Rancangan pengujian SVR ditunjukkan oleh Tabel 4.51 berikut.

Tabel 4.51 Rancangan skenario pengujian SVR – PSO

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

4.2.5 Perancangan skenario pengujian iterasi PSO

Pengujian iterasi PSO dilakukan untuk menentukan jumlah iterasi PSO dimana sistem mencapai konvergensi. Rancangan pengujian iterasi PSO ditunjukkan oleh Tabel 4.52 berikut.

Tabel 4.52 Rancangan pengujian iterasi PSO

Jumlah Iterasi	Percobaan ke -					*Fitness*	Rata-rata MAPE
	1	2	3	4	5		
10							
50							
100							
500							

4.2.6 Perancangan skenario pengujian jumlah partikel PSO

Pengujian jumlah partikel PSO dilakaukan untuk menentukan jumlah partikel PSO optimal yang memiliki nilai *fitness* tertinggi. Rancangan pengujian jumlah partikel PSO ditunjukkan oleh Tabel 4.53 berikut.

Tabel 4.53 Rancangan pengujian jumlah partikel PSO

Jumlah Partikel	Percobaan ke -					*Fitness*	Rata-rata MAPE
	1	2	3	4	5		
3							
5							
10							

50							
100							

4.2.7 Perancangan skenario pengujian normalisasi

Pengujian normalisasi dilakukan untuk membandingkan hasil peramalan metode SVR-PSO untuk data yang dinormalisasi dan tanpa dinormalisasi. Rancangan pengujian normalisasi ditunjukkan oleh Tabel 4.54 berikut.

Tabel 4.54 Rancangan pengujian normalisasi

Tanggal	Nilai Target	Normalisasi		Tanpa Normalisasi	
		Hasil Peramalan	MAPE	Hasil Peramalan	MAPE

BAB 5 HASIL

Bab hasil membahas mengenai hasil implementasi dari sistem yang telah dirancang pada tahapan perancangan. Bab ini akan dijabarkan dalam sub bab yang terdiri dari implementasi sistem dan implementasi antarmuka.

5.1 Implementasi sistem

Implementasi sistem akan menjelaskan kode program dari sistem peramalan dengan metode SVR-PSO. Implementasi sistem peramalan ini terdiri dari implementasi proses normalisasi, implementasi PSO, implementasi pelatihan SVR, dan implementasi pengujian SVR

5.1.1 Implementasi proses normalisasi

Implementasi proses normalisasi berisi implementasi dalam kode program dari proses normalisasi data. Kode program dari proses normalisasi dapat dilihat pada Gambar 5.1 berikut ini.

```
1   public double[][] normalisasi(double data[][]) {
2           double norm[][] = new
3   double[data.length][data[0].length];
4           this.min = getMin(datain);
5           this.max = getMax(datain);
6           this.range = max - min;
7
8           for (int i = 0; i < norm.length; i++) {
9               for (int j = 0; j < norm[0].length; j++) {
10                  norm[i][j] = (data[i][j] - min) / range;
11              }
12          }
13
14          return norm;
15      }
```

Gambar 5.1 Kode program normalisasi

Berikut penjelasan dari kode program normalisasi yang ditunjukkan pada Gambar 5.1.

Baris 1 : Deklarasi method normalisasi.
Baris 2-3 : Deklarasi variabel norm dengan tipe data double array dua dimensi. Variabel ini digunakan untuk menyimpan matriks hasil normalisasi.
Baris 4 : Menentukan nilai minimum dari datain.
Baris 5 : Menentukan nilai maksimum dari datain.
Baris 6 : Menghitung nilai range.
Baris 8-12 : Menghitung normalisasi dataset datain.
Baris 14 : Method mengembalikan nilai ke variabel norm.

5.1.2 Implementasi PSO

Implementasi PSO merupakan implementasi kode program dari metode PSO. Pada metode PSO terdapat beberapa tahapan yaitu inisialisasi, *update* partikel, *update* pBest, dan *update* gBest.

5.1.2.1 Implementasi proses inisialisasi partikel awal PSO

Kode program dari proses inisialisasi partikel awal PSO dapat dilihat pada Gambar 5.2 berikut ini.

```
1     public void inisialisasi() {
2         for (int i = 0; i < v.length; i++) {
3             for (int j = 0; j < v[0].length; j++) {
4                 v[i][j] = 0;
5             }
6         }
7         for (int i = 0; i < gbest.length; i++) {
8             gbest[i] = 0;
9         }
10
11        double range[] = new double[x[0].length];
12        for (int i = 0; i < x.length; i++) {
13            for (int j = 0; j < 4; j++) {
14                range[j] = batasX[1][j] - batasX[0][j];
15                x[i][j] = batasX[0][j] + (Math.random() *
16 range[j]);
17                pbest[i][j] = x[i][j];
18
19            }
20            x[i][4] = 1 / (1 + hitungFitness(x[i][0],
21 x[i][1], x[i][2], x[i][3]));
22            pbest[i][4] = x[i][4];
23        }
24    }
```

Gambar 5.2 Kode program inisialisasi partikel awal PSO

Berikut penjelasan dari kode program proses inisialisasi partikel awal PSO yang ditunjukkan pada Gambar 5.2.

Baris 1 : Deklarasi method inisialisasi.
Baris 2-6 : Inisialisasi kecepatan tiap partikel sama dengan 0.
Baris 7-9 : Inisialisasi Gbest sama dengan 0.
Baris 11 : Deklarasi variabel range untuk menyimpan nilai range batas tiap dimensi.
Baris 12-19 : Inisialisasi partikel awal dan Pbest awal.
Baris 20-21 : Menghitung *fitness* tiap partikel.
Baris 22 : Inisialisasi *fitness* PBest awal.

5.1.2.2 Implementasi proses update partikel PSO

Kode program dari proses *update* kecepatan partikel dapat dilihat pada Gambar 5.3 berikut ini.

```
1     public void updateV() {
2         for (int i = 0; i < v.length; i++) {
3             for (int j = 0; j < v[0].length; j++) {
4                 v[i][j] = (w * v[i][j]) + (c1 *
5 Math.random() * (pbest[i][j] - x[i][j])) + (c2 *
6 Math.random() * (gbest[j] - x[i][j]));
7                 if (v[i][j] < batasV[0][j]) {
8                     v[i][j] = batasV[0][j];
9                 } else if (v[i][j] > batasV[1][j]) {
10                    v[i][j] = batasV[1][j];
11                }
```

```
12              }
13          }
14      }
```

Gambar 5.3 Kode program update kecepatan partikel

Berikut penjelasan dari kode program proses update kecepatan partikel yang ditunjukkan pada Gambar 5.3.

Baris 1 : Deklarasi method update kecepatan.
Baris 2-6 : Menghitung kecepatan baru setiap partikel
Baris 7-10 : Memastikan kecepatan baru tidak melebihi batas kecepatan.

Kode program dari proses *update* posisi partikel dapat dilihat pada Gambar 5. 4 berikut ini.

```
1       public void updateX() {
2           for (int i = 0; i < v.length; i++) {
3               for (int j = 0; j < v[0].length; j++) {
4                   x[i][j] = x[i][j] + v[i][j];
5                   if (x[i][j] < batasX[0][j]) {
6                       x[i][j] = batasX[0][j];
7                   } else if (x[i][j] > batasX[1][j]) {
8                       x[i][j] = batasX[1][j];
9                   }
10              }
11              x[i][4] = 1 / (1 + hitungFitness(x[i][0],
12  x[i][1], x[i][2], x[i][3]));
13          }
14      }
```

Gambar 5.4 Kode program update posisi partikel

Berikut penjelasan dari kode program proses update posisi partikel yang ditunjukkan pada Gambar 5.4.

Baris 1 : Deklarasi method update posisi partikel.
Baris 2-4 : Menghitung posisi baru setiap partikel.
Baris 5-10 : Memastikan posisi baru setiap partikel tidak melebihi batas dimensi yang ditentukan.
Baris 11-12 : Menghitung *fitness* posisi partikel baru.

5.1.2.3 Implementasi proses update PBest

Kode program dari proses *update* PBest dapat dilihat pada Gambar 5.5 berikut ini.

```
1   public void updatePBest() {
2           for (int i = 0; i < pbest.length; i++) {
3               if (x[i][4] > pbest[i][4]) {
4                   for (int j = 0; j < x[0].length; j++) {
5                       pbest[i][j] = x[i][j];
6                   }
7               } else {
8                   for (int j = 0; j < x[0].length; j++) {
9                       pbest[i][j] = pbest[i][j];
10                  }
11              }
12          }
```

Gambar 5.5 Kode program proses update PBest

Berikut penjelasan dari kode program proses *update* PBest yang ditunjukkan pada Gambar 5.5.

Baris 1 : Deklarasi method normalisasi.
Baris 3-11 : Inisialisasi PBest dengan membandingkan *fitness* baru dengan PBest

5.1.2.4 Implementasi proses update GBest

Kode program dari proses *update* GBest dapat dilihat pada Gambar 5.6 berikut ini.

```
1     public int indexGBest() {
2         double temp[][] = new double[pbest.length][2];
3         for (int i = 0; i < pbest.length; i++) {
4             temp[i][0] = pbest[i][4];
5             temp[i][1] = i;
6         }
7
8         java.util.Arrays.sort(temp, new
9   java.util.Comparator<double[]>() {
10            public int compare(double[] a, double[] b) {
11                return Double.compare(a[0], b[0]);
12            }
13        });
14
15        return (int) temp[temp.length - 1][1];
16    }
17
18    public void updateGBest() {
19        for (int i = 0; i < gbest.length; i++) {
20            gbest[i] = pbest[indexGBest()][i];
21        }
```

Gambar 5.6 Kode program proses update GBest

Berikut penjelasan dari kode program normalisasi yang ditunjukkan pada Gambar 5.6.

Baris 1 : Deklarasi method indexGBest.
Baris 2 : Deklarasi variabel temp untuk menyimpan nilai *fitness* PBest dan indeks partikel.
Baris 3-6 : Inisialisasi variabel temp.
Baris 8-13 : Mengurutkan nilai *fitness* PBest.
Baris 15 : Mengembalikan nilai pada indeks partikel dengan *fitness* tertinggi
Baris 18 : Deklarasi method updateGBest.
Baris 19-21 : Inisialisasi GBest baru.

5.1.3 Implementasi SVR

Implementasi SVR merupakan implementasi kode program dari metode SVR. Pada metode SVR terdapat beberapa tahapan yaitu *sequential learning*, penghitungan nilai regresi, denormalisasi, penghitungan nilai MAPE, dan pengujian SVR.

5.1.3.1 Implementasi proses sequential learning

Kode program dari proses *sequential learning* dapat dilihat pada Gambar 5.7 berikut ini.

```
1     public void sequentialLearning(double cLR, double
2   epsilon, double C) {
3           getDistance(norm);
4           getHessianLatih();
5
6           double[] E = new double[hessianLatih.length];
7           double[] deltaAstar = new double[E.length];
8           double[] deltaA = new double[E.length];
9
10          gamma = cLR / getMax(hessianLatih);
11
12          for (int iter = 0; iter < iterasi; iter++) {
13              for (int i = 0; i < hessianLatih.length; i++) {
14                  double jumlahRij = 0;
15                  for (int j = 0; j < hessianLatih[0].length;
16  j++) {
17                      jumlahRij += (alfastar[j] - alfa[j]) *
18  hessianLatih[i][j];
19                  }
20                  E[i] = norm_train[i][4] - jumlahRij;
21                  deltaAstar[i] = Math.min(Math.max(gamma *
22  (E[i] - epsilon), -alfastar[i]), C - alfastar[i]);
23                  deltaA[i] = Math.min(Math.max(gamma * (-
24  E[i] - epsilon), -alfa[i]), C - alfa[i]);
25              }
26              for (int i = 0; i < hessianLatih.length; i++) {
27                  alfastar[i] = deltaAstar[i] + alfastar[i];
28                  alfa[i] = deltaA[i] + alfa[i];
29              }
30          }
31      }
```

Gambar 5.7 Kode program proses sequential learning

Berikut penjelasan dari kode program normalisasi yang ditunjukkan pada Gambar 5.7.

Baris 1-2 : Deklarasi method sequential learning.
Baris 3 : Menghitung jarak Euclidean variabel norm.
Baris 4 : Menghitung matriks hessian data latih.
Baris 6 : Deklarasi variabel E.
Baris 7 : Deklarasi variabel deltaAstar.
Baris 8 : Deklarasi variabel delta.
Baris 12-20 : Menghitung nilai E.
Baris 21-22 : Menghitung nilai deltaAstar.
Baris 23-24 : Menghitung nilai delta.
Baris 26-28 : Menghitung nilai alfa dan alfastar baru.

5.1.3.2 Implementasi proses penghitungan nilai regresi

Kode program dari proses penghitungan nilai regresi dapat dilihat pada Gambar 5.8 berikut ini.

```
1  public void regresiLatih() {
2          for (int i = 0; i < hessianLatih.length; i++) {
3              for (int j = 0; j < fxLatih.length; j++) {
4                  fxLatih[i] += (alfastar[j] - alfa[j]) *
5  hessianLatih[i][j];
6              }
7          }
8      }
```

Gambar 5.8 Kode program proses penghitungan nilai regresi

Berikut penjelasan dari kode program normalisasi yang ditunjukkan pada Gambar 5.8.

Baris 1 : Deklarasi method regresi.

Baris 2-6 : Menghitung nilai regresi/ hasil peramalan setiap data.

5.1.3.3 Implementasi proses denormalisasi

Kode program dari proses denormalisasi dapat dilihat pada Gambar 5.9 berikut ini.

```
1      public void denormalisasiLatih(double min, double
2  range) {
3          for (int i = 0; i < fxLatih.length; i++) {
4              denormLatih[i] = (fxLatih[i] * range) + min;
5          }
6      }
```

Gambar 5.9 Kode program proses denormalisasi

Berikut penjelasan dari kode program normalisasi yang ditunjukkan pada Gambar 5.9.

Baris 1 : Deklarasi method denormalisasi.

Baris 2-6 : Menghitung nilai denormalisasi data hasil peramalan.

5.1.3.4 Implementasi proses penghitungan nilai MAPE

Kode program dari proses penghitungan nilai MAPE dapat dilihat pada Gambar 5.10 berikut ini.

```
1      public void mapeLatih(double data[][]) {
2          double temp[] = new double[mapeLatih.length];
3          for (int i = 0; i < mapeLatih.length; i++) {
4              temp[i] = (data[i][4] - denormLatih[i]) /
5  data[i][4];
6              mapeLatih[i] = Math.abs(temp[i]) * 100;
7          }
8      }
```

Gambar 5.10 Kode program proses penghitungan nilai MAPE

Berikut penjelasan dari kode program normalisasi yang ditunjukkan pada Gambar 5.10.

Baris 1 : Deklarasi method penghitungan nilai MAPE.

Baris 2-8 : Menghitung nilai MAPE setiap data.

5.1.3.5 Implementasi proses pengujian SVR

Kode program dari proses pengujian SVR dapat dilihat pada Gambar 5.11 berikut ini.

```
1     public double pengujian(double lambda, double C, double
2  epsilon, double cLR, int iterasi) {
3          this.lambda = lambda;
4          this.C = C;
5          this.epsilon = epsilon;
6          this.cLR = cLR;
7          this.iterasi = iterasi;
8
9          sequentialLearning(cLR, epsilon, C);
10         getHessianUji();
11         regresiUji();
12         denormalisasiUji(in.min, in.range);
13         mapeUji(test);
14
15         return rata(mapeUji);
16    }
```

Gambar 5.11 Kode program proses pengujian SVR

Berikut penjelasan dari kode program normalisasi yang ditunjukkan pada Gambar 5.11.

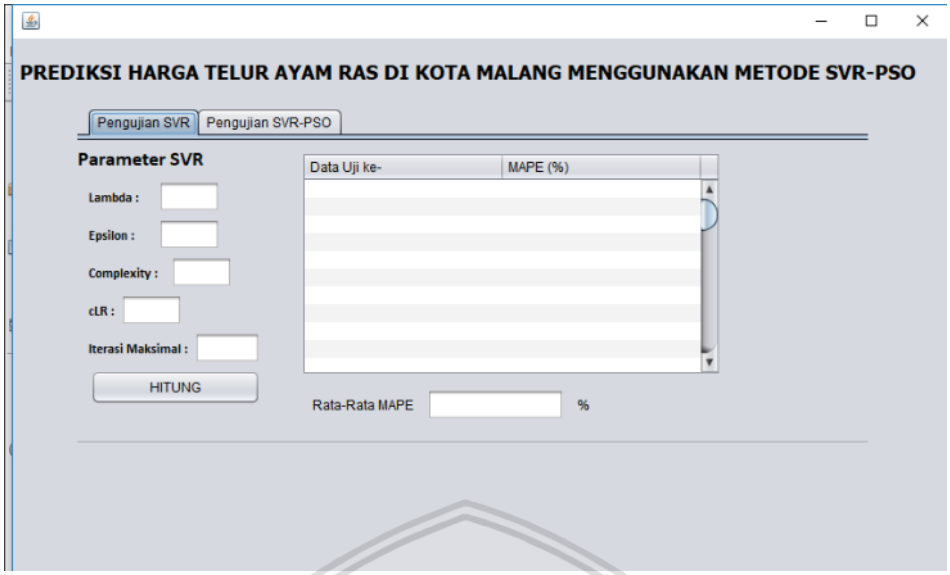
Baris 1 : Deklarasi method pengujian SVR.
Baris 3-7 : Deklarasi variabel global.
Baris 9 : Melakukan proses *sequential learning*.
Baris 10 : Menghitung matriks hessian data uji.
Baris 11 : Menghitung hasil prediksi data uji.
Baris 12 : Menghitung denormalisasi data uji
Baris 13 : Menghitung nilai MAPE data uji.
Baris 15 : Menghitung rata-rata nilai MAPE.

5.2 Implementasi antarmuka

Implementasi antarmuka akan membahas mengenai hasil implementasi dari perancangan antarmuka sistem peramalan menggunakan metode SVR-PSO. Implementasi antarmuka pada sistem peramalan ini terdiri dari implementasi halaman pengujian SVR dan implementasi halaman pengujian SVR-PSO.

5.2.1 Implementasi antarmuka halaman pengujian SVR

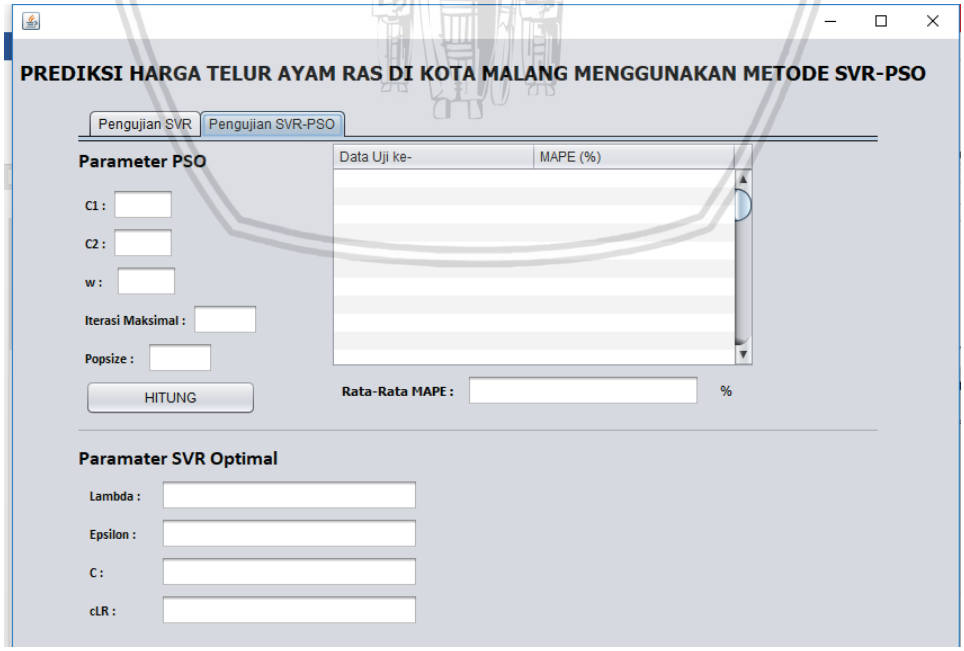
Halaman pengujian SVR merupakan sebuah tab panel yang menampilkan nilai MAPE hasil pelatihan SVR berdasarkan parameter SVR yang diinputkan oleh pengguna. Parameter yang diinputkan pengguna pada halaman ini yaitu, lambda, epsilon, cLR, C, dan iterasi maksimal pelatihan SVR. Setelah menekan tombol hitung, maka sistem akan menampilkan tabel nilai MAPE hasil peramalan untuk setiap data latih dan nilai rata-rata MAPE. Berikut gambaran implementasi antarmuka pengujian SVR dapat dilihat pada Gambar 5.12.



Gambar 5.12 Halaman pengujian SVR

5.2.2 Implementasi antarmuka halaman pengujian SVR-PSO

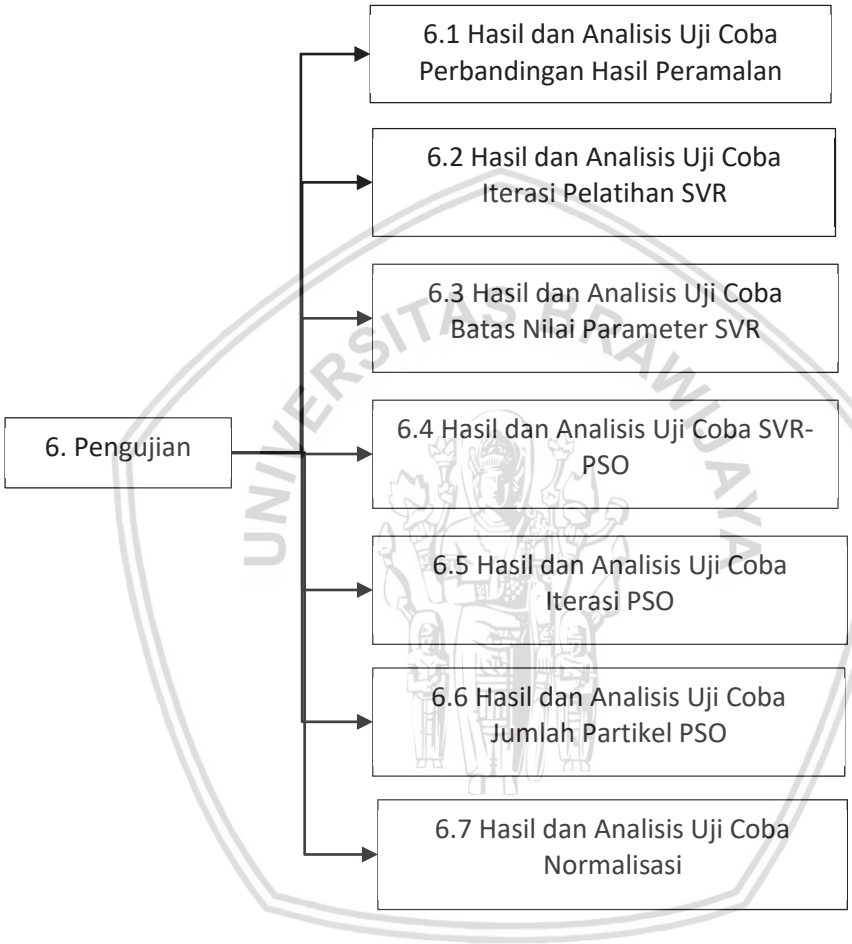
Halaman pengujian SVR-PSO merupakan sebuah tab panel yang menampilkan nilai MAPE hasil pelatihan SVR-PSO berdasarkan parameter PSO yang diinputkan oleh pengguna. Parameter yang diinputkan pengguna pada halaman ini yaitu, w, C1, C2, iterasi maksimal PSO, dan jumlah partikel PSO. Setelah menekan tombol hitung, maka sistem akan melakukan proses pelatihan SVR-PSO. Kemudian, sistem akan menampilkan tabel nilai MAPE hasil peramalan SVR-PSO dan parameter SVR hasil optimasi. Berikut gambaran implementasi antarmuka pengujian SVR-PSO dapat dilihat pada Gambar 5.13.



Gambar 5.13 Halaman pengujian SVR-PSO

BAB 6 PEMBAHASAN

Pada bab pembahasan akan membahas mengenai hasil pengujian sistem peramalan berdasarkan skenario pengujian yang telah dirancang sebelumnya. Pada tahapan pengujian ini, data yang digunakan adalah data runut waktu harga telur ayam ras di Kota Malang dengan pola data vertikal. Sistem ini menggunakan metode peramalan SVR dengan optimasi parameternya menggunakan metode PSO. Diagram alir tahapan pengujian ditunjukkan oleh Gambar 6.1.



Gambar 6.1 Diagram alir tahapan pengujian

6.1 Hasil dan Analisis Uji Coba Perbandingan Hasil Peramalan

Pengujian perbandingan hasil peramalan dilakukan untuk membandingkan hasil peramalan metode SVR dan metode SVR-PSO. Pengujian ini dilakukan dengan parameter awal SVR konstan dan parameter SVR-PSO juga konstan. Berikut parameter SVR dan PSO yang digunakan pada pengujian ini:

a. Parameter *lambda* (λ) : 0,5
b. Parameter kompleksitas (C) : 20
c. Parameter *epsilon* (ε) : 0,005
d. Parameter cLR : 0,01

e. Parameter *sigma* (σ) : 0,1
f. Jumlah Iterasi SVR : 100
g. Parameter C1 dan C2 : 2 dan 2
h. Parameter w : 0,5
i. Jumlah Iterasi PSO : 50
j. Jumlah Partikel PSO : 20

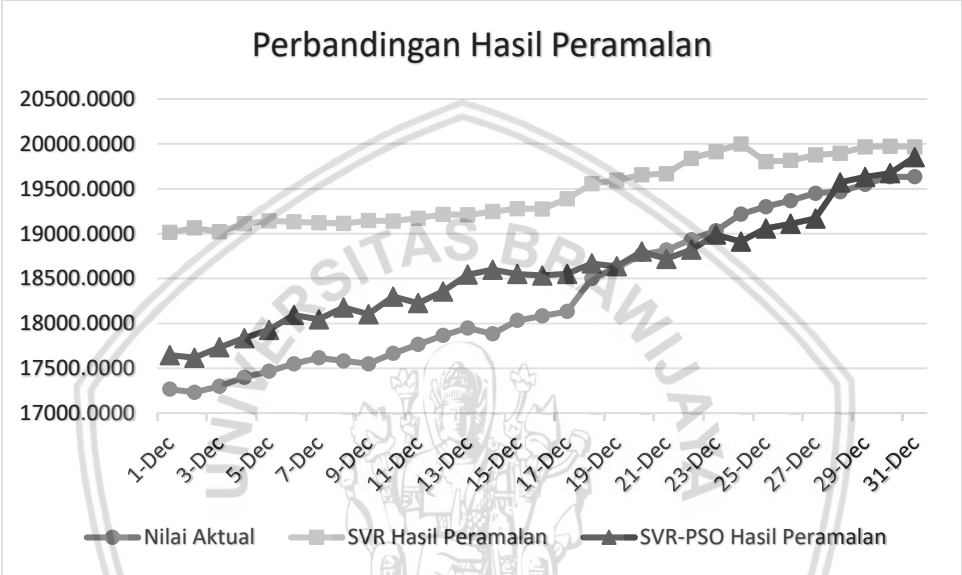
Hasil uji coba perbandingan hasil peramalan kedua metode pada bulan desember ditunjukkan oleh tabel 6.1 berikut.

Tabel 6.1 Hasil uji coba perbandingan hasil peramalan

Tanggal	Nilai Aktual	SVR		SVR-PSO	
		Hasil Peramalan	MAPE	Hasil Peramalan	MAPE
1-Dec	17266.6667	19013.2193	10.1152	17646.0928	2.1974
2-Dec	17233.3333	19063.2763	10.6186	17618.7293	2.2363
3-Dec	17300.0000	19020.1211	9.9429	17734.0028	2.5087
4-Dec	17400.0000	19109.9121	9.8271	17835.0787	2.5005
5-Dec	17466.6667	19142.7163	9.5957	17928.8979	2.6464
6-Dec	17550.0000	19130.0000	9.0028	18093.2376	3.0954
7-Dec	17616.6667	19121.7236	8.5434	18045.3824	2.4336
8-Dec	17583.3333	19112.3231	8.6957	18176.7214	3.3747
9-Dec	17550.0000	19144.9868	9.0882	18101.9281	3.1449
10-Dec	17666.6667	19139.2796	8.3355	18298.8931	3.5786
11-Dec	17766.6667	19171.7236	7.9084	18226.3413	2.5873
12-Dec	17866.6667	19215.1697	7.5476	18355.3174	2.7350
13-Dec	17950.0000	19208.1239	7.0090	18542.1038	3.2986
14-Dec	17883.3333	19243.9872	7.6085	18597.3714	3.9928
15-Dec	18033.3333	19279.0981	6.9081	18550.1876	2.8661
16-Dec	18083.3333	19274.0929	6.5848	18534.3179	2.4939
17-Dec	18133.3333	19387.0215	6.9137	18551.1732	2.3043
18-Dec	18500.0000	19553.8769	5.6966	18668.2763	0.9096
19-Dec	18616.6667	19596.0924	5.2610	18636.2316	0.1051
20-Dec	18766.6667	19654.8765	4.7329	18799.3176	0.1740
21-Dec	18816.6667	19666.0133	4.5138	18718.3813	0.5223
22-Dec	18933.3333	19838.2793	4.7796	18820.3818	0.5966
23-Dec	19033.3333	19913.2131	4.6228	18988.9487	0.2332
24-Dec	19216.6667	19998.3712	4.0678	18912.3918	1.5834
25-Dec	19300.0000	19801.8736	2.6004	19057.3179	1.2574
26-Dec	19366.6667	19815.2373	2.3162	19109.3818	1.3285
27-Dec	19450.0000	19875.3713	2.1870	19168.4814	1.4474
28-Dec	19466.6667	19895.0127	2.2004	19570.3183	0.5325
29-Dec	19550.0000	19965.2763	2.1242	19631.1038	0.4149

30-Dec	19633.3333	19972.9137	1.7296	19671.3176	0.1935
31-Dec	19633.3333	19966.8873	1.6989	19851.3183	1.1103

Berdasarkan hasil pengujian diatas, hasil peramalan metode SVR-PSO lebih akurat dibandingkan dengan metode SVR. Metode SVR menghasilkan nilai MAPE rata-rata sebesar 6.2186, sedangkan metode SVR-PSO menghasilkan nilai MAPE sebesar 1.8840. Metode PSO berhasil meningkatkan kinerja peramalan SVR, yaitu dengan melakukan pencarian terhadap parameter SVR optimal secara metaheuristik. Berikut grafik hasil uji coba perbandingan hasil peramalan yang ditunjukkan oleh Gambar 6.2.



Gambar 6.2 Grafik hasil pengujian perbandingan hasil peramalan

6.2 Hasil dan Analisis Uji Coba Iterasi Pelatihan SVR

Pengujian iterasi pelatihan SVR dilakukan untuk menentukan jumlah iterasi pelatihan SVR yang optimal. Pengujian ini dilakukan dengan parameter awal SVR acak. Berikut batas parameter SVR yang digunakan pada pengujian ini:

a. Batas parameter *lambda* (λ) : 0,1 - 1
b. Batas parameter kompleksitas (C) : 1 - 1000
c. Batas arameter *epsilon*(ε) : 0,0001 – 0,01
d. Batas parameter cLR : 0,01 – 0,1
e. Parameter *sigma* (σ) : 0,1

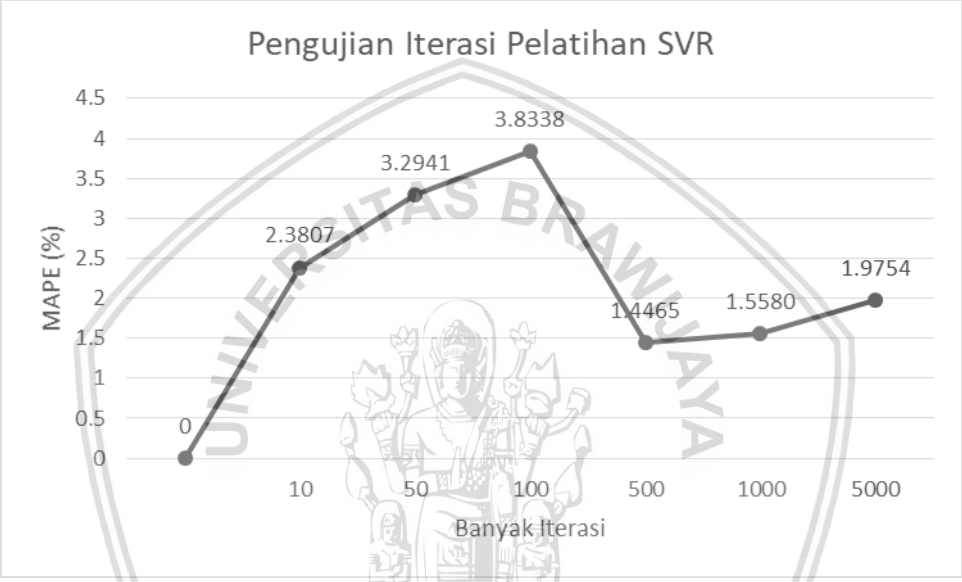
Hasil uji coba iterasi pelatihan SVR ditunjukkan oleh Tabel 6.2 berikut.

Tabel 6.2 Hasil uji coba iterasi pelatihan SVR

Jumlah Iterasi	MAPE Percobaan ke -					Rata-rata MAPE
	1	2	3	4	5	
10	1.9581	1.9538	1.9538	3.6422	2.3958	2.3807
50	3.5601	3.540825	3.5408	2.0391	3.7897	3.2941
100	3.8004	3.7734	3.7734	3.8984	3.9235	3.8338

500	1.4294	1.4231	1.4227	1.4787	1.4787	1.4465
1000	1.3925	1.5602	1.5597	1.6388	1.6388	1.5580
5000	1.4713	2.1449	2.1440	2.0584	2.0584	1.9754

Berdasarkan hasil pengujian pelatihan SVR pada semua iterasi yang ditunjukkan oleh Tabel 6.2 diatas, iterasi dengan rata-rata nilai MAPE terkecil adalah iterasi 50 dengan nilai rata-rata MAPE sebesar 1.4465. Selain itu, semakin banyak iterasi yang dilakukan saat pelatihan SVR, maka hasil peramalan akan semakin baik. Berikut grafik hasil uji coba iterasi pelatihan SVR yang ditunjukkan oleh Gambar 6.3.



Gambar 6.3 Grafik hasil pengujian iterasi pelatihan SVR

6.3 Hasil dan Analisis Uji Coba Nilai Batas Parameter SVR

Pengujian batas nilai parameter SVR yang akan di uji coba adalah parameter λ, C, dan ε. Pengujian ini dilakukan untuk menentukan batas ruang pencarian dimensi yang optimal sehingga menghasilkan kombinasi parameter SVR yang optimal.

6.3.1 Hasil dan analisis uji coba batas nilai parameter lambda λ

Pengujian ini bertujuan untuk menentukan batas nilai parameter *lambda* yang memiliki hasil terbaik. Berikut batas parameter SVR yang digunakan untuk menguji batas nilai parameter *lambda*.

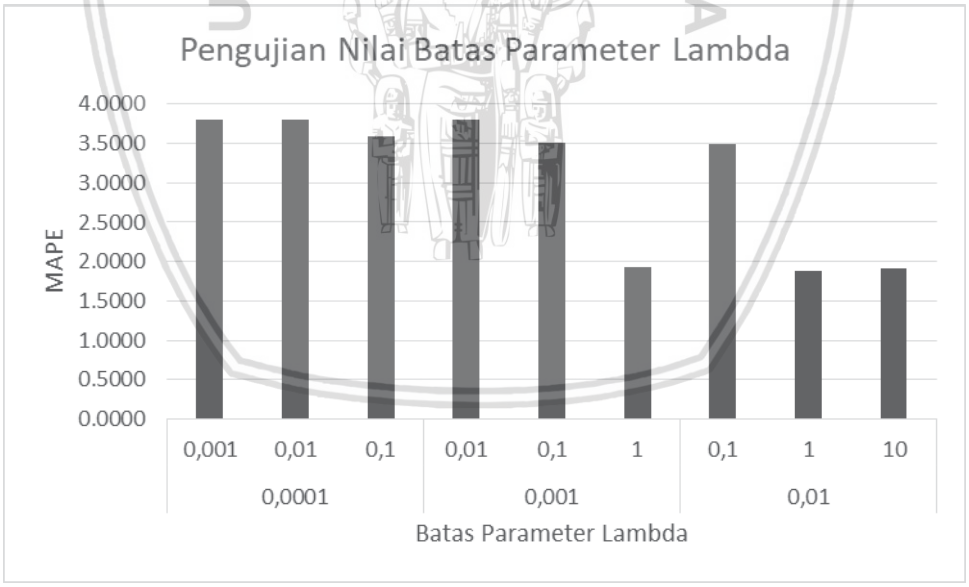
a. Batas parameter kompleksitas (C) : 1 - 1000
b. Batas parameter *epsilon*(ε) : 0,0001 – 0,01
c. Batas parameter cLR : 0,01 – 0,1
d. Parameter *sigma* (σ) : 0,1

Hasil uji coba batas nilai parameter λ ditunjukkan oleh Tabel 6.3 berikut.

Tabel 6.3 Hasil uji coba batas nilai parameter λ

Batas Min	Batas Maks	MAPE Percobaan ke -					Rata-rata MAPE
		1	2	3	4	5	
0,0001	0,001	3.8014	3.8014	3.8014	3.8014	3.8014	3.8014
	0,01	3.8014	3.8014	3.7948	3.7998	3.7972	3.7989
	0,1	3.8014	3.7982	3.4966	3.6412	3.2141	3.5903
0,001	0,01	3.8014	3.7948	3.8004	3.7998	3.7972	3.7987
	0,1	3.8014	3.2141	3.4101	3.3155	3.7948	3.5072
	1	3.8014	0.6191	3.7948	0.6467	0.7946	1.9313
0,01	0,1	3.7948	3.2141	3.6412	3.3155	3.4966	3.4924
	1	3.7948	0.6191	0.6948	3.6412	0.6467	1.8793
	10	3.7948	0.8248	0.6191	0.7141	3.6412	1.9188

Berdasarkan hasil pengujian batas nilai parameter *lambda* yang ditunjukkan oleh Tabel 6.3 diatas, Batas parameter *lambda* yang menghasilkan rata-rata nilai MAPE terkecil adalah 0.01 – 1 yaitu sebesar 1.8793. Selain itu, semakin besar nilai *lambda,* akurasi peramalan yang dihasilkan semakin baik. Berikut grafik hasil uji coba batas nilai parameter *lambda* yang ditunjukkan oleh Gambar 6.4.



Gambar 6.4 Grafik hasil pengujian batas nilai parameter lambda

6.3.2 Hasil dan analisis uji coba batas nilai parameter C

Pengujian ini bertujuan untuk menentukan batas nilai parameter kompleksitas (C) yang memiliki hasil terbaik. Berikut batas parameter SVR yang digunakan untuk menguji batas nilai parameter *C*.

a. Batas parameter *lambda* (λ) : 0,1 - 1
b. Batas parameter *epsilon*(ε) : 0,0001 – 0,01

c. Batas parameter cLR : 0,01 – 0,1

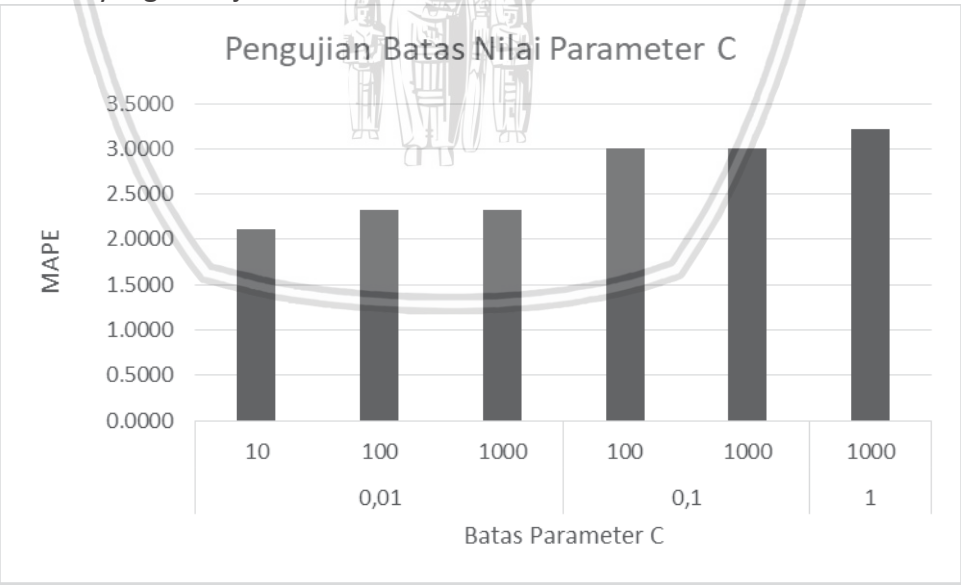
d. Parameter *sigma* (σ) : 0,1

Hasil uji coba batas nilai parameter C ditunjukkan oleh Tabel 6.4 berikut.

Tabel 6.4 Hasil uji coba batas nilai parameter C

Batas Min	Batas Maks	MAPE Percobaan ke -					Rata-rata MAPE
		1	2	3	4	5	
0.01	10	0.5432	3.2141	1.4672	2.1180	3.2141	2.1113
	100	0.5432	3.2141	1.4672	3.2141	3.2141	2.3305
	1000	0.5432	3.2141	1.4672	3.2141	3.2141	2.3305
0.1	100	2.2393	3.2141	3.1647	3.2141	3.2141	3.0093
	1000	2.2393	3.2141	3.1647	3.2141	3.2141	3.0093
1	1000	3.2141	3.2141	3.2141	3.2141	3.2141	3.2141

Berdasarkan hasil pengujian batas nilai parameter C yang ditunjukkan oleh Tabel 6.4 diatas, batas parameter C dengan nilai MAPE terkecil yaitu 0.01 – 10 sebesar 2,1113. Selain itu, semakin kecil nilai C, maka hasil peramalan akan semakin baik. Hal ini dikarenakan, parameter C ini dianggap sebagai nilai penalti, semakin kecil nilai C, maka semakin besar nilai error yang ditoleransi oleh sistem, sehingga nilai akurasi peramalan meningkat. Berikut grafik hasil uji coba batas nilai parameter C yang ditunjukkan oleh Gambar 6.5.



Gambar 6.5 Grafik hasil pengujian batas nilai parameter C

6.3.3 Hasil dan analisis uji coba batas nilai parameter epsilon ε

Pengujian ini bertujuan untuk menentukan batas nilai parameter *epsilon* yang memiliki hasil terbaik. Berikut batas parameter SVR yang digunakan untuk menguji batas nilai parameter *epsilon*.

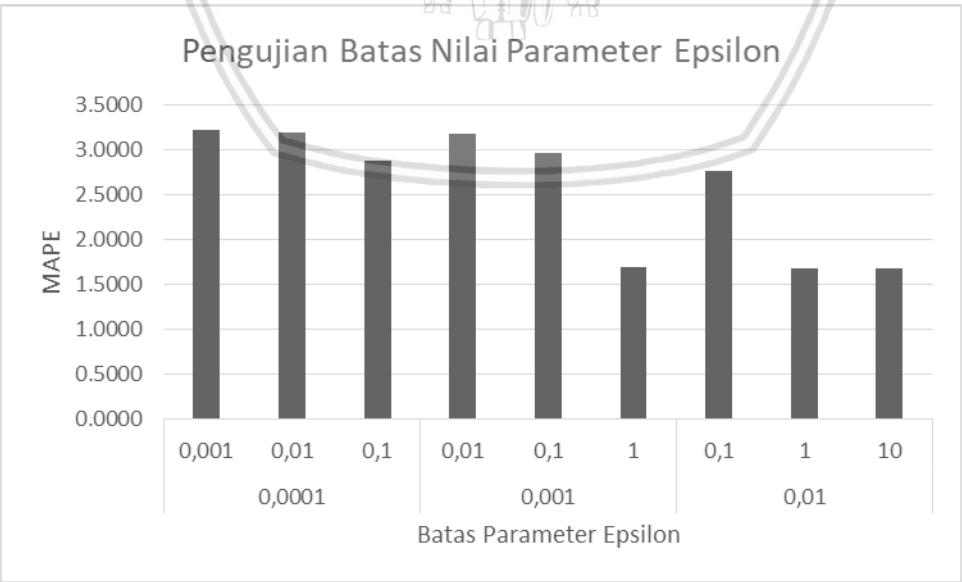
a. Batas parameter *lambda* (λ) : 0,1 - 1
b. Batas parameter kompleksitas (C) : 1 - 1000
c. Batas parameter cLR : 0,01 – 0,1
d. Parameter *sigma* (σ) : 0,1

Hasil uji coba batas nilai parameter ε ditunjukkan oleh Tabel 6.5 berikut.

Tabel 6.5 Hasil uji coba batas nilai parameter ε

Batas Min	Batas Maks	MAPE Percobaan ke -					Rata-rata MAPE
		1	2	3	4	5	
0,0001	0,001	3.2141	3.2099	3.2132	3.2118	3.2103	3.2118
	0,01	3.2141	3.1585	3.2103	3.1651	3.1891	3.1874
	0,1	3.2141	2.5197	3.1891	2.8831	2.5848	2.8782
0,001	0,01	3.2099	3.1585	3.1651	3.1838	3.1942	3.1823
	0,1	3.2099	2.5197	3.1942	2.9537	2.9537	2.9662
	1	3.2099	0.9227	2.9537	0.8739	0.5158	1.6952
0,01	0,1	3.1585	2.5197	2.8120	2.5848	2.7366	2.7623
	1	3.1585	0.9227	2.7366	0.6815	0.8739	1.6746
	10	3.1585	0.9227	2.7366	0.6815	0.9227	1.6844

Berdasarkan hasil pengujian batas nilai parameter *epsilon* yang ditunjukkan oleh Tabel 6.5 diatas, Batas parameter *epsilon* yang menghasilkan rata-rata nilai MAPE terkecil adalah 0,01 – 1 yaitu sebesar 1,6746. Selain itu, semakin besar nilai epsilon, maka akurasi peramalan akan semakin baik. Berikut grafik hasil uji coba batas nilai parameter *epsilon* yang ditunjukkan oleh Gambar 6.6.



Gambar 6.6 Grafik hasil pengujian batas nilai parameter epsilon

6.4 Hasil dan Analisis Uji Coba SVR-PSO

Pengujian SVR-PSO dilakukan untuk mengetahui kinerja metode SVR setelah dioptimasi dengan metode PSO untuk melakukan peramalan. Pada pengujian ini dilakukan sebanyak 10 kali percobaan dan parameter awal PSO selalu konstan. Berikut parameter PSO yang digunakan pada pengujian ini.

a. Parameter C_1 : 0.5
b. Parameter C_2 : 1.5
c. Parameter w : 0.5
d. Jumlah partikel : 5

Hasil uji coba SVR-PSO menggunakan parameter yang telah ditentukan sebelumnya ditunjukkan oleh Tabel 6.6 berikut.

Tabel 6.6 Hasil uji coba SVR-PSO

Percobaan ke -	MAPE	*Fitness*
1	4.3775	0.1860
2	0.9227	0.5201
3	0.9227	0.5201
4	0.9227	0.5201
5	3.0851	0.2448
6	4.6219	0.1779
7	4.3406	0.1872
8	0.9227	0.5201
9	0.9227	0.5201
10	3.7963	0.2085
Rata-rata	2.4835	0.2871
Best	0.9227	0.5201

Berdasarkan hasil pengujian SVR-PSO yang ditunjukkan oleh Tabel 6.6 diatas, didapatkan nilai *fitness* terbaik sebesar 0.5201 dengan nilai MAPE-nya yaitu 0.9227. Pada pengujian ini, nilai parameter SVR yang optimal yaitu:

- Parameter lambda : 0.820987174519858
- Parameter C : 1
- Parameter epsilon : 0.009094320238180402
- Parameter cLR : 0.6915749892776809

6.5 Hasil dan Analisis Uji Coba Iterasi PSO

Pengujian iterasi PSO dilakukan untuk menentukan jumlah iterasi PSO dimana sistem mencapai konvergensi. Berikut parameter PSO yang digunakan pada pengujian ini.

a. Parameter C_1 : 2
b. Parameter C_2 : 2

c. Parameter w : 0.5

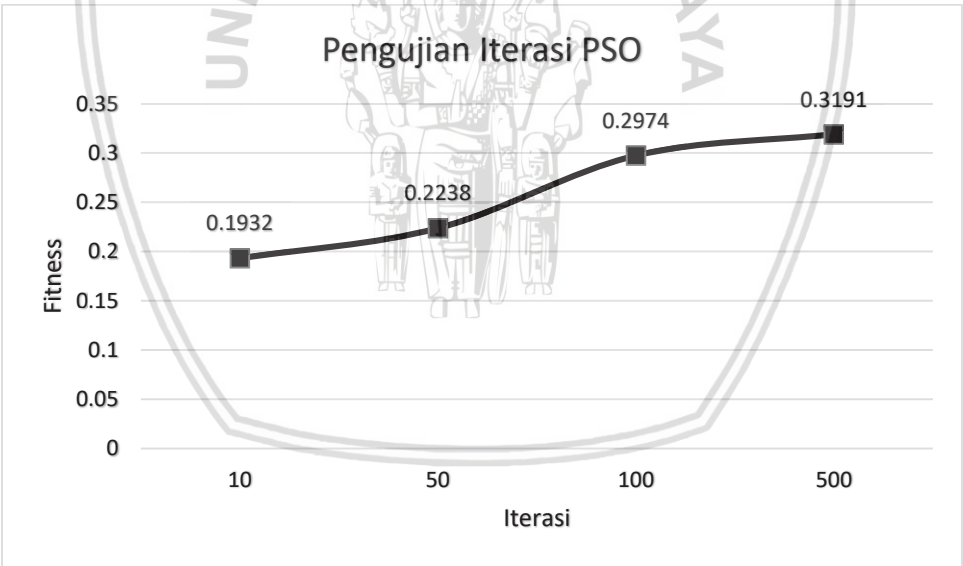
d. Jumlah partikel : 30

Hasil uji coba iterasi PSO menggunakan parameter yang telah ditentukan sebelumnya ditunjukkan oleh Tabel 6.8 berikut.

Tabel 6.7 Hasil uji coba iterasi PSO

Jumlah Iterasi	Percobaan ke -					*Fitness*
	1	2	3	4	5	
10	0.1783	0.1780	0.1779	0.1780	0.2539	0.1932
50	0.1727	0.1798	0.1779	0.1779	0.4106	0.2238
100	0.5201	0.1738	0.1750	0.3205	0.2978	0.2974
500	0.1799	0.2876	0.5201	0.2251	0.3828	0.3191

Berdasarkan hasil pengujian iterasi PSO yang ditunjukkan oleh Tabel 6.7 diatas, iterasi dengan nilai rata-rata *fitness* terbaik adalah 500 iterasi. Semakin banyak iterasi yang dilakukan, semakin baik nilai *fitness* yang dihasilkan oleh sistem. Hal ini disebabkan karena sistem akan terus mengeksplorasi ruang pencarian dan akan berhenti hingga mencapai iterasi maksimal. Berikut grafik hasil uji coba iterasi PSO yang ditunjukkan oleh Gambar 6.7.



Gambar 6.7 Grafik hasil pengujian iterasi PSO

6.6 Hasil dan Analisis Uji Coba Jumlah Partikel PSO

Pengujian jumlah partikel PSO dilakaukan untuk menentukan jumlah partikel PSO optimal yang memiliki nilai *fitness* tertinggi. Berikut parameter PSO yang digunakan pada pengujian ini.

a. Parameter C_1 : 0.5

b. Parameter C_2 : 1.5

c. Parameter w : 0,5

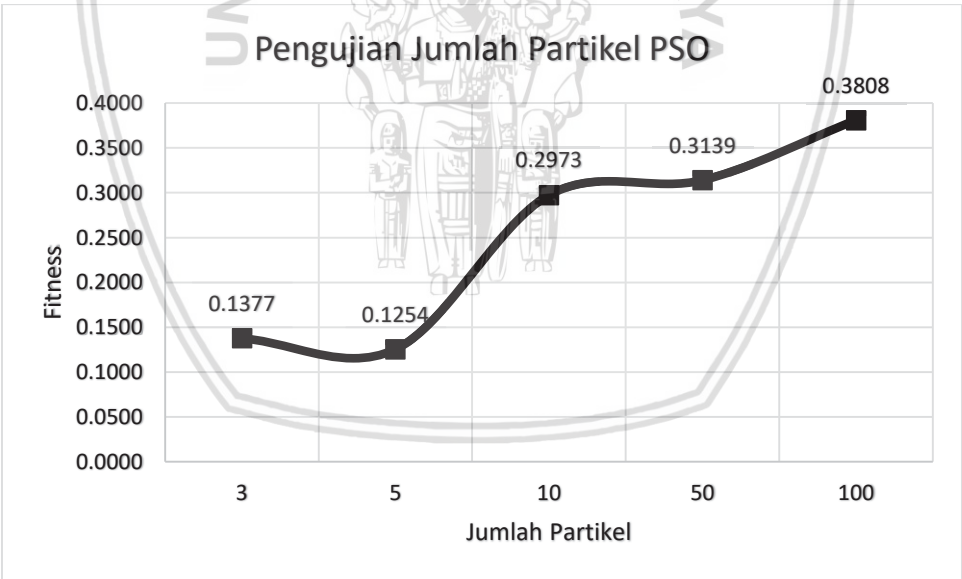
d. Iterasi PSO : 100

Hasil uji coba SVR-PSO menggunakan parameter yang telah ditentukan sebelumnya ditunjukkan oleh Tabel 6.8 berikut.

Tabel 6.8 Hasil uji coba jumlah partikel PSO

Jumlah Partikel	Percobaan ke -					Rata-rata *Fitness*	Rata-rata MAPE
	1	2	3	4	5		
3	0.2686	0.0053	0.0142	0.0193	0.3809	0.1377	62.3734
5	0.0144	0.0129	0.0141	0.1750	0.4106	0.1254	44.2485
10	0.5201	0.1779	0.1563	0.1121	0.5201	0.2973	2.3579
50	0.1731	0.5201	0.1779	0.5201	0.1783	0.3139	3.1705
100	0.5201	0.5201	0.1718	0.1720	0.5201	0.3808	2.4802

Berdasarkan hasil pengujian jumlah partikel PSO yang ditunjukkan oleh Tabel 6.8 diatas, jumlah partikel dengan nilai rata-rata *fitness* terbaik adalah 100 partikel. Semakin banyak partikel PSO, maka semakin baik dan stabil nilai *fitness* yang dihasilkan. Hal ini dikarenakan, semakin banyak partikel yang digunakan, semakin banyak dan beragam nilai acak yang di eksplor oleh sistem. Berikut grafik hasil uji coba jumlah partikel PSO yang ditunjukkan oleh Gambar 6.8.



Gambar 6.8 Grafik hasil pengujian jumlah partikel PSO

6.7 Hasil dan Analisis Uji Coba Normalisasi

Pengujian normalisasi ini bertujuan untuk membandingkan hasil peramalan menggunakan metode SVR-PSO untuk data yang telah dinormalisasi dan data yang tanpa dinormalisasi. Berikut parameter PSO yang digunakan pada pengujian ini.

a. Parameter C1 dan C2 : 2 dan 2
b. Parameter w : 0,5

c. Jumlah Iterasi PSO : 50

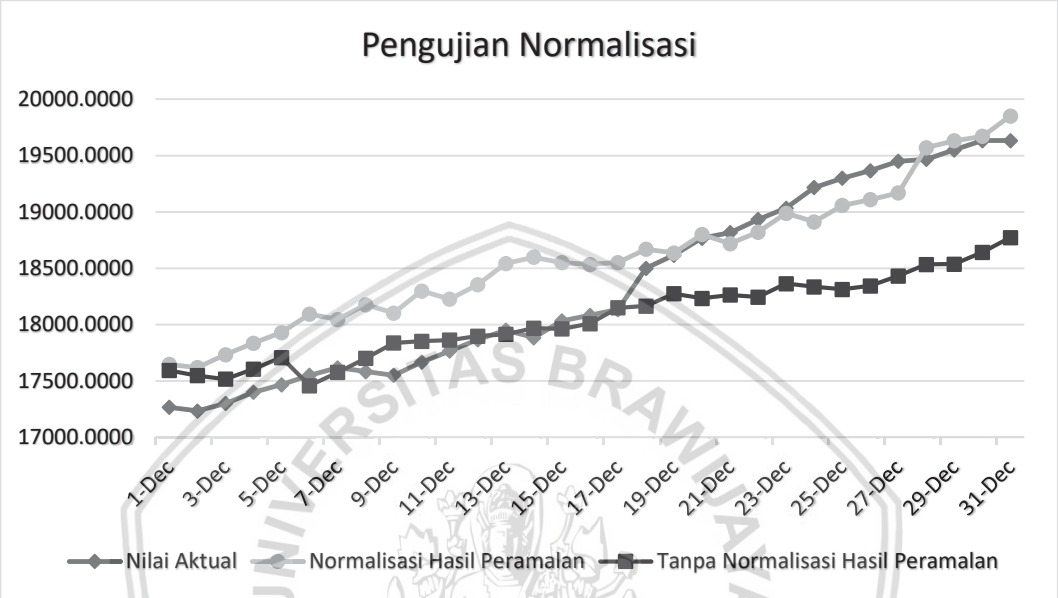
d. Jumlah Partikel PSO : 20

Hasil uji coba normalisasi menggunakan parameter yang telah ditentukan sebelumnya ditunjukkan oleh Tabel 6.9 berikut.

Tabel 6.9 Hasil uji coba normalisasi

Tanggal	Nilai Aktual	Normalisasi		Tanpa Normalisasi	
		Hasil Peramalan	MAPE	Hasil Peramalan	MAPE
1-Dec	17266.6667	17646.0928	2.1974	17592.8213	1.8889
2-Dec	17233.3333	17618.7293	2.2363	17547.3131	1.8219
3-Dec	17300.0000	17734.0028	2.5087	17516.3717	1.2507
4-Dec	17400.0000	17835.0787	2.5005	17606.2631	1.1854
5-Dec	17466.6667	17928.8979	2.6464	17708.3163	1.3835
6-Dec	17550.0000	18093.2376	3.0954	17457.6765	0.5261
7-Dec	17616.6667	18045.3824	2.4336	17576.3162	0.2290
8-Dec	17583.3333	18176.7214	3.3747	17701.3165	0.6710
9-Dec	17550.0000	18101.9281	3.1449	17836.3163	1.6314
10-Dec	17666.6667	18298.8931	3.5786	17852.3162	1.0508
11-Dec	17766.6667	18226.3413	2.5873	17863.2714	0.5437
12-Dec	17866.6667	18355.3174	2.7350	17897.6557	0.1734
13-Dec	17950.0000	18542.1038	3.2986	17913.3761	0.2040
14-Dec	17883.3333	18597.3714	3.9928	17969.7323	0.4831
15-Dec	18033.3333	18550.1876	2.8661	17962.3123	0.3938
16-Dec	18083.3333	18534.3179	2.4939	18009.3152	0.4093
17-Dec	18133.3333	18551.1732	2.3043	18149.3176	0.0881
18-Dec	18500.0000	18668.2763	0.9096	18164.3123	1.8145
19-Dec	18616.6667	18636.2316	0.1051	18274.3124	1.8390
20-Dec	18766.6667	18799.3176	0.1740	18232.5787	2.8459
21-Dec	18816.6667	18718.3813	0.5223	18262.2365	2.9465
22-Dec	18933.3333	18820.3818	0.5966	18244.3123	3.6392
23-Dec	19033.3333	18988.9487	0.2332	18364.3123	3.5150
24-Dec	19216.6667	18912.3918	1.5834	18335.3162	4.5864
25-Dec	19300.0000	19057.3179	1.2574	18312.3213	5.1175
26-Dec	19366.6667	19109.3818	1.3285	18343.7231	5.2820
27-Dec	19450.0000	19168.4814	1.4474	18431.3126	5.2375
28-Dec	19466.6667	19570.3183	0.5325	18534.3632	4.7892
29-Dec	19550.0000	19631.1038	0.4149	18537.3126	5.1800
30-Dec	19633.3333	19671.3176	0.1935	18641.3213	5.0527
31-Dec	19633.3333	19851.3183	1.1103	18772.3123	4.3855
Rata - Rata		**1.8840**		**2.2634**	

Berdasarkan hasil pengujian normalisasi yang ditunjukkan oleh Tabel 6.9 diatas, nilai MAPE metode SVR-PSO dengan menggunakan normalisasi data sebesar 1.8840, sedangkan metode SVR-PSO yang tanpa menggunakan normalisasi data menghasilkan nilai MAPE sebesar 2.2634. Proses normalisasi data terbukti mampu meningkatkan nilai akurasi hasil peramalan, hal ini dikarenakan proses normalisasi menyetarakan *range* data, sehingga nilai *error* berkurang. Berikut grafik hasil uji normalisasi yang ditunjukkan oleh Gambar 6.9.



Gambar 6.9 Grafik hasil pengujian normalisasi

BAB 7 PENUTUP

7.1 Kesimpulan

Kesimpulan yang dapat diambil dari penelitian peramalan harga telur ayam ras yang didasarkan pada rumusan masalah di Bab 1.

1. Perancangan peramalan yang tepat dan baik sangat berpengaruh terhadap hasil peramalan runut waktu yang akan dilakukan. Perancangan peramalan terdiri dari definisi masalah, pengumpulan data, analisis data, seleksi model peramalan, validasi model peramalan, dan pengamatan peramalan. Dalam melakukan peramalan data runut waktu dilakukan secara kuantitatif yang disesuai dengan pola data yang digunakan.
2. Metode yang digunakan untuk peramalan data runut waktu hara telur ayam ras ini adalah metode SVR-PSO. Metode SVR digunakan untuk melakukan peramalan dan metode PSO digunakan untuk mengoptimasi parameter SVR. Parameter SVR yang dioptimasi adalah parameter *lambda*, kompleksitas, *epsilon*, dan *constanta learning rate* (cLR). Evaluasi kinerja peramalan SVR menggunakan nilai *Mean Absolute Percentage Error* (MAPE). Penghitungan nilai *fitness* setiap partikel PSO dihitung berdasarkan nilai MAPE hasil dari pelatihan SVR dari partikel. Nilai *fitness* terbaik selanjutnya akan digunakan untuk pengujian SVR.
3. Pada penelitian ini, evaluasi kinerja sistem peramalan harga telur ayam mengacu pada nilai MAPE. Nilai MAPE adalah persentase rata-rata absolut hasil perbandingan nilai *error* dari nilai hasil peramalan dengan data aktual. Semakin kecil nilai MAPE yang dihasilkan, maka semakin baik sistem melakukan peramalan. Berdasarkan hasil pengujian yang telah dilakukan menggunakan metode SVR dan metode SVR-PSO didapatkan hasil berbeda. Normalisasi data dilakukan untuk memperkecil *range* data, sehingga nilai error semakin kecil. Pada pengujian SVR bulan desember didapatkan nilai MAPE terkecil yaitu sebesar 6,2186% dan pada pengujian SVR-PSO didapatkan nilai MAPE terkecil sebesar 1,8840%.

7.2 Saran

Berdasarkan penelitian yang telah dilakukan pada peramalan harga telur ayam menggunakan metode SVR-PSO, diperoleh saran yang dapat digunakan untuk pengembangan penelitian selanjutnya, yaitu:

1. Diperlukan pengembangan terhadap metode PSO, seperti melakukan velocity clamping, TVIC dan TVAC.
2. Melakukan pengujian parameter awal PSO yaitu parameter bobot inersia (w), konstanta c_1 dan c_2.
3. Melakukan pengujian fungsi kernel SVR lainnya seperti kernel linier, kernel polinomial, kernel ANOVA, kernel eksponensial, dan lain-lain.
4. Penambahan pengujian terhadap pola data horizontal.

DAFTAR PUSTAKA

Alam, D. S., Koh, R. Y. & P, R. S., 2014. *Research on Particle Swarm Optimization Based Clustering: A Sistematic Review of Literature and Techniques.* s.l.:Swarm and Evolutionary Computation.

Arumsari, N. & Pamuji, F. A., 2017. PERAMALAN IRRADIANCE CAHAYA MATAHARI PADA SEL SURYA UNTUK MEMENUHI KEBUTUHAN ENERGI LISTRIK DENGAN METODE SUPPORT VECTOR REGRESSION (SVR). *Jurnal Nasional Teknik Elektro,* 6(1), pp. 2302-2949.

Astuti, N., 2016. *Penerapan Logika Fuzzy Mamdani Untuk Prediksi Hasil Produksi Telur Ayam.* s.l.:s.n.

Attiya, I. & Zhang, X., 2017. A Simplified Particle Swarm Optimization for Job Scheduling in Cloud Computing. *International Journal of Computer Applications,* Volume 163.

Ceperic, E. & Ceperic, V., 2013. A Strategy for Short-Term Load Forecasting by Suport Vector Regression Machines. *IEEE TRANSACTIONS ON POWER SYSTEMS.*

Cortes & Vapnik, 1995. *Support-vector networks.* s.l.:s.n.

Disperindag, 2011. *Sistem Informasi Ketersediaan dan Perkembangan Harga Bahan Pokok di Jawa Timur.* [Online] Available at: www.siskaperbapo.com/harga/tabel [Accessed 05 06 2018].

Duan, P., Xie, K., Guo, T. & Huang, X., 2011. Short-term Load Forecasting for Electric Power Systems Using the PSO-SVR and FCM Clustering Techniques. *Energies.*

Eberhart, R. C. & Shi, Y., 2001. Particle Swarm Optimization : Developments, Applications, Resources.

Eliantara, F., Cholissodin, I. & Indriarti, 2016. Optimasi Pemenuhan Kebutuhan Gizi Keluarga Menggunakan Particle Swarm Optimization. *Prosiding SNRT (Seminar Nasional Riset Terapan.*

Fattahi, H., 2015. Prediction of Earthquake Induced Displacements of Slopes Using Hybrid Support Vector Regression with Particle Swarm Optimization. *International Journal of Optimization in Civil Engineering.*

Febrianto, N. & Putritamara, J. A., 2013. Proyeksi elastisitas permintaan telur ayam ras di Malang Raya. *Jurnal Ilmu-Ilmu Peternakan.*

Ferdiansyah, R. M., 2013. *Sistem Prediksi Harga Daging Sapi Lokal Di Kota Semarang Menggunakan Metode Moving Average.* s.l.:s.n.

Guajardo, J., Weber, R. & Miranda, J., 2006. A Forecasting Methodology Using Support Vector Regression and Dynamic Feature Selection. *Journal of Information & Knowledge Management.*

Hao, W. & Yu, S., 2006. Support Vector Regression For Financial Time Series Forecasting. *International Federation for Information Processing,* Volume 207, pp. 825-830.

Hassibi, K., 2016. *Machine Learning vs. Traditional Statistics: Different philosophies, Different Approaches.* [Online] Available at: https://www.datasciencecentral.com/profiles/blogs/machine-

learning-vs-traditional-statistics-different-philosophi-1
[Accessed 5 6 2018].

Hsieh, H. I., Lee, T.-P. & Lee, T.-S., 2011. A Hybrid Particle Swarm Optimization and Support Vector. *International Journal of Business Administration.*

Hu, G., Li, H., Li, K. & Liu, W., 2010. Grid Resources Prediction with Support Vector Regression and Particle Swarm Optimization. *Third International Joint Conference on Computational Science and Optimization.*

Hu, W., Yan, L., Liu, K. & Wang, H., 2015. PSO-SVR: A Hybrid Short-term Traffic Flow Forecasting Method. *2015 IEEE 21st International Conference on Parallel and Distributed Systems (ICPADS).*

Ju & Hong, 2013. *Application of seasonal SVR with chaotic gravitational search algorithm in electricity forecasting.* s.l.:Applied Mathematical.

Junyou, B., 2007. Stock Price Forecasting using PSO-trained Neural Network. *2007 IEEE Congress on Evolutionary Computation.*

Kalanaki, M., Soltani, J. & Tavassoli, S., 2013. The use of hybrid SVR-PSO model to predict pipes failure rates. *International Journal of Scientific & Engineering Research.*

Lin, S., Wang, G., Zhang, S. & Li, J., 2006. Time Series Prediction Based on Support Vector Regression. 5(353-357).

Lin, T.-L.et al., 2010. An efficient job-shop scheduling algorithm based on particle swarm optimization. *Expert Systems with Applications 37,* pp. 2629-2636.

Li, S.-F. & Cheng, C.-Y., 2017. Particle swarm optimization with *fitness* adjustment parameters. *Computer & Industrial Engineering.*

Lu, X. & Geng, X., 2011. Car Sales Volume Prediction Based on Particle Swarm Optimization Algorithm and Support Vector Regression. *Fourth International Conference on Intelligent Computation Technology and Automation.*

Mahmudy, W. F., 2015. *Dasar-Dasar Algoritme Evolusi.* s.l.:s.n.

Makridakis, S., Wheelwright, S. C. & McGree, V. E., 1983. *Forecasting: Method and Applications.* s.l.:s.n.

Marini, F. & Walczak, B., 2015. Particle Swarm Optimization (PSO. A Tutorial). *Chemometrics and Intelligent Laboratory Systems.*

Min, J. H. & Lee, Y.-C., 2005. Bankruptcy Prediction Using Support Vector Machines with Optimal Choice of Kernel Function Parameters. *Expert Systems with Applications.*

Mishra, S. & Patra, S. K., 2008. Short Term Load Forecasting using Neural Network trained with Genetic Algoritm & Particle Swarm Optimization. *First International Conference on Emerging Trends in Engineering and Technology.*

Montgomery, D. C., Jennings, C. L. & Kulahci, M., 2015. *Introduction to Time Series Analysis And Forecasting.* 2nd ed. s.l.:Wiley.

Novitasari, D., Cholissodin, I. & Mahmudy, W. F., 2016. Optimizing SVR using Local Best PSO for Software Effort Estimation. *Journal of Information Technology and Computer Science,* Volume 1, pp. 28-37.

Patro, S. G. K. & Sahu, K. K., 2015. Normalization: A Preprocessing Stage. *International Advanced Research Journal in Science, Engineering and Technology.*

Pei-you, C. & Lu, L., 2013. Study on Coal Logistics Demand Forecast Based on PSO-SVR. *2013 10th International Conference on Service Systems and Service Management,* pp. 130-133.

Ratnaweera, A., Halgamuge, S. K. & Watson, H. C., 2004. Self-Organizing Hierarchical Particle Swarm Optimizer With Time-Varying Accelaration Coefficients. *IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION.*

Rodriguez, A. N. & Ferreire, T., 2009. Prime Step in The Time Series Forecasting With Hybrid Methods: The *Fitness* Function Choice. *Proceedings of International Joint Conference on Neural Network, Atlanta, Georgia, USA.*

Rosli, N., Ibrahim, R. & Ismail, I., 2016. Intelligent Prediction System for Gas Metering System Using Particle Swam Optimization in Training Neural Network. *2016 IEE Symposium on Robotics and Intelligent Sensors,* Volume 105, pp. 165-169.

Smola, A., 2003. *A Tutorial on Support Vector Regression.* s.l.:Royal Holloway College, University of London UK.

Soebroto, A. A., Cholissodin, I. & Wihandika, R. C., 2015. PREDIKSI TINGGI MUKA AIR (TMA) UNTUK DETEKSI DINI BENCANA BANJIR MENGGUNAKAN SVR-TVIWPSO. *Jurnal Teknologi Informasi dan Ilmu Komputer (JTIIK),* pp. 79-86.

Souza, C. R. d., 2010. *Kernel Functions for Machine Learning Applications.* [Online] Available at: http://crsouza.com/2010/03/17/kernel-functions-for-machine-learning-applications/#linear [Accessed 27 2 2018].

Stalony, V. E., 2013. *Manajemen Industri : Metode Peramalan (Forecasting).* [Online] Available at: https://vebyenandes.wordpress.com/2013/03/27/manajemenindustri-metodeperamalanforecasting/ [Accessed 18 September 2017].

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

Wang, K.-P., Huang, L., Zhou, C.-G. & Pang, W., 2003. Particle Swarm Optimization For Traveling Salesman Problem. *Proceedings of the Second International Conference on Machine Learning and Cybernetics.*

Wang, W.-C., Chau, K.-W., Cheng, C.-T. & Qiu, L., 2009. A comparison of performance of several artificial intelligence methods for forecasting monthly discharge time series. *Journal of Hydrology.*

Woschnagg, E. & Cipan, J., 2004. Evaluating Forecast Accuracy. *406347 UK Ökonometrische Prognose.*

Wu, C.-H.et al., 2003. Travel Time Prediction with Support Vector Regression. *IEEE.*

Yeh, W.-C., 2013. New Parameter-Free Simplified Swarm Optimization for Artificial Neural Network Training and Its Application in the Prediction of Time Series. *IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS.*

Yu, X., Qi, Z. & Zhao, Y., 2013. Support Vector Regression for Newspaper/Magazine Sales Forecasting. 17(1055-1062).

Zeng, Q.-w., Fu, A.-Y. & Xu, Z.-H., 2009. Application of Support Vector Regression and Particle Swarm Optimization in. *International Conference on Information Management, Innovation Management and Industrial Engineering.*

Zhao, S. & Wang, L., 2010. Support Vector Regression Based on Particle Swarm Optimization for Rainfall. *Third International Joint Conference on Computational Science and Optimization.*