TESIS - KS185411

OPTIMASI PARAMETER MODEL LEAST-SQUARE SUPPORT VECTOR REGRESSION MENGGUNAKAN GENETIC ALGORITHM DAN PARTICLE SWARM OPTIMIZATION

(Studi Kasus : Beban Listrik Jangka Pendek Area Jawa Timur)

KINANTI HANUGERA GUSTI
06211 8500 100 04

Dosen Pembimbing
Irhamah, M.Si, Ph.D
Dr.rer.pol. Heri Kuswanto, S.Si, M.Si

Program Magister
Departemen Statistika
Fakultas Sains dan Analitika Data
Institut Teknologi Sepuluh Nopember
Surabaya
2020

TESIS – KS185411

OPTIMASI PARAMETER MODEL LEAST-SQUARE SUPPORT VECTOR REGRESSION MENGGUNAKAN GENETIC ALGORITHM DAN PARTICLE SWARM OPTIMIZATION

(Studi Kasus : Beban Listrik Jangka Pendek Area Jawa Timur)

KINANTI HANUGERA GUSTI
06211 8500 100 04

Dosen Pembimbing
Irhamah, M.Si, Ph.D
Dr.rer.pol. Heri Kuswanto, S.Si, M.Si

Program Magister
Departemen Statistika
Fakultas Sains dan Analitika Data
Institut Teknologi Sepuluh Nopember
Surabaya
2020

TESIS – KS185411

OPTIMIZATION THE PARAMETER OF LEAST-SQUARE SUPPORT VECTOR REGRESSION USING GENETIC ALGORITHM AND PARTICLE SWARM OPTIMIZATION

(Study Case : Short Term Electricity Load in East Java)

KINANTI HANUGERA GUSTI
06211 8500 100 04

Supervisors
Irhamah, M.Si, Ph.D
Dr.rer.pol. Heri Kuswanto, S.Si, M.Si

Magister Program
Department of Statistics
Faculty of Science and Data Analytics
Institut Teknologi Sepuluh Nopember
Surabaya
2020

LEMBAR PENGESAHAN

Tesis disusun untuk memenuhi salah satu syarat memperoleh gelar

Magister Statistik (M.Stat)

di

Institut Teknologi Sepuluh Nopember

oleh

KINANTI HANUGERA GUSTI

NRP : 06211850010004

Tanggal Ujian: 27 Juli 2020

Periode Wisuda: September 2020

disetujui oleh:

Pembimbing:

1. Irhamah, M.Si, Ph.D
   NIP. 19780406 200112 2 002 .....

2. Dr.rer.pol Heri Kuswanto, M.Si.
   NIP. 19820326 200312 1 004 .....

Penguji:

3. Prof. NUR Iriawan, M.Ikom, Ph.D
   NIP. 19621015 198803 1 002 .....

4. Santi Wulan Purnami, M.Si, Ph.D
   NIP. 19720923 199803 2 001 .....

KEMENTERIAN PENDIDIKAN DAN KEBUDAYAAN INSTITUT TEKNOLOGI SEPULUH NOPEMBER FAKULTAS SAINS DAN ANALITIKA DATA DEPARTEMEN STATISTIKA

Kepala Departemen Statistika
Fakultas Sains dan Analitika Data

Dr. Dra. Kartika Fithriasari, M.Si.
NIP: 196912121993032 002

OPTIMASI PARAMETER MODEL LEAST-SQUARE SUPPORT VECTOR REGRESSION MENGGUNAKAN GENETIC ALGORITHM DAN PARTICLE SWARM OPTIMIZATION

(Studi kasus : Beban listrik jangka pendek area Jawa Timur)

Nama Mahasiswa	:	Kinanti Hanugera Gusti
NRP	:	062118 5001 0004
Dosen Pembimbing	:	Irhamah, M.Si, Ph.D
	:	Dr.rer.pol. Heri Kuswanto, S.Si, M.Si

ABSTRAK

Akurasi pada analisis deret waktu merupakan hal terpenting dalam peramalan dimana pemodelan yang didapat bergantung dari pola data historis, linier ataupun nonlinier. Pendekatan linier dilakukan dengan menggunakan metode ARIMA. Pendekatan nonlinier menggunakan SVR dan LSSVR yang diketahui memiliki fungsi kernel sehingga dapat menangkap pola nonlineritas data, tetapi komputasi LSSVR cenderung lebih cepat sebab membutuhkan penyelesaian yang lebih sederhana tanpa menghilangkan prinsip SVR. Peramalan dilakukan dengan mengunakan lag signifikan berdasarkan ARIMA sebagai input variabel. Pemilihan parameter sangat mempengaruhi hasil akurasi, sehingga dibutuhkan metode optimasi. Metode optimasi *grid search* yang biasa dilakukan tidak menjamin mencapai solusi optimum dan yang lebih penting tidak efisien dalam beberapa aplikasi praktis. Sehingga diperlukan metode optimasi metaheuristik, diantaranya GA dan PSO yang mampu mencari kemungkinan keseluruhan ruang pencarian solusi. Optimasi PSO lebih mudah diaplikasikan tetapi rentan terhadap konvergensi prematur yang menyebabkan dapat terjebak di lokal minimum, sehingga akan diperbaiki dengan Modifikasi PSO yang selalu mencari kembali posisi optimal sesuai kriteria yang ditentukan. Hasil dalam penelitian ini yaitu akurasi pendekatan nonliner jauh lebih baik dibandingkan dengan pendekatan linier. Penambahan metode optimasi pada SVR memberi perubahan yang berarti dibandingkan dengan LSSVR. Sedangkan GA merupakan metode optimasi terbaik karena menghasilkan nilai RMSE yang lebih rendah.

Kata Kunci: Beban Listrik, GA, LSSVR, Optimasi, PSO

(Halaman ini sengaja dikosongkan)

OPTIMIZATION THE PARAMETER OF LEAST-SQUARE SUPPORT VECTOR REGRESSION USING GENETIC ALGORITHM AND PARTICLE SWARM OPTIMIZATION

(Case study : Short Term Electricity Load in East Java)

Name of Student	:	Kinanti Hanugera Gusti
ID	:	062118 5001 0004
Supervisor	:	Irhamah, M.Si, Ph.D
Co-Supervisor	:	Dr.rer.pol. Heri Kuswanto, S.Si, M.Si

ABSTRACT

Accuracy is the most important thing in time series analysis for forecasting where the obtained model depends on historical, linear or nonlinear patterns. The linear approach is carried out using ARIMA method. Nonlinear modelling uses SVR and LSSVR which are known to have kernel functions so that they can capture patterns of nonlinearity data, but LSSVR computation tends to be faster because it requires a simpler solution without eliminating the SVR principle. Forecasting is done by using a significant lag based on ARIMA as the input variable. Selection of parameters greatly affects the result of accuracy, so an optimization method is needed. The usual, grid search optimization method does not guarantee reaching the optimum solution and not efficient in some practical applications. So, metaheuristic optimization methods is needed, including GA and PSO that are able to find the entire possible space for the solutions. PSO optimization is easier to apply but prone to premature convergence which causes it to be trapped in minimum locale, so it will be fixed by Modification of PSO which always looks back for the optimal position according to the specified criteria. The result in this study is the accuracy of nonlinear approach is better than the linear approach. The addition of an optimization method to the SVR gives a significant change compared to LSSVR. Meanwhile, GA is the best optimization method because produces a lower RMSE value.

Keywords: Electricity Load, GA, Optimization, LSSVR, PSO

(Halaman ini sengaja dikosongkan)

KATA PENGANTAR

Segala puji dan syukur bagi kehadirat Allah SWT yang telah melimpahkan rahmat, taufik beserta hidayah-Nya kepada penulis, sehingga penulis dapat menyelesaikan tesis yang berjudul "Optimasi Parameter Model Least-Square Support Vector Regression Menggunakan Genetic Algorithm dan Particle Swarm Optimization" ini dapat diselesaikan pada waktu yang tepat.

Penulisan tesis ini tidak akan berjalan dengan lancar tanpa bantuan dan dukungan beberapa pihak, oleh karena itu pada kesempatan ini penulis ingin mengucapkan terimakasih kepada:

1. Ibu Dr. Kartika Fithriasari, M.Si. selaku Kepala Departemen Statistika Fakultas Sains dan Analitika Data, Institut Teknologi Sepuluh Nopember, Surabaya.
2. Ibu Santi Wulan Purnami, M.Si., Ph.D. selaku Sekretaris Departemen I (Bidang Akademik, Kemahasiswaan, Penelitian, dan Pengabdian Kepada Masyarakat).
3. Ibu Dr. Vita Ratnasari, M.Si. selaku Sekretaris Departemen II (Bidang Sumber Daya Keuangan, Sumber Daya Manusia, dan Sarana Prasarana).
4. Bapak Dr.rer.pol. Dedy Dwi Prasetyo, S.Si., M.Si. selaku Kepala Program Studi Pascasarjana Statistika, Departemen Statistika, Fakultas Sains dan Analitika Data, Institut Teknologi Sepuluh Nopember, Surabaya.
5. Ibu Irhamah, M.Si., Ph.D. dan Bapak Dr.rer.pol. Heri Kuswanto, M.Si. selaku dosen pembimbing tesis yang telah meluangkan banyak waktu untuk membimbing penulis dengan penuh kesabaran, memberikan masukan, serta arahan demi terselesaikannya tesis ini.
6. Bapak Prof. Nur Iriawan, M.Ikom, Ph.D. dan Ibu Santi Wulan Purnami, M.Si., Ph.D. selaku penguji yang memberikan masukan dan saran pada penyelesaian tesis ini.
7. Seluruh dosen Departemen Statistika ITS yang telah memberikan ilmu selama perkuliahan dan karyawan yang telah membantu dalam menyelenggarakan perkuliahan.

8. Kedua orang tua dan keluarga penulis yang telah memberikan dukungan baik secara moril dan materiil sehingga penulis dapat menyelesaikan tesis ini dengan baik.
9. Kamu yang telah memberikan semangat dan pelajaran hidup yang sangat berharga, terimakasih.
10. Teman-teman S2 Statistika 2018 periode ganjil yang telah berjuang bersama-sama untuk menyelesaikan studi magister.
11. Semua teman, relasi, dan berbagai pihak yang tidak bisa penulis sebutkan semuanya satu per satu yang telah membantu dalam penulisan laporan ini.

Penulis menyadari bahwa tesis ini jauh dari kata sempurna. Oleh karena itu, kritik dan saran diharapkan dari semua pihak untuk tahap pengembangan selanjutnya. Besar harapan penulis bahwa informasi sekecil apapun dalam tesis ini bermanfaat bagi semua pihak dan dapat menambah wawasan serta pengetahuan.

Surabaya, Agustus 2020

Penulis

DAFTAR ISI

ABSTRAK ..... **ii**
ABSTRACT ..... **iv**
KATA PENGANTAR ..... **vi**
DAFTAR ISI ..... **viii**
DAFTAR TABEL ..... **x**
DAFTAR GAMBAR ..... **xii**
DAFTAR LAMPIRAN ..... **xiv**
1. BAB 1 ..... **1**
1.1 Latar Belakang 1
1.2 Rumusan Masalah 6
1.3 Tujuan Penelitian 6
1.4 Manfaat Penelitian 7
1.5 Batasan Masalah 7
2. BAB 2 ..... **9**
2.1 *Auto Regressive Integrated Moving Average* (ARIMA) 9
2.1.1 Tahap Identifikasi 9
2.1.2 Tahap Estimasi Parameter 10
2.2 *Double Seasonal* ARIMA (DSARIMA) 13
2.3 *Support Vector Regression* (SVR) 14
2.4 *Least Square Support Vector Regression* (LSSVR) 19
2.5 *Genetic Algorithm* (GA) 21
2.6 *Particle Swarm Optimization* (PSO) 23
2.7 *Modified Particle Swarm Optimization* (MPSO) 25
2.8 Kriteria Kebaikan Model 26
2.9 Beban Listrik 27
3. BAB 3 ..... **29**
3.1 Sumber Data 29
3.2 Variabel Penelitian 29
3.3 Langkah-Langkah Penelitian 30
4. BAB IV **41**
4.1 Kajian Algoritma LSSVR dengan Optimasi GA, PSO, dan MPSO 41
A. Desain Algoritma LSSVR dengan Optimasi GA, PSO, dan MPSO 41

Algoritma 1. LSSVR 41
Algoritma 2. LSSVR-GA 42
Algoritma 3. LSSVR-PSO 44
Algoritma 4. LSSVR-Modified PSO 45
B. Illustrasi Algoritma LSSVR dengan Optimasi GA dan PSO 47
4.2 Analisis Karakteristik Beban Listrik Jangka Pendek Area Jawa Timur. 61
4.3 Estimasi Parameter ARIMA 69
4.4 Optimasi *Grid Search* 73
4.5 Optimasi Genetic Algorithm 75
4.6 Optimasi Particle Swarm Optimization 81
4.7 Optimasi Modified Particle Swarm Optimization 85
4.8 Perbandingan Algoritma 88
BAB V 91
5.1 Kesimpulan 91
5.2 Saran 91
DAFTAR PUSTAKA 93
LAMPIRAN 97

DAFTAR TABEL

Judul Tabel Halaman

Tabel 2.1 Kriteria Model ARIMA Berdasarkan Plot ACF dan PACF 10
Tabel 3.1 Struktur Data Penelitian 29
Tabel 4.1. Karakteristik Data Bangkitan AR(1) 47
Tabel 4.2 Uji Signifikansi Model ARIMA Data Bangkitan 48
Tabel 4.3 Ilustrasi Pembentukan Kromosom 51
Tabel 4.4 Ilustrasi Proses *Roulette Wheel* 52
Tabel 4.5 Ilustrasi Populasi Awal PSO 56
Tabel 4.6 Hasil Nilai RMSE pada Simulasi *Inertia Weight* 57
Tabel 4.7 Ringkasan Hasil Performa LSSVR pada Data AR(1) 61
Tabel 4.8 Nilai Estimasi Parameter Model DSARIMA 72
Tabel 4.9 Perbandingan Akurasi Optimasi Grid Search 74
Tabel 4.10 Pembentukan Kromosom 76
Tabel 4.11 Proses *Roulette Wheel* 77
Tabel 4.12 Perbandingan Akurasi Optimasi Genetic Algorithm 80
Tabel 4.13 Populasi Awal PSO 82
Tabel 4.14 Perbandingan Akurasi Optimasi Particle Swarm Optimization 84
Tabel 4.15 Perbandingan Akurasi Optimasi Modified Particle Swarm Optimization 86
Tabel 4.17 Tabel Perbandingan Akurasi dan Lama Komputasi Tiap Metode 89

(Halaman ini sengaja dikosongkan)

DAFTAR GAMBAR

Judul Gambar Halaman

Gambar 2.1 Ilustrasi Batas *Error* pada SVR 16
Gambar 2.2 *Sparseness* antara SVM dan LS-SVM untuk Regresi Linier 20
Gambar 3.1 Diagram Alir Penelitian Secara Umum 34
Gambar 3.2 Diagram Alir Pemodelan ARIMA 35
Gambar 3.3 Diagram Alir Peramalan SVR 36
Gambar 3.4 Diagram Alir Peramalan LSSVR 37
Gambar 3.5 Diagram Alir Peramalan SVR-GA dan LSSVR-GA 38
Gambar 3.6 Diagram Alir Peramalan SVR-PSO dan LSSVR-PSO 39
Gambar 3.7 Diagram Alir Peramalan SVR-MPSO dan LSSVR-MPSO 40
Gambar 4.1 Pola Data AR(1) 48
Gambar 4.2 Plot ACF dan PACF Data AR(1) 48
Gambar 4.3 Plot *Actual* vs *Predicted* LSSVR Data Testing AR(1) 50
Gambar 4.4 Ilustrasi Bentuk Kromosom 51
Gambar 4.5 Ilustrasi *Crossover* Kromosom 52
Gambar 4.6 Ilustrasi Mutasi Uniform (kiri sebelum mutasi, kanan sesudah mutasi) 53
Gambar 4.7 Performansi *Genetic Algorithm* Data AR(1) 54
Gambar 4.8 Plot *Actual* vs *Predicted* LSSVR-*Genetic Algorithm* Data Testing AR(1) 55
Gambar 4.9 Ilustrasi Pergerakkan Partikel pada Iterasi kedua Algoritma PSO 57
Gambar 4.10 Performa Algoritma PSO untuk Data AR(1) 58
Gambar 4.11 Plot *Actual* vs *Predicted* LSSVR-*Particle Swarm Optimization* Data Testing AR(1) 58
Gambar 4.12 Performa Algoritma MPSO untuk Data AR(1) 60
Gambar 4.13 Plot *Actual* vs *Predicted* LSSVR-*Modified Particle Swarm Optimization* Data Testing AR(1) 60
Gambar 4.14 Time Series Plot Beban Konsumsi Listrik 62
Gambar 4.15 Pola Konsumsi Beban Listrik Per Setengah Jam 64
Gambar 4.16 Plot Rata-Rata dan Varians Beban Listrik Per Setengah Jam 65
Gambar 4.17 Boxplot Beban Listrik Per Setengah Jam 66
Gambar 4.18 Plot Musiman Beban Listrik Regional Jawa Timur 66
Gambar 4.19 Time Series Plot Konsumsi Beban Listrik Per Hari 67
Gambar 4.20 Plot Rata-Rata dan Varians Konsumsi Beban Listrik Per Hari 68
Gambar 4.21 Plot ACF dan PACF Data Beban Listrik 69
Gambar 4.22 Plot ACF dan PACF Data Beban Listrik *Differencing* 1 70
Gambar 4.23 Plot ACF dan PACF Data Beban Listrik *Differencing* 48 70
Gambar 4.24 Plot ACF dan PACF PACF Data Beban Listrik *Differencing* 336 71
Gambar 4.25 Plot *Actual* vs *Predicted* DSARIMA 73

Gambar 4.26 Plot *Actual* vs *Predicted* SVR dan LSSVR..... 75
Gambar 4.27 Bentuk Kromosom 76
Gambar 4.28 Proses *Crossover Singlepoint* Kromosom..... 78
Gambar 4.29 Mutasi Uniform Kromosom (kiri sebelum mutasi, kanan sesudah mutasi)..... 78
Gambar 4.30 Performansi Algoritma *Genetic Algorithm* LSSVR..... 79
Gambar 4.31 Plot *Actual* vs *Predicted* SVR-GA dan LSSVR-GA..... 80
Gambar 4.32 Pergerakan Partikel pada Iterasi Kedua Algoritma PSO..... 83
Gambar 4.33 Perfomansi Algoritma PSO LSSVR 83
Gambar 4.34 Plot *Actual* vs *Predicted* SVR-PSO dan LSSVR-PSO..... 84
Gambar 4.35 Performansi Algoritma MPSO LSSVR 87
Gambar 4.36 Plot *Actual* vs *Predicted* SVR-MPSO dan LSSVR-MPSO 87
Gambar 46. Perbandingan Plot Hasil Estimasi pada Model SVR 89
Gambar 47. Perbandingan Plot Hasil Estimasi pada Model LSSVR..... 90

DAFTAR LAMPIRAN

Judul Lampiran Halaman

Lampiran 1. Model DSARIMA 97
Lampiran 2. Syntax Pemodelan DSARIMA 98
Lampiran 3. Syntax Training Testing Data R 99
Lampiran 4. Syntax SVR 100
Lampiran 5. Syntax SVR-GA 101
Lampiran 6. Syntax SVR-PSO 102
Lampiran 7. Syntax SVR-MPSO 105
Lampiran 8. Syntax Training Testing Data Matlab 109
Lampiran 9. Syntax LSSVR 110
Lampiran 10. Syntax LSSVR-GA 111
Lampiran 11. Syntax LSSVR-PSO 112
Lampiran 12. Syntax LSSVR-MPSO 115

(Halaman ini sengaja dikosongkan)

BAB 1
PENDAHULUAN

1.1 Latar Belakang

Peramalan data runtun waktu (*time series forecasting*) merupakan salah satu metode statistika yang didasarkan penggunaan pola hubungan variabel yang akan dicari (*dependent*) dengan variabel yang mempengaruhinya (*independent*) dan dikaitkan dengan selang waktu tertentu. Tujuan dari metode ini adalah menemukan pola deret historis dan mengekstrapolasikan pola tersebut ke masa depan sehinggga hasilnya dapat dijadikan acuan untuk peramalan nilai di masa yang akan datang (Purba, 2015). Hal terpenting dalam time series forecasting yakni keakuratan model yang menjadi pokok dari proses pengambilan keputusan, sedangkan model yang terbentuk bergantung dari pola data historis, baik itu pola linier ataupun pola nonlinier.

Metode time series forecasting terbagi menjadi dua bagian. Pertama, model peramalan yang didasarkan pada model matematika statistik seperti *moving average*, *exponential smoothing*, dan ARIMA (*Autoregressive Integrated Moving Average*). Kedua, model peramalan yang didasarkan pada kecerdasan buatan seperti *Neural Network* dan *Support Vector Regression*. Dalam penelitian ini yang akan dibahas ialah metode ARIMA dan *Support Vector Regression*.

Metode ARIMA terdiri dari dua komponen yaitu model *autoregressive* dan model *moving average*. Model *Autoregressive* memodelkan autokorelasi variabel deret waktu yang tergantung secara linier pada nilai-nilai variabel sebelumnya, sedangkan model *moving average* memodelkan autokorelasi dari kesalahan sebelumnya yang terdapat pada deret waktu. Metode ARIMA dikenal memiliki ketepatan yang sangat baik untuk peramalan jangka pendek dan untuk data deret waktu nonstasioner pada saat linier, tetapi pada saat peramalan data dilakukan dalam periode yang cukup panjang ketepatannya cenderung menjadi kurang baik karena akan cenderung *flat* bekisar pada nilai rata-ratanya (Wiyanti & Pulungan, 2012). Meskipun model ARIMA cukup fleksibel, untuk identifikasi model yang lebih kompleks memerlukan pengalaman yang

lebih dan asumsi linier dari data yang dimodelkan dianggap tidak sesuai untuk pemodelan deret waktu nonlinier yang kompleks (Zhang, 2003).

Metode dengan pendekatan kecerdasan buatan dinilai lebih fleksibel dibandingkan dengan model matematik statistik. Pendekatan kecerdasan buatan, diantaranya yaitu NN dan SVR, mampu menangkap pola yang cenderung tidak teratur dan tidak stabil atau pola nonlinier. Pendekatan menggunakan NN dan SVR memiliki kemampuan untuk meramalkan secara akurat pada data yang memiliki pola nonlinier dibandingkan dengan model ARIMA. Menurut Amin dan Hoque (2019) metode NN dinilai tidak dapat menghasilkan hasil yang lebih akurat dibandingkan dengan SVR sebab estimasi parameter didapat dari prinsip *empirical risk minimization*, sedangkan SVR menggunakan prinsip *structural risk minimization* dengan meminimalkan batas dari *generalization error* sehingga dapat mengatasi adanya *overfitting*. Jika dibandingkan dengan NN, metode SVR mampu memecahkan masalah yang melibatkan fitur dari ukuran sampel kecil, fitur dimensi tinggi, dan minimum lokal dengan kemampuan generalisasi yang baik. Metode SVR juga diketahui memiliki fungsi kernel yang dapat mengatasi sifat nonlinieritas data.

Penyelesaian dengan metode SVR dinilai memiliki komputasi yang cukup rumit, sehingga Suykens dan Vandewalle (1999) mengusulkan penyederhanaan SVR menggunakan *Least Square Support Vector Regression* (LSSVR) yang hanya membutuhkan penyelesaian satu set persamaan linier tanpa menghilangkan prinsip pada SVR yaitu meminimumkan risiko struktural. Berdasarkan Khosravi dan Jouybari-Moghaddam (2017), metode LSSVR diketahui memiliki kemampuan komputasi yang cepat dibandingkan dengan SVR, NN dan *Adaptive Neuro-Fuzzy Inference System* (ANFIS). Meskipun metode SVR dan LSSVR memiliki kelebihan dalam hal akurasi hasil peramalan dibandingkan dengan metode berbasis matematik statistik, tetapi kelebihan tersebut bergantung pada pemilihan nilai parameter optimal. Parameter SVR dan LSSVR dapat ditentukan dengan menggunakan *trial and error*, tetapi tidak mungkin dilakukan dikarenakan begitu banyak kombinasi dari nilai-nilai yang dapat digunakan bahkan tak berhingga nilainya sehingga diperlukan suatu teknik optimasi

dalam pemilihan nilai parameter yang tidak membutuhkan terlalu banyak percobaan dan membutuhkan waktu yang relatif singkat (Prangga, 2017).

Pendekatan yang biasa dilakukan dalam proses optimasi khususnya metode SVR dan LSSVR yaitu *grid search*. Ide dari algoritma *grid search* yaitu dengan membagi jangkauan parameter yang akan dioptimalkan kedalam grid dan melintasi semua titik untuk mendapatkan parameter yang optimal, tetapi harus dipandu oleh metrik kinerja yang biasanya diukur dengan *cross-validation* pada data (Septiningrum dkk, 2015). Kelemahan dari optimasi *grid search* yaitu jika parameter tambahan seperti tahapan *grid search* atau jumlah tingkatan sarang yang digunakan semakin banyak, komputasi yang digunakan akan semakin kompleks, selain itu tidak menjamin mencapai solusi optimum dan yang lebih penting tidak efisien dalam beberapa aplikasi praktis (Cortez, 2014). Hal tersebut sesuai dengan Pei-you dan Lu (2013) yang menyebut bahwa optimasi *grid search* tidak menjamin tercapainya solusi global optimal. Salah satu alternatif metode penyelesaian optimasi adalah dengan menggunakan metode metaheuristik seperti *Genetic Algorithm* (GA) dan *Particle Swarm Optimization* (PSO).

Metode metaheuristik menggunakan mekanisme meniru perilaku sosial ataupun strategi yang ada di alam untuk mencari solusi yang efektif dan komprehensif. Jika dibandingkan dengan teknik pengoptimalan konvensional, metaheuristik memiliki kemampuan superior untuk menghindari optimal lokal. Hal ini disebabkan oleh karakteristik stokastik metaheuristik yang mencari keseluruhan ruang pencarian secara ekstensif sehingga menghindari stagnansi solusi lokal (Pangestu dkk, 2017). Menurut Sivanandam dan Deepa (2008) kelebihan yang dimiliki oleh *Genetic Algorithm* yaitu sangat cocok digunakan untuk menyelesaikan masalah global optimum, fleksibel untuk diimplementasikan pada berbagai masalah dan ruang solusi lebih luas. Selain GA, terdapat algoritma PSO yang terinspirasi oleh sekawanan burung, sekumpulan ikan, atau sekawanan binatang dalam beradaptasi dengan lingkungan mereka, mencari makan, atau menghindari predator dengan cara menerapkan pendekatan “saling bertukar informasi” (Hassan *et al*, 2005). Berdasarkan (Hu *et al*, 2015) dibandingkan

dengan GA, algoritma PSO lebih mudah diterapkan dan tidak banyak parameter yang perlu ditentukan. Disamping kelebihan tersebut, PSO memiliki kekurangan yaitu dapat terjebak di lokal minimum atau konvergensi prematur. Konvergensi prematur pada PSO telah dicegah dengan berbagai cara, salah satunya pengaturan parameter inersia yang jika bernilai besar akan menghasilkan pencarian global dan sebaliknya akan menghasilkan pencarian lokal jika nilai inersia kecil. Ketika pengaturan parameter inersia tidak sesuai dapat menyebabkan konvergensi yang melambat, sebab tidak adanya keseimbangan global dan lokal. Berdasarkan Niu dan Dai (2017) terdapat perkembangan modifikasi PSO untuk menyeimbangkan antara pencarian global ataupun lokal yaitu dengan menambah parameter yang akan mencari kembali posisi partikel optimal sesuai dengan kriteria yang telah ditentukan. Hasil pada penelitian tersebut menunjukkan akurasi modifikasi PSO lebih baik dibandingkan dengan PSO.

Studi kasus yang digunakan dalam penelitian ini yaitu mengenai beban listrik. Salah satu jenis peramalan beban listrik yang mempunyai peran penting dalam *real-time control* dan fungsi-fungsi keamanan dari suatu sistem manajemen energi yaitu peramalan beban jangka pendek yang bertujuan untuk memprediksi besarnya konsumsi listrik harian per menit, jam, hari, atau minggu (Singh *et al*, 2017). Peramalan beban jangka pendek untuk jangka waktu satu hingga 24 jam kedepan berperan penting untuk operasi harian meliputi perencanaan produksi dan distribusi dari sebuah utilitas daya dan sistem tenaga listrik (Handoko, 2010). Karakteristik beban listrik pada hari tertentu seperti hari libur, hari libur berturut-turut dan hari-hari sebelum setelah liburan dalam kondisi beban anomali yang berbeda dengan hari-hari biasa, dimana karakteritistik tersebut juga bergantung pada klasifikasi beban. Seperti pada klasifikasi beban industri memiliki beban puncak yang cenderung tinggi di siang hari pada hari biasa, tetapi pada hari libur cenderung turun, dan sebaliknya pada beban rumah tangga beban puncak yang tercatat cenderung tinggi saat malam hari.

Adanya kondisi beban anomali menunjukkan karakteristik beban listrik merupakan salah satu contoh kasus seperti pada Wiyanti dan Pulungan (2012) yang menyebutkan bahwa sebagian besar data deret waktu dihasilkan oleh proses linier dan

nonlinier. Hal tersebut diperkuat oleh Wijaya (2017) bahwa pada beberapa daerah tertentu beban listrik memiliki akurasi terbaik saat dimodelkan kombinasi liniear dan nonlinier, dan didaerah lain terdapat beberapa metode linier yang mempunyai akurasi yang lebih baik daripada nonlinier. Begitu juga pada penelitian Fahmi dan Sofyan (2017) pemodelan beban listrik menggunakan kombinasi metode linier dan nonlinier menghasilkan akurasi yang lebih baik dibandingkan dengan pemodelan parsial linier ataupun nonlinier.

Penelitian terkait peramalan beban listrik mengenai optimasi dari SVR pernah dilakukan oleh Paramita dan Irhamah (2012) dengan menggunakan *hybrid* SVR-GA didapat akurasi peramalan beban listrik yang lebih baik dibandingkan dengan SVR. *Hybrid* SVR-PSO juga menghasilkan hasil akurasi yang lebih tinggi dibandingkan dengan SVR dan ANN (Trapsilasiwi & Sutijo, 2011). Sedangkan Wijaya (2017), melakukan penelitian tentang perbandingan antara SVR, *hybrid* SVR-GA, dan *hybrid* SVR-PSO didapat *hybrid* SVR-GA memiliki hasil peramalan dengan akurasi yang tertinggi disusul oleh SVR-PSO dan SVR. Akan tetapi, penelitian terkait peramalan beban listrik menggunakan LSSVR masih jarang ditemui. Berdasarkan (Yan, 2013) pada kasus peramalan harga beban listrik jangka menengah, metode LSSVR diketahui memiliki akurasi yang lebih rendah dibandingkan SVR, tetapi hasil akurasi pada pembentukan model LSSVR lebih baik daripada SVR, sehingga diperlukan optimasi model LSSVR agar didapatkan hasil peramalan yang lebih akurat. Adapun penelitian mengenai optimasi LSSVR dilakukan oleh Razak *et al* (2017) pada kasus peramalan harga beban listrik jangka menengah menggunakan optimasi GA dan didapatkan hasil metode *hybrid* LSSVR-GA mempunyai akurasi yang lebih baik dibandingkan SVR. Selain itu, optimasi LSSVR menggunakan PSO pada (Min & Huanqi, 2011) mengenai peramalan beban listrik jangka pendek menghasilkan akurasi yang lebih baik dibandingkan LSSVR. Berdasarkan uraian yang telah dijelaskan sebelumnya, pada penelitian ini akan dilakukan perbandingan antara metode SVR dan LSSVR dengan ketiga metode optimasi yaitu GA, PSO, dan MPSO sehingga akan didapatkan model

peramalan beban listrik jangka pendek. Pemilihan model terbaik akan ditentukan berdasarkan kriteria nilai kebaikan model yang didapat dari metode tersebut.

1.2 Rumusan Masalah

Tingkat akurasi merupakan isu utama dalam peramalan, dimana model yang dihasilkan bergantung dari karakteristik deret waktu, baik linier atau nonlinier. Metode nonlinier, salah satunya SVR memiliki akurasi yang lebih tinggi dibandingkan dengan metode linier, sebab SVR diketahui memiliki fungsi kernel sehingga penggunannya lebih fleksibel. Penyelesaian SVR dinilai memiliki komputasi yang relatif cukup rumit sehingga diperbarui oleh metode LSSVR yang menggunakan solusi penyelesaian satu set persamaan linier tanpa menghilangkan prinsip pada SVR yang membuat komputasi LSSVR lebih mudah dan lebih cepat. Adapun hasil akurasi dari metode SVR dan LSSVR bergantung dari pemilihan parameter, jika ditentukan mengunakan proses *trial and error* dan/atau *grid search* tidak menjamin mencapai solusi optimum dan yang lebih penting tidak efisien dalam beberapa aplikasi praktis, sehingga akan digunakan metode optimasi GA dan PSO agar didapat hasil peramalan yang optimal.

1.3 Tujuan Penelitian

Berdasarkan rumusan masalah pada uraian sebelumnya, maka tujuan penulisan penelitian ini yang ingin dicapai adalah sebagai berikut:

1. Mengkaji metode optimasi GA, PSO, dan MPSO pada LSSVR untuk mendapatkan parameter optimum pada data simulasi.
2. Mendapatkan model peramalan beban listrik terbaik di Jawa Timur berdasarkan model ARIMA, SVR, dan LSSVR
3. Mendapatkan hasil optimasi terbaik untuk model peramalan beban listrik tiap kota di Jawa Timur berdasarkan model *hybrid* SVR-GA, SVR-PSO, SVR-MPSO, LSSVR-GA, LSSVR-PSO, dan LSSVR-MPSO.

1.4 Manfaat Penelitian

Melalui penelitian ini diharapkan dapat memberi manfaat yaitu peneliti maupun pembaca dapat menambah dan wawasan dari metode berbasis kecerdasan buatan seperti SVR dan LSSVR terutama pada studi kasus data deret waktu, menambah keilmuan mengenai optimasi GA dan PSO dalam penentuan input parameter, serta memberikan saran dan perbandingan untuk pengembangan metode dari penelitian yang akan dilakukan selanjutnya.

1.5 Batasan Masalah

Adapun batasan masalah yang perlu diperhatikan dalam penelitian ini adalah sebagai berikut:

1. Beban listrik yang digunakan merupakan beban listrik yang tercatat pada trafo distribusi 20kV
2. Pemodelan ARIMA digunakan untuk mencari lag-lag yang signifikan.
3. *Support Vector* yang digunakan pada metode SVR dan LSSVR merupakan lag-lag yang signifikan berdasarkan model ARIMA.
4. Fungsi kernel yang digunakan pada metode SVR dan LSSVR adalah RBF Gaussian.

(Halaman ini sengaja dikosongkan)

BAB 2
TINJAUAN PUSTAKA

2.1 *Auto Regressive Integrated Moving Average* (ARIMA)

Model ARIMA merupakan model *time series* univariat yang termasuk dalam kelompok linier dan digunakan pada data yang memiliki pola tren, musiman, ataupun non musiman. Terdapat prosedur Box-Jenkins yang bertujuan membentuk model ARIMA dengan pendekatan empirik yang bertujuan untuk identifikasi orde dari model ARIMA (p, d, q), estimasi dan uji signifikansi parameter, pemeriksaan diagnostik, menentukan model terbaik serta melakukan peramalan.

2.1.1 Tahap Identifikasi

Identifikasi model baru dapat dilakukan jika asumsi stasioneritas terpenuhi, sebab memerlukan nilai *autocorrelation* (ACF) dan *partial autocorrelation* (PACF) yang diperoleh ketika data tidak memperlihatkan adanya perubahan yang jelas dari waktu ke waktu (stasioner dalam varians) dan data berfluktuasi disekitar garis yang sejajar dengan sumbu waktu (*t*) atau stasioner dalam *mean*. Secara matematis dapat dituliskan nilai $E(\dot{Z}_t) = \mu$ dan $Var(\dot{Z}_t) = \sigma^2$. Prosedur menstasionerkan dilakukan dengan (1) membuat time series plot data dan memilih transformasi yang sesuai apabila data tidak stasioner dalam varians, (2) menghitung sampel ACF dan PACF untuk menguji stasioner dalam mean, jika plot ACF turun lambat maka data belum stasioner dalam mean sehingga perlu dilakukan proses *differencing*, (3) menghitung sampel ACF dan PACf dari data yang telah stasioner untuk menentukan orde *p* dan *q*. Petunjuk umum penentuan orde *p* dan *q* disajikan pada Tabel 2.1

Tabel 2.1 Kriteria Model ARIMA Berdasarkan Plot ACF dan PACF

Model	**ACF**	**PACF**
AR(*p*)	Turun cepat secara eksponensial atau sinusoidal	Terputus (signifikan) setelah lag ke-*p*
MA(*q*)	Terputus (signifikan) setelah lag ke-*q*	Turun cepat secara eksponensial atau sinusoidal
ARMA(*p*,*q*)	Turun cepat secara eksponensial atau sinusoidal	Turun cepat secara eksponensial atau sinusoidal
AR(*p*) atau MA(*q*)	Terputus (signifikan) setelah lag ke-*p*	Terputus (signifikan) setelah lag ke-*q*
White Noise	Tidak ada yang signifikan	Tidak ada yang signifikan

2.1.2 Tahap Estimasi Parameter

Setelah diperoleh model yang sekiranya cocok, selanjutnya yaitu melakukan estimasi parameter model. Metode yang digunakan untuk estimasi parameter model ARIMA estimasi *Conditional Least Square* (CLS). Prinsip dari metode CLS yaitu dengan meminimalkan jumlah kuadrat error.

Misalkan pada model $AR(1)$: $\dot{Z}_t - \mu = \phi_1\left(\dot{Z}_{t-1} - \mu\right) + a_t$ akan didapat fungsi *conditional* jumlah kuadrat error sebagai berikut

$$S_c(\mu,\phi_1) = \sum_{t=2}^{n} a_t^2 = \sum_{t=2}^{n}\left[\left(\dot{Z}_t - \mu\right) - \phi_1\left(\dot{Z}_{t-1} - \mu\right)\right]^2 \tag{2.1}$$

Selanjutnya, persamaan (2.1) akan diturunkan terhadap parameter μ dan ϕ_1

$$\frac{\partial S_c(\mu,\phi_1)}{\partial \mu} = \sum_{t=2}^{n} 2\left[\left(\dot{Z}_t - \mu\right) - \phi_1\left(\dot{Z}_{t-1} - \mu\right)\right]\left(-1 + \phi_1\right) = 0 \tag{2.2}$$

$$\frac{\partial S_c(\mu,\phi_1)}{\partial \phi_1} = \sum_{t=2}^{n} 2\left[\left(\dot{Z}_t - \mu\right) - \phi_1\left(\dot{Z}_{t-1} - \mu\right)\right]\left(-\dot{Z}_{t-1} + \mu\right) = 0 \tag{2.3}$$

Dengan menyelesaikan persamaan (2.2) dan (2.3) akan diperoleh penyelesaian sebagai berikut:

$$\hat{\mu} = \bar{\dot{Z}} \tag{2.4}$$

$$\hat{\phi}_1 = \frac{\sum_{t=2}^{n}\left(\dot{Z}_t - \bar{\dot{Z}}\right)\left(\dot{Z}_{t-1} - \bar{\dot{Z}}\right)}{\sum_{t=2}^{n}\left(\dot{Z}_t - \bar{\dot{Z}}\right)^2} \tag{2.5}$$

Pada model MA, tidak mudah mendapatkan estimator model dengan menggunakan CLS, sehingga model MA perlu direpresentasikan terlebih dahulu pada model AR. Misalkan pada model $MA(1) : \dot{Z}_t = a_t - \theta_1 a_{t-1}$ dapat direpresentasikan sebagai model AR tak hingga menjadi

$$\dot{Z}_t = -\theta_1\left(\dot{Z}_{t-1}\right) - \theta_1^2\left(\dot{Z}_{t-1}\right) - \theta_1^3\left(\dot{Z}_{t-1}\right) - \ldots + a_t \tag{2.6}$$

dan CLS dapat diaplikasikan dengan meminimumkan nilai θ sebagai berikut

$$S_c(\boldsymbol{\theta}) = \sum_{t=2}^{n} a_t^2 = \sum_{t=2}^{n}\left[\dot{Z}_t + \theta_1\dot{Z}_{t-1} + \theta_1^2\dot{Z}_{t-2} + \theta_1^3\dot{Z}_{t-3} + \ldots\right]^2 \tag{2.7}$$

Persamaan (2.7) merupakan persamaan dengan parameter tidak linier, sehingga meminimumkan $S_C(\boldsymbol{\theta})$ tidak diperkenankan. Sehingga model umum $MA(q)$ diperlukan algoritma optimasi numerik karena merupakan persamaan tidak linier dalam penyelesaian estimasi parameter.

Pada model ARMA, misal ARMA(1,1) proses estimasi dimulai dari menentukan error model yakni:

$$\begin{aligned} a_t &= \dot{Z}_t - \phi_1\dot{Z}_{t-1} + \theta_1 a_{t-1} \\ a_t &= \dot{Z}_t - \phi_1\dot{Z}_{t-1} + \theta_1\left(\dot{Z}_{t-1} - \phi_1\dot{Z}_{t-2} + \theta_1 a_{t-2}\right) \end{aligned} \tag{2.8}$$

Sama halnya pada model MA, persamaan (2.8) tidak diperkenankan untuk mengestimasi parameter menggunakan metode CLS sebab didapat persamaan yang tidak linier dalam parameter. Sehingga model ARMA akan dilakukan prosedur estimasi non linier *least square*. Prosedur non linier *least square* dilakukan dengan teknik iterasi. Berikut merupakan persamaan utama dari non linier *least square* dengan menggunakan model regresi linier.

$$Z_t = E\left(Z_t \middle| X_{t_1}\text{'s}\right) + a_t \tag{2.9}$$

dimana $t=1,2,\ldots,n$ dan $a_t's$ adalah i.i.d $N(0,\sigma^2{}_a)$ independen untuk setiap X_{t_i}. Estimasi *least square*, baik linier atau nonlinier, dimana model linier merupakan kasus khusus dari model nonlinier, dapat dilakukan perhitungan menggunakan teknik iterasi dengan prosedur berikut:

a. Menentukan vektor inisial yaitu $\boldsymbol{\alpha}$, kemudian hitung residual $\boldsymbol{e} = (\boldsymbol{Z} - \overline{\boldsymbol{Z}})$ dan residual *sum of square*.

$$S(\boldsymbol{\alpha}) = \mathbf{e}'\mathbf{e} = \left(\mathbf{Z} - \overline{\mathbf{Z}}\right)'\left(\mathbf{Z} - \overline{\mathbf{Z}}\right) \tag{2.10}$$

dimana $\boldsymbol{y} = f(\boldsymbol{\alpha})$ adalah vektor prediksi yang mengganti nilai parameter yang tidak diketahui dengan suatu hasil prediksi. Model $f(\boldsymbol{X_t}, \boldsymbol{\alpha})$ didekati menggunakan deret *taylor* orde pertama dengan ekspansi nilai inisial $\tilde{\alpha}$ sebagai berikut.

$$f(\alpha) = f(\tilde{\alpha}) + \overline{X}_{\tilde{\alpha}}\delta \tag{2.11}$$

dimana $\boldsymbol{\delta} = (\boldsymbol{\alpha} - \overline{\boldsymbol{\alpha}})$ dan $\overline{\boldsymbol{X}}_{\tilde{\boldsymbol{\alpha}}} = [X_{ij}]$ adalah matriks $n \times p$ yang merupakan turunan parsial dari $\tilde{\boldsymbol{\alpha}}$, kemudian hitung

$$\delta = \left(\overline{\mathbf{X}}_{\tilde{\alpha}}{}'\overline{\mathbf{X}}\right)^{-1}\overline{\mathbf{X}}_{\tilde{\alpha}}{}'\mathbf{e} = \left(\delta_1,\ldots,\delta_p\right) \tag{2.12}$$

untuk model nonlinier maka $\overline{\boldsymbol{X}}_{\tilde{\boldsymbol{\alpha}}}$ akan berubah disetiap iterasi.

b. Hitung estimasi *least square* terbaru

$$\hat{\alpha} = \tilde{\alpha} + \delta \tag{2.13}$$

dan hitung residual dari *sum square* $S(\hat{\boldsymbol{\alpha}})$. Pada model nonlinier langkah b menghasilkan nilai inisial baru untuk iterasi selanjutnya.

Jika diberikan model ARMA(*p*,*q*), maka untuk menyelesaikannya bisa menggunakan prosedur *nonlinear least square* yang meminimumkan *error* dari *sum square* $S(\emptyset, \mu, \boldsymbol{\theta})$. Iterasi akan terus dilakukan hingga kriteria konvergen terpenuhi yaitu mereduksi relatif dari *sum square*, dimana maksimum selisih nilai parameter dari setiap iterasi kurang dari setiap iterasi kurang dari suatu nilai yang telah ditentukan atau banyak iterasi lebih besar dari nilai yang ditetapkan (Wei, 2006).

Selanjutnya, model ARIMA yang telah didapat perlu dilakukan uji signifikansi parameter. Model yang baik adalah model yang memiliki estimasi parameter yang

signifikan. Uji signifikansi parameter dilakukan dengan menghitung stastistik uji *t* yang dapat dihitung menggunakan $t_{hitung} = \frac{\widehat{\beta}_i}{SE(\widehat{\beta}_i)}$ dan parameter akan signifikan ketika $|t_{hitung}| > t_{\alpha/2,(n-n_p)}$ dengan n : banyak pengamatan pada data *time series*, n_p : banyaknya parameter, dan β_i : parameter yang mencakup $\phi_i, \theta_i, \Phi_{1i}, \Phi_{2i}, \Theta_{1i}, \Theta_{2i}$

2.2 *Double Seasonal* ARIMA (DSARIMA)

ARIMA merupakan salah satu metode yang popular dalam analisis *time series* dan tersusun dari dua model utama, yaitu *Auto Regressive* (AR) dan *Moving Average* (MA). Model ARIMA (p, d, q) merupakan kombinasi dari model AR (p) dengan MA (q) pada data tidak stasioner yang kemudian dilakukan proses *differencing* pada orde d. Secara metamatis, model ARIMA (p, d, q) dituliskan pada persamaan berikut

$$\phi_p(B)(1-B)^d Z_t = \theta_q(B)a_t \tag{2.14}$$

Model ARIMA yang memiliki efek musiman pada waktu ke-s dapat dirumuskan seperti persamaan (2.15). Sementara moel *multiplicative seasonal* ARIMA yang merupakan kombinasi persamaan (2.14) dan (2.15) dirumuskan pada persamaan (2.16)

$$\Phi_P(B^s)(1-B^s)^D Z_t = \Theta_Q(B^s)a_t \tag{2.15}$$

$$\phi_p(B)\Phi_P(B^s)(1-B)^d(1-B^s)^D Z_t = \theta_q(B)\Theta_Q(B^s)a_t \tag{2.16}$$

dengan:

p : orde dari AR

q : orde dari MA

d : orde dari *differencing*

P : orde dari *seasonal* AR

Q : orde dari *seasonal* MA

D : orde dari *seasonal differencing*

s : periode *seasonal*

$\phi_p(B)$: $1 - \phi_1 B - \phi_2 B^2 - \cdots - \phi_p B^p$

$\theta_q(B)$: $1 - \theta_1 B - \theta_2 B^2 - \cdots - \theta_q B^q$

$\phi_P(B^s)$: $1 - \phi_1 B^s - \phi_2 B^{s2} - \cdots - \phi_P B^{Ps}$

$\theta_Q(B^s)$: $1 - \theta_1 B^s - \theta_2 B^{s2} - \cdots - \theta_Q B^{Qs}$

Data konsumsi listrik jangka pendek pada umumnya memiliki pola *double seasonal* (Khusna & Suhartono, 2015). Karena itu, model ARIMA yang cocok untuk peramalan listrik jangka pendek adalah model *multiplicative double seasonal* ARIMA atau model ARIMA (p, d, q) $(P_1, D_1, Q_1)^{s1}(P_2, D_2, Q_2)^{s2}$. Mengacu pada persamaan (2.16), secara matematis model *multiplicative double seasonal* ARIMA dapat ditulis sesuai dengan persamaan berikut

$$\phi_p(B)\Phi_{P_1}(B^{s_1})\Phi_{P_2}(B^{s_2})(1-B)^d(1-B^{s_1})^{D_1}(1-B^{s_2})^{D_2}Z_t = \theta_q(B)\Theta_{Q_1}(B^{s_1})\Theta_{Q_2}(B^{s_2})a_t \quad (2.16)$$

dengan:

P_1 : orde dari *seasonal* 1 AR

P_2 : orde dari *seasonal* 2 AR

Q_1 : orde dari *seasonal* 1 MA

Q_2 : orde dari *seasonal* 2 MA

D_1 : orde dari *seasonal* 1 *differencing*

D_2 : orde dari *seasonal* 2 *differencing*

s_1 : periode *seasonal* 1

s_2 : periode *seasonal* 2

$\Phi_{P_1}(B^{s_1})$: $1 - \Phi_{11}B^{s_1} - \Phi_{12}B^{2s_1} - \cdots - \Phi_{1P_1}B^{P_1 s_1}$

$\Phi_{P_2}(B^{s_2})$: $1 - \Phi_{21}B^{s_2} - \Phi_{22}B^{2s_2} - \cdots - \Phi_{2P_2}B^{P_2 s_2}$

$\Theta_{Q_1}(B^{s_1})$: $1 - \Theta_{11}B^{s_1} - \Theta_{12}B^{2s_1} - \cdots - \Theta_{1Q_1}B^{Q_1 s_1}$

$\Theta_{Q_2}(B^{s_2})$: $1 - \Theta_{21}B^{s_2} - \Theta_{22}B^{2s_2} - \cdots - \Theta_{2Q_2}B^{Q_2 s_2}$

2.3 *Support Vector Regression* (SVR)

Metode SVR merupakan metode yang dikembangkan dari *Support Vector Machine* (SVM). Estimasi fungsi pada SVM menggunakan prinsip *structural risk minimization* dengan meminimalkan batas dari *generalization error* diketahui lebih

baik daripada meminimalkan *error* dari proses *training* seperti pada ANN yang berisiko adanya *overfitting*. Tujuan dari SVR adalah untuk menemukan sebuah fungsi $f(x)$ yang memiliki paling banyak ε deviasi dari target sebenarnya (y_i) berdasarkan data *training* dan pada saat yang sama fungsi tersebut harus setipis mungkin atau *flateness*. Dengan kata lain, *error* tidak dipedulikan selama kurang dari ε (Smola & Schölkopf, 2004). Perumusan SVR berdasarkan permasalahan *Quadratic Programming* (QP) yang merupakan penyelesaian optimasi *convex*. Hal tersebut diperoleh dengan mengembangkan konsep *ε–insensitive loss function* pada SVM dan perumusan menggunakan *inequality* sebagai *constraint* dari permasalahan optimasi serta penggunaan kondisi *Mercer* untuk menghubungkan fitur nonlinear yang akan dipetakan ke ruang dimensi yang lebih tinggi menggunakan fungsi kernel.

Misal diberikan data *training* sebagai berikut:

$$D = \{(\mathbf{x}_1, y_1),\ldots,(\mathbf{x}_i, y_i),\ldots,(\mathbf{x}_T, y_T)\},\ \mathbf{x}_i \in R^n,\ y_i \in R \tag{2.17}$$

dengan fungsi nonlinier

$$f(\mathbf{x}) = \langle \boldsymbol{\omega}, \varphi(\mathbf{x}) \rangle + b \tag{2.18}$$

dimana $\langle \cdots \rangle$ menunjukkan *dot product*, $\boldsymbol{\omega} \in R^{n_k}$ merupakan *weight vector* pada ruang *primal weight*, $\varphi(.) = R^n \rightarrow R^{n_k}$ sebagai fungsi nonlinier yang memetakan ruang *input* pada ruang dengan dimensi tinggi, dan b merupakan bias. Besaran dimensi dari n_k secara implisit didefinisikan yang berarti dapat menjadi dimensi tak terbatas. Berdasarkan persamaan (2.18) agar didapat fungsi regresi yang setipis mungkin, maka solusi yang didapat yaitu dengan mengoptimasi fungsi berikut:

$$\min \frac{1}{2}\|\boldsymbol{\omega}\|^2 + C\sum_{k=1}^{T}\left(\xi_i + \xi_i^*\right) \tag{2.19}$$

dengan batasan:

$y_k - \langle \boldsymbol{\omega}, \varphi(\boldsymbol{x}_i) \rangle - b \leq \varepsilon + \xi_i$

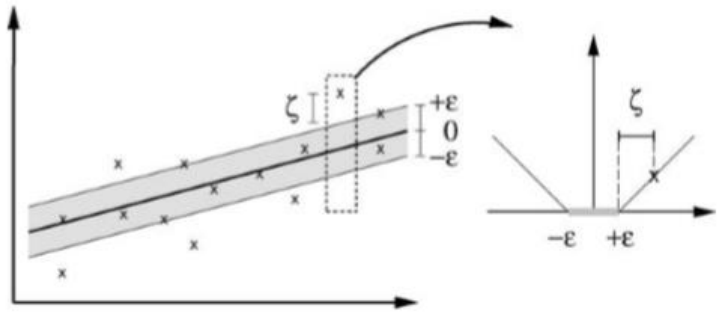
$\langle \boldsymbol{\omega}, \varphi(\boldsymbol{x}_i) \rangle + b - y_k \leq \varepsilon + \xi_i^*$

$\xi_i, \xi_i^* \geq 0$

dan *ε–insensitive loss function* didefinisikan sebagai berikut:

$$\left|y - f(\mathbf{x},\boldsymbol{\omega})\right|_{\varepsilon} = \begin{cases} 0, & \left|y - f(\mathbf{x},\boldsymbol{\omega})\right| \le \varepsilon \\ \left|y - f(\mathbf{x},\boldsymbol{\omega})\right| - \varepsilon, & \text{lainnya} \end{cases} \tag{2.20}$$

Fungsi f diasumsikan dapat mendekati semua titik (x_i, y_i) dengan presisi sebesar ε atau dapat dikatakan dalam keadaan *feasible*. Jika terjadi suatu kasus dimana ada beberapa titik (x_i, y_i) berada diluar batas ε-*tube*, kondisi tersebut dinamakan *infeasible*. Pada titik-titik *infeasible* akan ditambahkan variabel *slack* ξ dan ξ^* untuk mengatasi *infeasible constrain*. Konstanta $C > 0$ menentukan *trade-off* antara ketipisan dari fungsi f dan batas atas deviasi lebih besar dari ε yang masih ditoleransi. Semua deviasi yang lebih besar dari ε akan dikenai pinalti sebesar C. Ilustrasi batas ε ditunjukkan pada Gambar 2.1:



Gambar 2.1 Ilustrasi Batas *Error* pada SVR
(Smola dan Schölkopf, 2004)

Berdasarkan Gambar 2.1 ditunjukkan bahwa hanya pengamatan di luar daerah arsiran atau di luar batas ε diberi nilai konstanta C, begitu juga berapapun deviasi pada daerah arsiran akan diberi nilai nol. Hal tersebut menunjukkan bahwa permasalahan optimasi pada persamaan (2.19) dapat lebih mudah diselesaikan melalui *dual formulation*. Penyelesaian dari *dual formulation* yakni dengan membentuk *primal langrangian* dari fungsi objektif dan *constraint* fungsi tersebut.

$$L_{SVM} = \frac{1}{2}\|\boldsymbol{\omega}\|^2 + C\sum_{i=1}^{T}\left(\xi_i + \xi_i^*\right) - \sum_{k=1}^{T}\left(\eta_i\xi_i - \eta_i^*\xi_i^*\right) \tag{2.21}$$
$$- \sum_{i=1}^{T}\alpha_i\left(\varepsilon + \xi_i - y_i + \langle\boldsymbol{\omega}, \varphi(\mathbf{x}_i)\rangle + b\right)$$
$$- \sum_{i=1}^{T}\alpha_i^*\left(\varepsilon + \xi_i^* - y_i + \langle\boldsymbol{\omega}, \varphi(\mathbf{x}_i)\rangle + b\right)$$

dengan batasan $\alpha_i, \alpha_i^*, \eta_i, \eta_i^* \geq 0$ yang merupakan *lagrange multipliers*. Persamaan (2.21) kemudian diminimalkan pada variabel *primal* $(\boldsymbol{\omega}, b, \xi, \xi^*)$ yang ditampilkan pada persamaan berikut:

$$\frac{\partial L_{svr}}{\partial \boldsymbol{\omega}} = 0 \quad \rightarrow \quad \boldsymbol{\omega} = \sum_{k=1}^{T}\left(\alpha_i - \alpha_i^*\right)\varphi(\mathbf{x}_i) \tag{2.22}$$
$$\frac{\partial L_{svr}}{\partial b} = 0 \quad \rightarrow \quad \sum_{k=1}^{N}\left(\alpha_i^* - \alpha_i\right) = 0$$
$$\frac{\partial L_{svr}}{\partial \xi_i} = 0 \quad \rightarrow \quad C - \alpha_i - \eta_i = 0$$
$$\frac{\partial L_{svr}}{\partial \xi_i^*} = 0 \quad \rightarrow \quad C - \alpha_i^* - \eta_i^* = 0$$

Optimasi *dual langrangian* didapat setelah mensubstitusikan persamaan (2.22) pada (2.21) sehingga persamaan optimasi tersebut dapat dituliskan sebagai

$$\max_{\alpha,\alpha^*} Q = -\frac{1}{2}\sum_{i,l=1}^{T}\left(\alpha_i - \alpha_i^*\right)\left(\alpha_i - \alpha_l^*\right)\langle\varphi(\mathbf{x}_i), \varphi(\mathbf{x}_l)\rangle \tag{2.23}$$
$$- \varepsilon\sum_{i=1}^{T}\left(\alpha_i - \alpha_i^*\right) + \sum_{i=1}^{T} y_i\left(\alpha_i - \alpha_i^*\right)$$

dengan batasan $\sum_{k=1}^{T}(\alpha_i - \alpha_i^*) = 0$ dan $\alpha_i, \alpha_i^* \epsilon [0, C]$

Adapun parameter b pada persamaan (2.18) dapat dihitung berdasarkan kondisi KKT yang menyatakan solusi optimal akan didapat ketika perkalian antara *dual* variabel dan *constraint* harus dihilangkan. Sehingga akan dilakukan proses sebagai berikut:

$$\alpha_i\left[\varepsilon + \xi_i - y_i + \langle\boldsymbol{\omega}, \varphi(\mathbf{x}_i)\rangle + b\right] = 0 \tag{2.24}$$
$$\alpha_i^*\left[\varepsilon + \xi_i^* - y_i + \langle\boldsymbol{\omega}, \varphi(\mathbf{x}_i)\rangle - b\right] = 0$$

$$\eta_i \xi_i = (C - \alpha_i)\xi_i = 0 \qquad (2.25)$$
$$\eta_i^* \xi_i^* = (C - \alpha_i^*)\xi_i^* = 0$$

Jika memenuhi kondisi $|f(\boldsymbol{x}, \boldsymbol{\omega}) - y| \geq \varepsilon$, maka akan didapat nilai *lagrange multipliers* pada persamaan (2.24) tidak nol, atau jika semua data *training* masuk kedalam ε-*tube* (seperti pada Gambar 2.1) maka α_i, α_i^* akan hilang. Semua pengamatan yang berada dalam ε-*tube* disebut sebagai *Support Vectors* (SVs). Berdasarkan persamaan (2.24) dan (2.25), ada beberapa kesimpulan yang didapat: (1) sampel (x_i, y_i) yang berlaku $\alpha_i, \alpha_i^* = C$ maka akan jatuh diluar ε-*tube* yang disebut sebagai *Boundary Support Vectors* (BSVs), (2) $\alpha_i \alpha_i^* = 0$, tidak pernah ada satu set dari dual variabel α_i, α_i^* yang keduanya bernilai nol karena membutuhkan variabel *slacks* yang bernilai tidak nol, (3) untuk *support vectors* yang memenuhi $\alpha_i, \alpha_i^* \in (0, C)$ dan $\xi_i, \xi_i^* \in 0$ disebut sebagai *Normal Support Vectors* (NSVs). Adapun b dapat dihitung dengan rumus sebagai berikut:

$$b = y_i - \langle \boldsymbol{\omega}, \varphi(\mathbf{x}_i) \rangle - \varepsilon \text{ untuk } \alpha_i \in (0, C) \qquad (2.26)$$
$$b = y_i - \langle \boldsymbol{\omega}, \varphi(\mathbf{x}_i) \rangle + \varepsilon \text{ untuk } \alpha_i^* \in (0, C)$$

Sehingga, estimasi dari fungsi nonlinier SVM dapat dituliskan menjadi

$$f(\mathbf{x}) = \sum_{i=1}^{T} (\alpha_i - \alpha_i^*) \langle \varphi(\mathbf{x}), \varphi(\mathbf{x}_i) \rangle + b \qquad (2.27)$$

Sebagaimana kondisi Mercer's, *inner prodct* $\langle \varphi(\boldsymbol{x}), \varphi(\boldsymbol{x}_i) \rangle$ dapat didefinisikan sebagai fungsi kernel $K(\boldsymbol{x}, \boldsymbol{x}_i)$, sehingga persamaan (2.27) dapat dijabarkan menjadi persamaan (2.28):

$$f(\mathbf{x}) = \sum_{k=1}^{T} (\alpha_i - \alpha_i^*) K(\mathbf{x}, \mathbf{x}_i) + b \qquad (2.28)$$

Salah satu dari sekian fungsi kernel yang paling sering digunakan adalah *Gaussian Radial Basis Function* (RBF) seperti pada persamaan (2.29):

$$K(\mathbf{x}, \mathbf{x}_i) = \exp\left(\|\mathbf{x} - \mathbf{x}_i\|^2 / 2\sigma^2 \right) \qquad (2.29)$$

(Wang dan Hu, 2006)

2.4 *Least Square Support Vector Regression* (LSSVR)

Metode *Least Square Support Vector Regression* (LSSVR) merupakan pengembangan dari SVR yang memodifikasi *loss function* menjadi *quadratic loss function* dan *constraint* dari *inequality* menjadi *equality* tanpa menghilangkan prinsip utama dari SVR, meminimumkan *risk function*. Adanya *equality* pada SVR diketahui menyebabkan beban komputasi lebih tinggi, tetapi *inequality* menyederhanakan beban komputasi tersebut sebab hanya membutuhkan satu set persamaan linier. Reformulasi ini dapat diselesaikan dengan KarushKuhn-Tucker (KKT) yang mengambil bentuk seperti sistem linier yang dapat diselesaikan secara efisien dengan metode iteratif seperti *conjugate gradient*.

Estimasi fungsi persamaan (2.18) dapat diselesaikan menggunakan optimasi berikut:

$$\min_{\boldsymbol{\omega},b,e} J(\boldsymbol{\omega},e) = \frac{1}{2}\|\boldsymbol{\omega}\|^2 + \frac{1}{2}C\sum_{i=1}^{T} e_i^2 \tag{2.30}$$

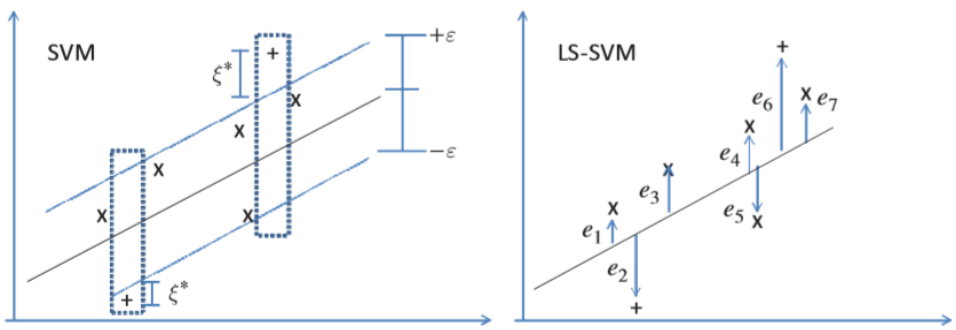
dengan batasan $y_i = \langle \boldsymbol{\omega}, \varphi(\boldsymbol{x}_i)\rangle + b + e_i$ dimana $i = 1,2,\dots,T$, $e_i \in R$ merupakan *error*, dan $C \geq 0$ adalah konstanta regularisasi. Semakin kecil nilai C dapat menghindari *overfitting* untuk kasus *noisy* data. Perlu diperhatikan bahwa ada *Sum of Square Error* (SSE) dan regularisasi pada *cost function* dimana proseduralnya mirip dengan *training* metode FFNN dan *ridge regression*. Adapun optimasi persamaan (2.30) tersebut menggunakan fungsi *Lagrange* sebagai berikut:

$$L_{LS-SVM} = \frac{1}{2}\|\boldsymbol{\omega}\| + \frac{1}{2}C\sum_{i=1}^{T} e_i^2 - \sum_{i=1}^{T}\alpha_i\left\{\langle \boldsymbol{\omega}, \varphi(\mathbf{x}_i)\rangle + b + e_i - y_i\right\} \tag{2.31}$$

dengan *langrange multipliers* $\alpha_i \in R$. Solusi untuk persamaan (2.31) didapat dengan penurunan parsial terhadap masing-masing parameter.

$$\frac{\partial L_{ls-svm}}{\partial \boldsymbol{\omega}} = 0 \quad \rightarrow \quad \boldsymbol{\omega} = \sum_{i=1}^{T} \alpha_i \varphi(\mathbf{x}_i) \tag{2.32}$$
$$\frac{\partial L_{ls-svm}}{\partial b} = 0 \quad \rightarrow \quad \sum_{i=1}^{N} \alpha_i = 0$$
$$\frac{\partial L_{ls-svm}}{\partial e_i} = 0 \quad \rightarrow \quad \alpha_i = Ce_i$$
$$\frac{\partial L_{ls-svm}}{\partial \alpha_i} = 0 \quad \rightarrow \quad \langle \boldsymbol{\omega}, \varphi(\mathbf{x}_i) \rangle + b + e_i - y_i = 0$$

Persamaan (2.32) tersebut mirip dengan optimasi SVR seperti pada persamaan (2.22), kecuali untuk $\alpha_i = Ce_i$, sebab tidak ada *sparseness* pada LS-SVR. Hal tersebut terlihat seperti pada Gambar 2.2 berikut:



Gambar 2.2 *Sparseness* antara SVM dan LS-SVM untuk Regresi Linier (Shabri & Suhartono, 2012)

Berdasarkan Gambar 2.2, pada SVR diperlihatkan dengan adanya *ε-tube* dan beberapa variabel *slack* ξ dan ξ^* yang bersesuaian dengan dua *support vectors*, sedangkan pada LS-SVR *ε-tube* dan variabel *slack* diganti dengan *error* tiap pengamatan yang menunjukkan jarak antara tiap pengamatan terhadap fungsi regresi. Selanjutnya, setelah didapat persamaan (2.32) akan dilakukan eliminasi $\boldsymbol{\omega}, e$ hingga didapat persamaan linier sebagai berikut:

$$\begin{bmatrix} 0 & 1_v^T \\ 1_v & \boldsymbol{\Omega} + \mathbf{I}/C \end{bmatrix} \begin{bmatrix} b \\ \alpha \end{bmatrix} = \begin{bmatrix} 0 \\ y \end{bmatrix} \tag{2.33}$$

dimana $y = [y_1, \dots, y_T]$, $1_v = [1, \dots, 1]$, $\alpha = [\alpha_1, \dots, \alpha_T]$ dan kondisi Mercer akan diaplikasikan lagi sebagai

$$\mathbf{\Omega}_{il} = \langle \varphi(\mathbf{x}_i), \varphi(\mathbf{x}_l) \rangle = K(\mathbf{x}_i, \mathbf{x}_l), \; k, l = 1, 2, \ldots, T \tag{2.34}$$

Hasil dari estimasi model LS-SVR didapat seperti persamaan (2.34) berikut:

$$f(\mathbf{x}) = \sum_{i=1}^{T} \alpha_i K(\mathbf{x}, \mathbf{x}_i) + b \tag{2.35}$$

(Wang dan Hu, 2006)

2.5 *Genetic Algorithm* (GA)

Genetic Algorithm (GA) merupakan suatu teknik optimasi yang didasarkan pada prinsip genetik dan seleksi alam. Populasi terbentuk dari banyak individu yang berkembang sesuai aturan seleksi spesifik dengan memaksimalkan *fitness*. Algoritma ini digunakan uuntuk mendapatkan nilai optimum global dengan cara melakukan perulangan atau iterasi pada konsep evolusi Darwin. Menurut Ismail dan Irhamah, (2008) dan Prosedur GA terdiri dari 7 tahapan utama yang perlu dilakukan yaitu:

1. Mendefinisikan variabel yang dianggap sebagai kromosom.
2. Membentuk populasi awal dari beberapa kromosom yang didalamnya memuat gen kromosom sebagai kandidat solusi permasalahan. Gen pada *genetic algorithm* berisi variabel yang ingin dioptimumkan, pada penelitian ini gen berisi parameter model.
3. Mengevaluasi kemampuan masing-masing kromosom menggunakan nilai fitness. Pada penelitian ini fungsi fitness berupa kriteria kebaikan model.
4. Memilih kromosom dengan nilai fitness maksimum sebagai orang tua.
5. Mengkombinasikan informasi genetika yang ada pada replikasi orang tua melalui pindah silang.
6. Melakukan mutasi untuk memperkenalkan unsur gen baru pada kromosom secara acak.
7. Tahapan diulangi dari tahapan dua hingga mencapai kekonvergenan.

1. Pengkodean

Pengkodean merupakan proses yang menggambarkan bentuk gen dalam kromosom. Pengkodean dapat berupa *bit* atau *real number*, dan lain-lain. Pengkodean

yang dilakukan pada penelitian ini adalah pengkodean *real number* yang merupakan semua kemungkinan nilai parameter untuk dioptimasi

2. Fitness

Fungsi *fitness* digunakan untuk mengukur tingkat kebaikan suatu solusi yang dicari dan bisa merupakan fungsi tujuan atau modifikasi dari fungsi tujuan. Semakin besar nilai *fitness*, semakin besar pula kemungkinan kromosom tersebut untuk tetap *survive* pada generasi selanjutnya. Secara umum *genetics algorithm* mengoperasikan kromosom dengan *fitness* positif dan mencari untuk memaksimumkan *fitness*. Proses minimasi dapat dilakukan dengan membalik fungsi maksimasi (Arkeman dkk, 2012)

3. Selection

Proses *selection* merupakan tahap untuk mendapatkan calon orang tua yang akan dipindahsilangkan pada tahapan berikutnya. Metode seleksi yang akan digunakan yaitu *Roullete Wheel*. Pada metode ini, masing-masing kromosom menempati potongan lingkaran pada Roda *Roulette* secara proporsional sesuai dengan nilai *fitness*nya dan kromosom akan terpilih jika bilangan random yang dibangkitkan berada dalam nilai interval dari kumulatifnya.

4. Crossover

Proses *crossover* merupakan metode untuk memasangkan orang tua/dua kromosom yang untuk membentuk generasi baru. Orangtua diperoleh dari tahapan *selection* menggunakan RWS. Metode *crossover* yang digunakan yaitu *crossover single point* dengan menukarkan satu dari dua sel antar kromosom tersebut sehingga menghasilkan dua kromosom baru.

5. Mutation

Proses mutasi berperan mengubah struktur kromosom secara spontan dengan tujuan terbentuknya suatu mutan yang secara genetik berbeda dari kromosom sebelumnya sehingga didapatkan keberagaman gen. Terdapat dua tahapan pada metode ini, (1) memilih kromosom yang akan dimutasi dengan membandingkan bilangan

random berdistribusi uniform (0,1) di tiap kromosom dengan nilai peluang *mutation* yang telah ditentukan. Jika nilai bilangan random yang dimiliki kromosom tersebut lebih kecil dari Pm maka kromom akan mengalami mutasi, (2) mengganti nilai salah satu gen dengan bilangan random yang dibangkitkan dari batas bawah dan batas atas nilai gen dalam kromosom.

6. Replace

Semua kromosom baik dari hasi pindah silang dan mutasi, akan dihitung nilai dari *fitness function*-nya. Prosedur yang dilakukan yaitu dengan cara mengurutkan kromosom dari nilai *fitness* terbesar hingga terkecil. Ambil N kromosom dengan nilai *fitness* tertinggi dan anggap sebagai generasi baru.

2.6 *Particle Swarm Optimization* (PSO)

Metode *Particle Swarm Optimization* (PSO) dikembangkan berdasarkan perilaku gerakan kawanan burung atau ikan dalam mencari sumber makanan. Kawanan burung tersebut akan menyebar secara acak untuk mencari sumber makanan yang optimum. Arah pergerakan menuju sumber makanan dipengaruhi insting yang juga mampu menemui sumber makanan yang pernah dilalui bahkan sumber makanan terbaik yang pernah ditemukan oleh salah satu burung dalam kelompok. Pada algoritma, hal tersebut identik dengan eksplorasi yang merupakan kemampuan untuk mendapatkan sebuah hasil yang optimum pada setiap daerah yang berbeda-beda (Ghalia, 2008). Sedangkan pada algoritma dasar PSO, sekumpulan m partikel atau individu yang bergerak dipengaruhi oleh 1) inersia, 2) posisi lokal yang tebaik (*local best position*), dan 3) posisi global yang terbaik (*global best position*). Posisi dari m individu tersebut dipengaruhi oleh kecepatan gerak tiap individu. Jika x_{id}^{k} merupakan posisi individu ke-*i* pada dimensi *d* dalam kurun waktu ke-*k* dengan kecepatan v_{id}^{k}, maka kecepatan dan posisi setiap individu ke-*i* berubah menurut persamaan berikut:

$$v_{id}^{k+1} = v_{id}^{k} + c_1 r_1^{t} \left(pbest_{id}^{k} - x_{id}^{k} \right) + c_2 r_2^{k} \left(gbest_{d}^{k} - x_{id}^{k} \right) \tag{2.36}$$

$$x_{id}^{k+1} = x_{id}^{k} + v_{id}^{k+1} \tag{2.37}$$

Persamaan diatas v_{id}^{k} dan x_{id}^{k} menunjukkan kecepatan dan posisi partikel ke-*i* pada waktu ke-k dan dimensi ke-*d*; $pbest_{id}^{k}$ merupakan nilai individu ke-*i* di waktu ke-k dan dimensi ke-*d* pada posisi yang optimum (*local best position*); $gbest_{d}^{k}$ merupakan partikel dimensi ke-*d* pada posisi terbaik dari seluruh gerombolan (*global best position*). Untuk menghindari partikel terlalu jauh dari ruang pencarian, kecepatan partikel dibuat pada tiap arahnya terbatas diantara $-v_{dmax}$ dan v_{dmax}. Jika nilai v_{dmax} terlalu besar, solusi yang didapat akan jauh dari optimal, jika v_{dmax} terlalu kecil, solusi yang didapat akan berisiko jatuh pada lokal optimal; c_1 dan c_2 merupakan koefisien yang mengatur kecepatan partikel sehingga solusi yang didapatkan tidak akan menjadi optimum lokal dan nilai keduanya yang biasa digunakan adalah 2. r_1 dan r_2 merupakan vektor random berdistribusi uniform dengan nilai 0 hingga 1.

Pengembangan dari PSO yang digunakan untuk mengurangi waktu kekonvergenan dan meningkatkan kualitas dari hasil optimasi yakni *Inertia Weight.* Mekanisme *Inertia Weight* dilakukan untuk melakukan kontrol pada eksplorasi dan kecepatan yang dilakukan individu. Penambahan bobot inersia pada PSO biasa disebut *Global Best PSO* (GBPSO) yang merupakan varian PSO paling dasar. Iiterasi kecepatan pada GBPSO ditentukan menggunakan rumus sebagai berikut

$$v_{id}^{k+1} = \omega v_{id}^{k} + c_1 r_1^{t}\left(pbest_{id}^{k} - x_{id}^{k}\right) + c_2 r_2^{k}\left(gbest_{d}^{k} - x_{id}^{k}\right) \tag{2.38}$$

Berdasarkan persamaan (2.40), pengaruh kecepatan terakhir terhadap kecepatan saaat ini dikontrol oleh *inertia weight*. Semakin besar nilai ω maka kemampuan PSO untuk mencari kemungkinan solusi di ruang pencarian semakin besar, begitu sebaliknya. Secara umum, jika ω bernilai satu, maka pada iterasi berikutnya ada kekurangan kemampuan pencarian secara parsial. Hasil eksperimen menunjukkan bahwa PSO memilki konvergensi terbesar ketika ω bernilai antara 0,8 dan 1,2. Jika tidak didapat hasil yang konvergen maka dapat dilakukan dengan membatasi jumlah iterasi, berhenti setelah mendapatkan solusi yang diterima, berhenti

setelah peningkatan tidak ditemukan, berhenti ketika radius dari tiap individu mendekati nol dan berhenti ketika *slope* mendekati nol.

(Bai, 2010)

2.7 *Modified Particle Swarm Optimization* (MPSO)

Pada permalasahan optimasi yang kompleks, algoritma PSO berisiko pada konvergensi dini. Sebagai solusi dari masalah tersebut, rata-rata jarak partikel dan mutase operator digunakan untuk memandu pemilihan populasi awal dan mengevaluasi konvergensi partikel agar terhindar dari optimum lokal. Pemilihan sekumpulan partikel awal dapat ditentukan secara acak. Idealnya, partikel harus berdistribusi seragam di seluruh kemungkinan ruang solusi. Namun, kondisi ideal tersebut sulit dicapai. Parameter rata-rata jarak partikel digunakan untuk meningkatkan probabilitas penemuan solusi global optimal dan terhindari dari lokal optimal. Rata-rata jarak partikel didefinisikan sebagai persamaan berikut:

$$D(t) = \frac{1}{mL} \sum_{i=1}^{m} \sqrt{\sum_{d=1}^{n} \left(x_{id} - \bar{x}_d \right)} \tag{2.39}$$

dimana,

$D(t)$: *discrete degree* dari distribusi partikel pada populasi

L : panjang diagonal maksimum dari ruang pencarian

m : banyak partikel

n : dimensi ruang solusi

x_{id} : partikel ke-*i* pada dimensi ke-*d*

$\bar{x}_d$: rata-rata dari partikel dimensi ke-*d*

Konvergensi algoritma PSO cenderung cepat di tahap awal dan semakin lama akan semakin lamban. Hal tersebut berisiko pada mudahnya populasi terjebak pada lokal optimum dan kehilangan kemampuan untuk berevolusi. Posisi partikel menentukan nilai *fitness*, sehingga keadaan populasi saat ini dan konvergensi dini dapat dievaluasi berdasarkan variasi dari fitness pada semua partikel di populasi. Misal nilai

fitness dari partikel ke-*i* adalah f_i dan rata-rata *fitness* dari populasi saat ini adalah $\bar{f}$. Nilai variansi *fitness* dari populasi dapat didefinisikan sebagai berikut:

$$\sigma^2 = \sum_{i=1}^{m} \left(\frac{f_i - \bar{f}}{f} \right)^2 \tag{2.40}$$

dengan f merupakan skala normalisasi dari σ^2 yang mempunyai kriteris seperti pada persamaan (2.41)

$$f = \begin{cases} \max\left| f_i - \bar{f} \right|, \max\left| f_i - \bar{f} \right| > 1 \\ 1 \qquad , \min\left| f_i - \bar{f} \right| \leq 1 \end{cases} \tag{2.41}$$

Nilai variansi *fitness* menggambarkan *discrete degree* dari partikel populasi. Semakin kecil σ^2, maka semakin kecil juga *discrete degree*, begitu sebaliknya.

Dengan bertambahnya banyak iterasi, nilai $D(t)$ dan σ^2 akan semakin kecil. Ketika $D(t) < \alpha$ dan $\sigma^2 < \beta$ (α, β merupakan nilai *threshold* yang ditentukan), algoritma dianggap masuk pada tahap pencarian akhir dan poplasi akan mengalami konvergensi dini. Algoritma MPSO mempertahankan lokasi optimal dari sekelompok partikel saat ini dan mencari kembali ruang solusi partikel agar dapat keluar dari lokal optimal.

(Niu & Dai, 2017)

2.8 Kriteria Kebaikan Model

Pemilihan model terbaik dilakukan dengan cara evaluasi menggunakan nilai *Root Mean Square Error* (RMSE) dan *Symmetric Mean Absolute Percentage Error* (sMAPE). Metode dengan nilai RMSE dan sMAPE terkecil merupakan metode yang terbaik dari metode lainnya. Nilai RMSE dan sMAPE dihitung berdasarkan perbandingan antara data *outsample* dengan hasil ramalan di tiap metode.

$$RMSE = \sqrt{MSE} = \sqrt{\frac{1}{T} \sum_{t=1}^{T} e_t^2} \tag{2.46}$$

$$sMAPE = \frac{100\%}{T} \sum_{t=1}^{T} \frac{\left| Y_t - \hat{Y}_t \right|}{\left(\left| Y_t \right| + \left| \hat{Y}_t \right| \right) / 2} \tag{2.47}$$

dengan T merupakan banyak data *outsample*, Y_t merupakan data *insample* ke-t dan $\hat{Y}_t$ merupakan data hasil ramalan ke-t.

(Wei, 2006)

2.9 Beban Listrik

Menurut UU No. 30 Tahun 2009 tentang ketenagalistrikkan, didefinisikan tenaga listrik merupakan suatu bentuk energi sekunder yang dibangkitkan, ditransmisikan, dan didistribusikan untuk segala macam keperluan, tetapi tidak meliputi listrik yang dipakai untuk komunikasi, elektronika, atau isyarat. Proses penyaluran tenaga listrik terdiri dari tiga komponen utama yaitu pembangkit, penghantar (saluran transmisi/distribusi), dan beban. Beban listrik dapat diartikan sebagai peralatan yang terkoneksi dengan sistem daya dan mengkonsumsi energi listrik dari daya hasil keluaran pembangkit. Besarnya konsumsi lisrik yang digunakan tergantung dari perilaku konsumen dalam menggunakan alat-alat listriknya. Dari perilaku konsumen tersebut, selanjutnya PT PLN (Persero) harus mampu menyesuaikan agar daya yang dibangkitkan sama dengan beban sistem. Hal tersebut menunjukkan bahwa peramalan beban listrik perlu mendapat perhatian khusus karena sangat berpengaruh terhadap segi manajerial ataupun operasional. Peramalan beban listrik sebagai kajian dalam bidang perencanaan dan evaluasi operasi sistem tenaga listrik memegang peranan yang sangat penting karena berdasarkan jangka waktu peramalannya sesuai dengan Peraturan Menteri dan Sumber Daya Mineral No. 03 Tahun 2007 tentang Aturan Jaringan Sistem Tenaga Listrik Jawa-Madura-Bali dalam Aturan Perencanaan dan Pelaksanaan Operasi, peramalan beban listrik memiliki tujuan berikut:

1. Peramalan Jangka Panjang (*Long-Term Forecasting*) merupakan peramalan beban listrik untuk rencana operasional jangka panjang atau tahunan yang memiliki tujuan utuk menentukan kapasitas serta ketersediaan unit pembangkitan, sistem transmisi, dan sistem distribusi.
2. Peramalan Jangka Menengah (*Mid-Term Forecasting*) merupakan peramalan beban listrik untuk rencana operasional bulanan atau mingguan yang memiliki tujuan

untuk mempersiapkan kebutuhan energi bulanan, penjadwalan, pemeliharaan, dan operasional baik itu unit pembangkitan, sistem transmisi, maupun sistem distribusi.

3. Peramalan Jangka Pendek (*Short-Term Forecasting*), merupakan peramalan beban listrik untuk rencana operasional harian yang memiliki tujuan untuk analisis, perencanaan, dan evaluasi neraca energi, serta studi perbandingan beban listrik hasil peramalan dengan actual tiap jamnya (*real time*).

BAB 3
METODOLOGI PENELITIAN

3.1 Sumber Data

Data yang digunakan dalam penelitian ini adalah data sekunder yang diperoleh dari PT PLN Distribusi Jawa Timur mengenai permintaan beban listrik di kabupaten dan kota Jawa Timur. Beban listrik tersebut dicatat per 30 menit mulai periode Januari 2016 hingga Desember 2016. Data tersebut dibagi menjadi dua kelompok, *training* dan *testing*. Data *training* yang digunakan untuk membangun model yaitu dari periode Januari 2016 hingga November 2016 dan data *testing* pada Desember tahun 2016.

3.2 Variabel Penelitian

Data yang digunakan dalam penelitian ini adalah beban listrik jangka pendek yang dicatat per 30 menit pada gardu 20kV (bagian distribusi) di Jawa Timur. Data penelitian tersebut disusun dalam sebuah struktur data yang disajikan pada Tabel 3.1.

Tabel 3.1 Struktur Data Penelitian

Waktu	**Hari**	**Bulan**	**Beban Listrik**	**Keterangan**
1	1	Januari	$Z_{i,1}$	*Training*
⋮	⋮	⋮	⋮	⋮
16031	334	November	$Z_{i,16031}$	*Training*
16032	334	November	$Z_{i,16032}$	*Training*
16033	335	Desember	$Z_{i,16033}$	*Testing*
⋮	⋮	⋮	⋮	⋮
17520	365	Desember	$Z_{i,17520}$	*Testing*

Variabel yang digunakan dalam penelitian ini terdiri atas variabel dependen (Y_t) dan variabel independen (X) sebagai berikut:

1. Variabel dependen (Y_t) adalah data beban listrik jangka pendek di Jawa Timur periode Januari hingga Desember 2016 yang digunakan sebagai unit observasi

2. Variabel independen (X) adalah lag yang signifikan pada data beban listrik jangka pendek di Jawa Timur dari waktu yang sebelumnya, dengan
   $(\boldsymbol{X} = [Y_{t-1}, Y_{t-2}, \dots, Y_{t-k}])$

3.3 Langkah-Langkah Penelitian

Prosedur analisis data yang digunakan untuk mencapai tujuan penelitian antara lain sebagai berikut:

A. Mendapatkan model peramalan beban listrik terbaik berdasarkan model ARIMA, SVR, dan LSSVR dilakukan dengan langkah-langkah sebagai berikut:
   1. Pemodelan dengan metode ARIMA, dilakukan dengan prosedur berikut
      a. Lakukan pengecekan stasioneritas data melalui identifikasi plot *time series.*
      b. Jika data telah stasioner dalam *mean*, identifikasi kemungkinan model yang terbentuk berdasarkan plot ACF dan PACF
      c. Melakukan estimasi parameter dan pengujian signifikansi parameter.
      d. Lakukan peramalan pada data *outsample* serta hitung RMSE dan sMAPE seperti pada persamaan (2.46) dan (2.47).
      e. Mendapatkan model terbaik dengan membandingkan RMSE dan sMAPE.
   2. Pemodelan dengan metode SVR dilakukan dengan langkah-langkah berikut:
      a. Menentukan range nilai parameter kernel *gaussian* (σ^2), parameter *cost* (C), parameter ε.
      b. Memasukkan nilai lag-lag yang signifikan pada model ARIMA sebagai variabel $\boldsymbol{X}$.
      c. Memodelkan SVR menggunakan data *training* sehingga didapat fungsi (2.30)
      d. Evaluasi pemilihan model SVR pada data *training* dan *testing* berdasarkan nilai RMSE terkecil
      e. Melakukan peramalan menggunakan model terbaik yang didapatkan pada langkah d.

f. Membandingkan hasil peramalan dengan data *outsample* dan menghitung nilai RMSE dan sMAPE yang didapat

3. Pemodelan dengan metode LSSVR dilakukan dengan langkah-langkah berikut:
   a. Menentukan range nilai parameter kernel *gaussian* (σ^2), parameter *cost* (C),
   b. Memasukkan nilai lag-lag yang signifikan pada model ARIMA sebagai variabel $\boldsymbol{X}$.
   c. Memodelkan LSSVR menggunaka data *training* sehingga didapat fungsi (2.37)
   d. Evaluasi pemilihan model SVR pada data *training* dan *testing* berdasarkan nilai RMSE terkecil
   e. Melakukan peramalan menggunakan model terbaik yang didapatkan pada langkah d.
   f. Membandingkan hasil peramalan dengan data *outsample* dan menghitung nilai RMSE dan sMAPE yang didapat.

B. Mendapatkan hasil optimasi terbaik untuk model peramalan beban listrik berdasarkan model SVR-GA, SVR-PSO, LSSVR-GA, dan LSSVR-PSO dilakukan dengan langkah-langkah sebagai berikut:

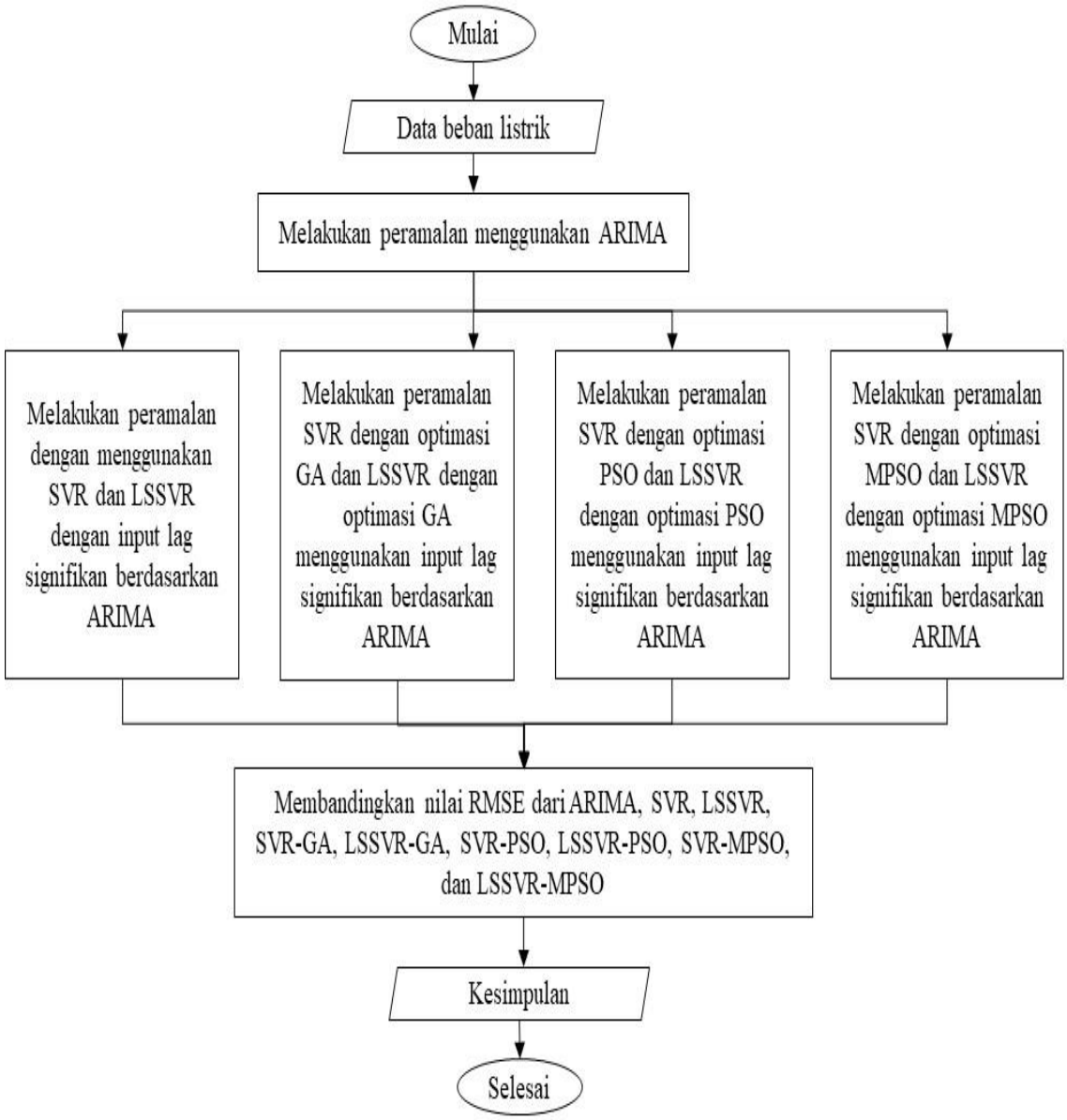
1. Optimasi menggunakan *Genetic Algorithm* pada metode SVR dan LSSVR dapat dilakukan dengan langkah-langkah berikut:
   a. Membentuk populasi awal yang terdiri dari beberapa kromosom
   b. Mengevaluasi tiap kromosom dengan *fitness*. Adapun fungsi *fitness* yang digunakan merupakan kebalikan dari sMAPE sebab kriteria model terbaik pada penelitian ini yaitu meminimumkan fungsi
   c. Melakukan proses seleksi sebanyak N kromosom dari P induk yang berasal dari populasi dengan seleksi *roullete wheel*. Nilai kromosom yang tinggi memiliki peluang untuk terseleksi dan memilih pasangan induk secara acak untuk bereproduksi lebih besar.
   d. Melakukan pindah silang pada individu yang terpilih sebagai orang tua.
   e. Melakukan proses mutasi

f. Pergantian populasi lama dengan populasi generasi yang baru dengan cara memilih kromosom terbaik dari induk dan anak baru yang memiliki fitness tertinggi setelah terjadinya seleksi, pindah silang dan mutasi
g. Melihat apakah solusi yang didapat sudah memenuhi kriteria atau belum. Apabila solusi yang didapat belum mencapai kriteria kekonvergenan maka kembali ke-*b*.
h. Melakukan peramalan dengan nilai parameter SVR dan LSSVR yang telah optimal.

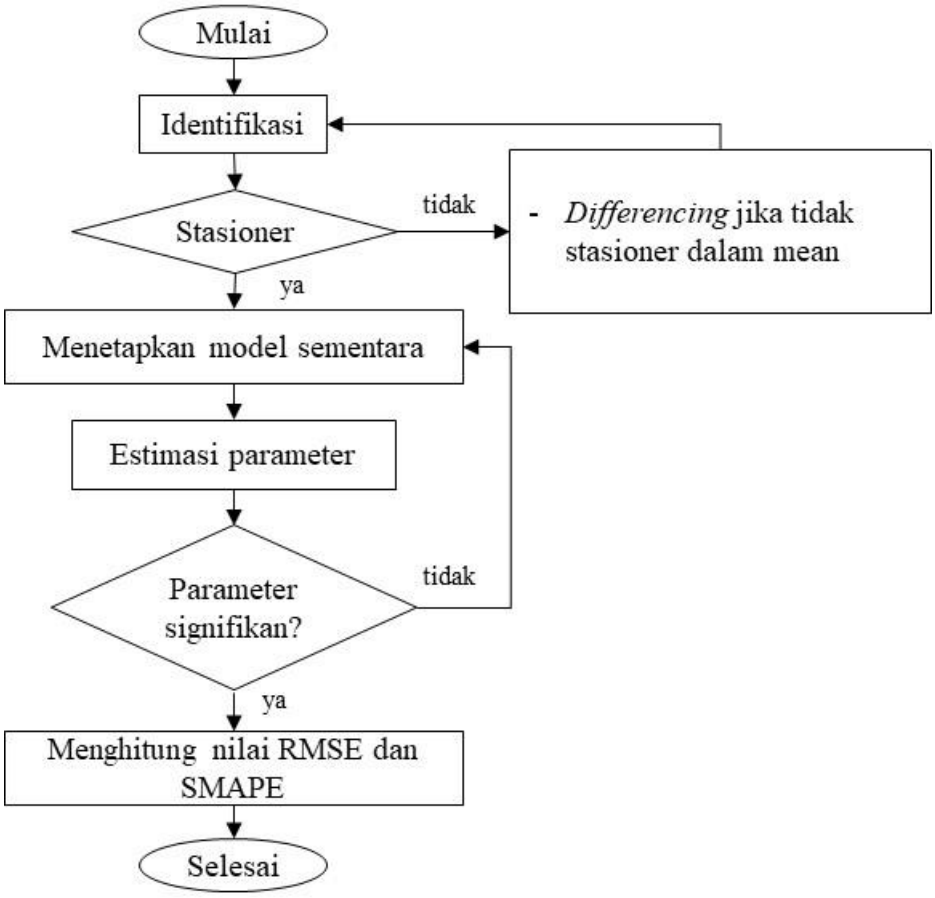
2. Optimasi menggunakan *Particle Swarm Optimization* pada metode SVR dan LSSVR dapat dilakukan dengan langkah-langkah berikut:
   a. Tentukan nilai parameter algoritma PSO yaitu batas iterasi maksimum, inisiasi *inertia weight*, batas maksimum kecepatan, koefisien komponen tiap partikel C_1, dan koefisien komponen semua partikel C_2.
   b. Inisiasi posisi partikel
   c. Menentukan *fitness function* berdasarkan posisi partikel yang merupakan kandungan jumlah makanan.
   d. Melakukan pergeseran pada tiap partikel dengan jarak dan percepatan tertentu
   e. Menghitung nilai *fitness* partikel setelah dilakukan pergeseran.
   f. Mendapatkan nilai best solution dari tiap partikel.
   g. Mendapatkan nilai global best solution dari keseluruhan partikel.
   h. Melakukan peramalan dengan nilai parameter SVR dan LSSVR yang telah optimal
3. Optimasi menggunakan *Modified Particle Swarm Optimization* pada metode SVR dan LSSVR dapat dilakukan dengan langkah-langkah berikut:
   a. Tentukan nilai parameter algoritma PSO yaitu batas iterasi maksimum, inisiasi *inertia weight*, batas maksimum kecepatan, koefisien komponen tiap partikel C_1, dan koefisien komponen semua partikel C_2 beserta nilai Panjang diagonal pergerakan partikel L dan nilai threshold α, β

b. Inisiasi posisi partikel
c. Menentukan *fitness function* berdasarkan posisi partikel yang merupakan kandungan jumlah makanan.
d. Melakukan pergeseran pada tiap partikel dengan jarak dan percepatan tertentu
e. Menghitung nilai parameter *discrete degree* dan *sigma*, jika kurang dari nilai *threshold* maka kembali ke-*b*, dan sebaliknya jika lebih dari *threshold* maka lakukan perbaruan pada partikel dengan menghitung nilai *fitness* partikel setelah dilakukan pergeseran
f. Mendapatkan nilai best solution dari tiap partikel.
g. Mendapatkan nilai global best solution dari keseluruhan partikel.
h. Evaluasi nilai *discrete degree* dan *sigma* dengan kriteria yang sama seperti langkah ke-*e*.
i. Melakukan peramalan dengan nilai parameter SVR dan LSSVR yang telah optimal

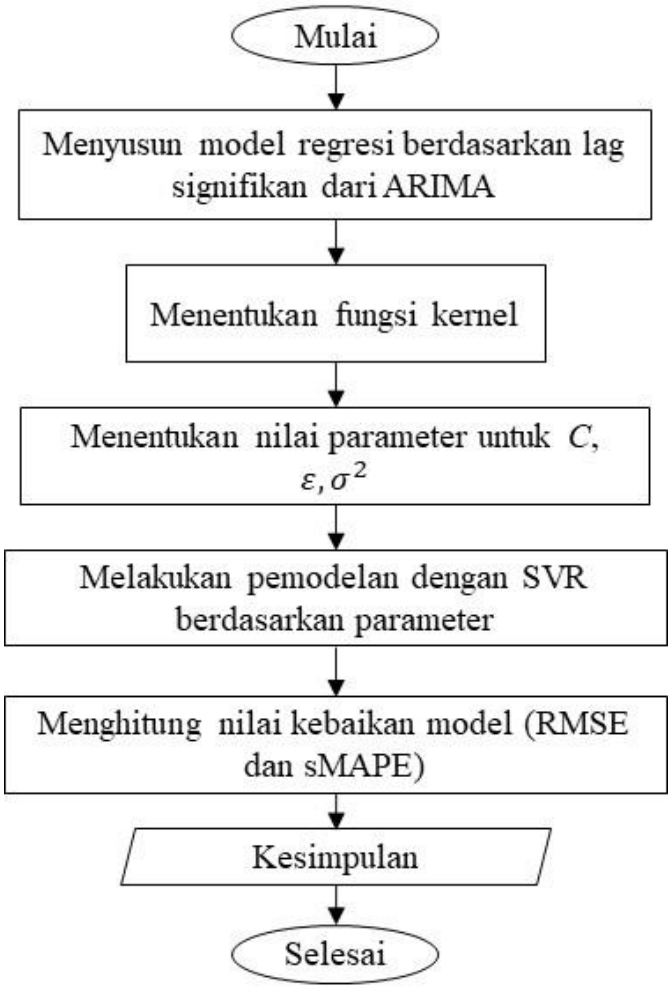
Hasil optimasi terbaik diketahui setelah membandingkan nilai RMSE dan sMAPE dari enam metode yang berbeda, sehingga didapat metode terbaik. Langkah penelitian yang akan dilakukan secara umum, peramalan menggunakan ARIMA, SVR, LSSVR, SVR-GA dan LSSVR-GA, SVR-PSO dan LSSVR-PSO disajikan dalam bentuk diagram alir seperti berikut:



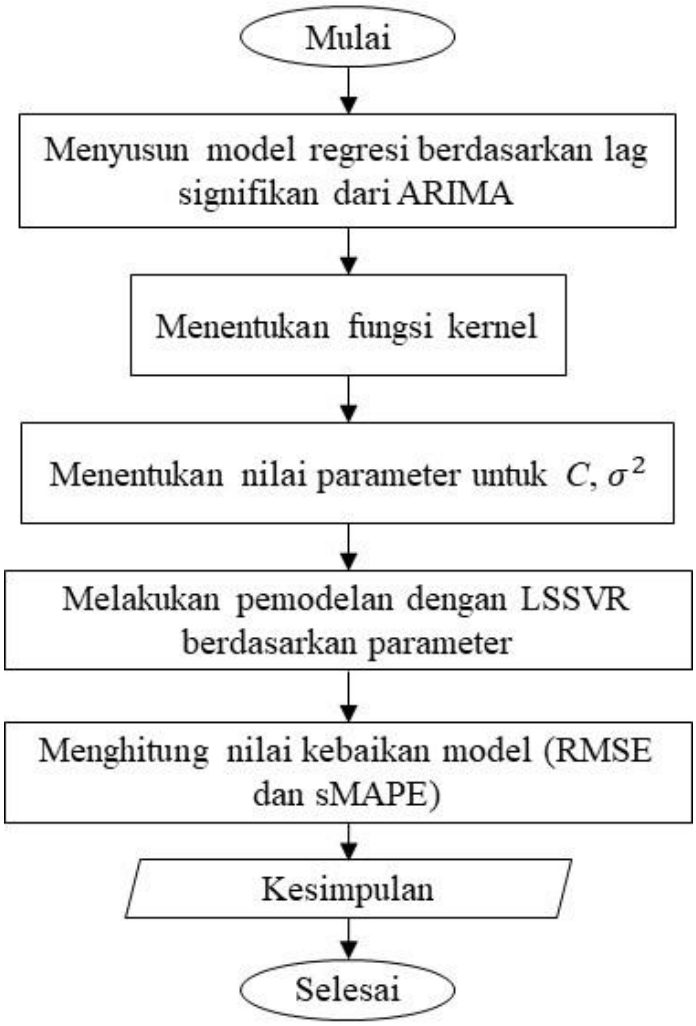
Gambar 3.1 Diagram Alir Penelitian Secara Umum



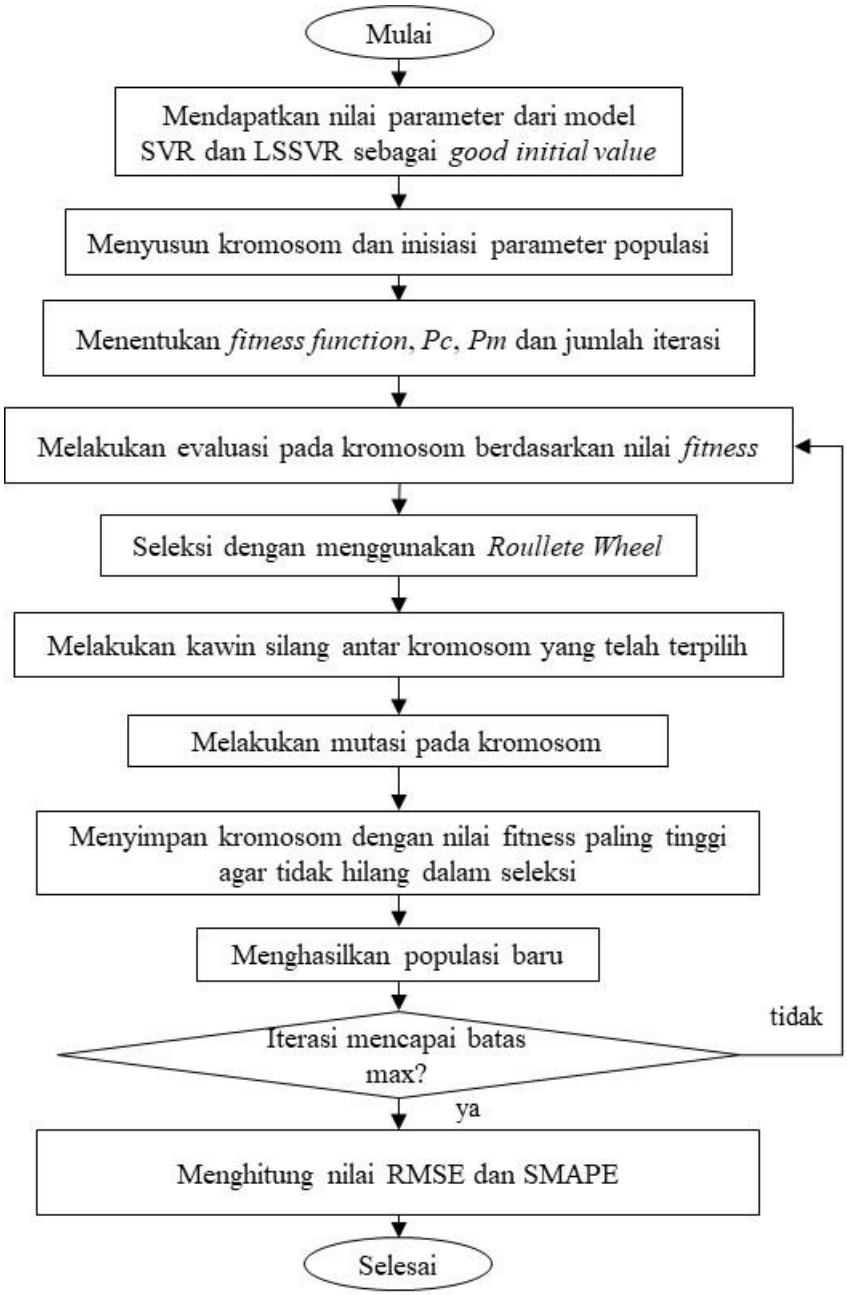
Gambar 3.2 Diagram Alir Pemodelan ARIMA



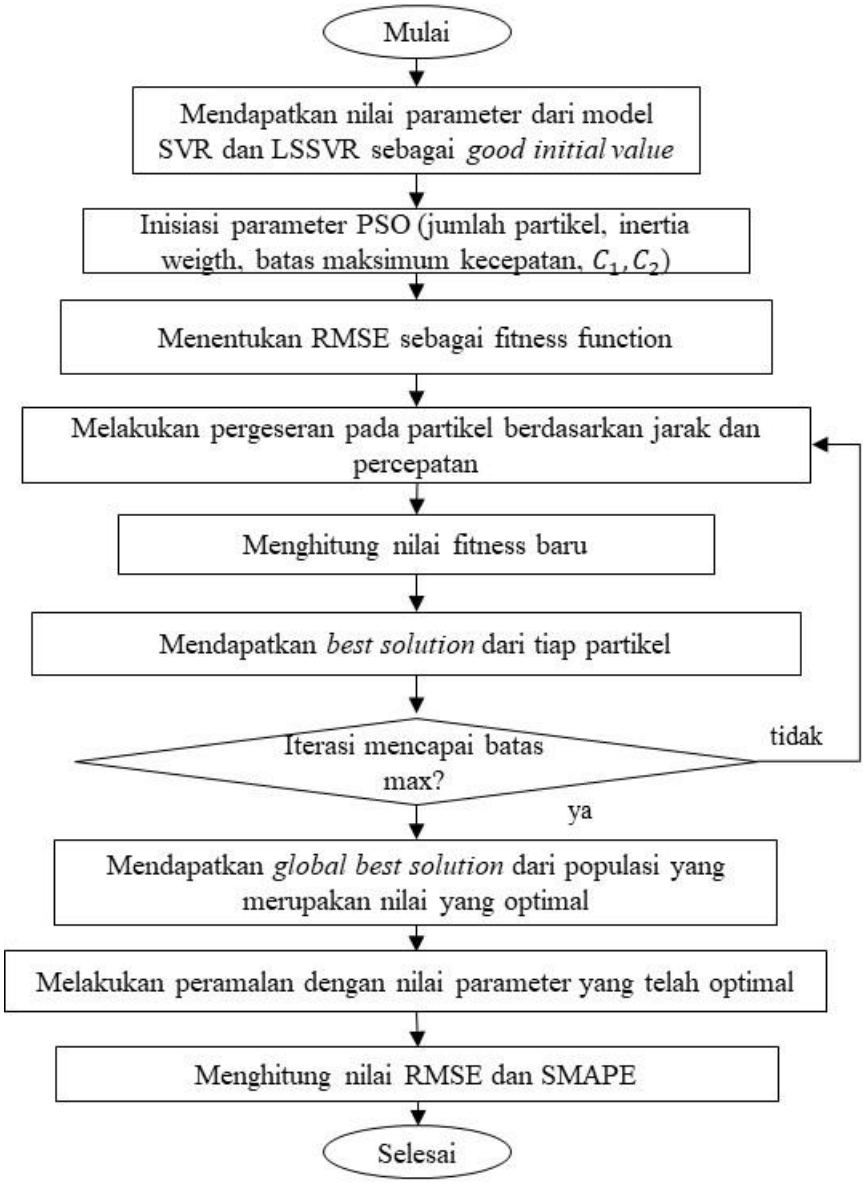
Gambar 3.3 Diagram Alir Peramalan SVR



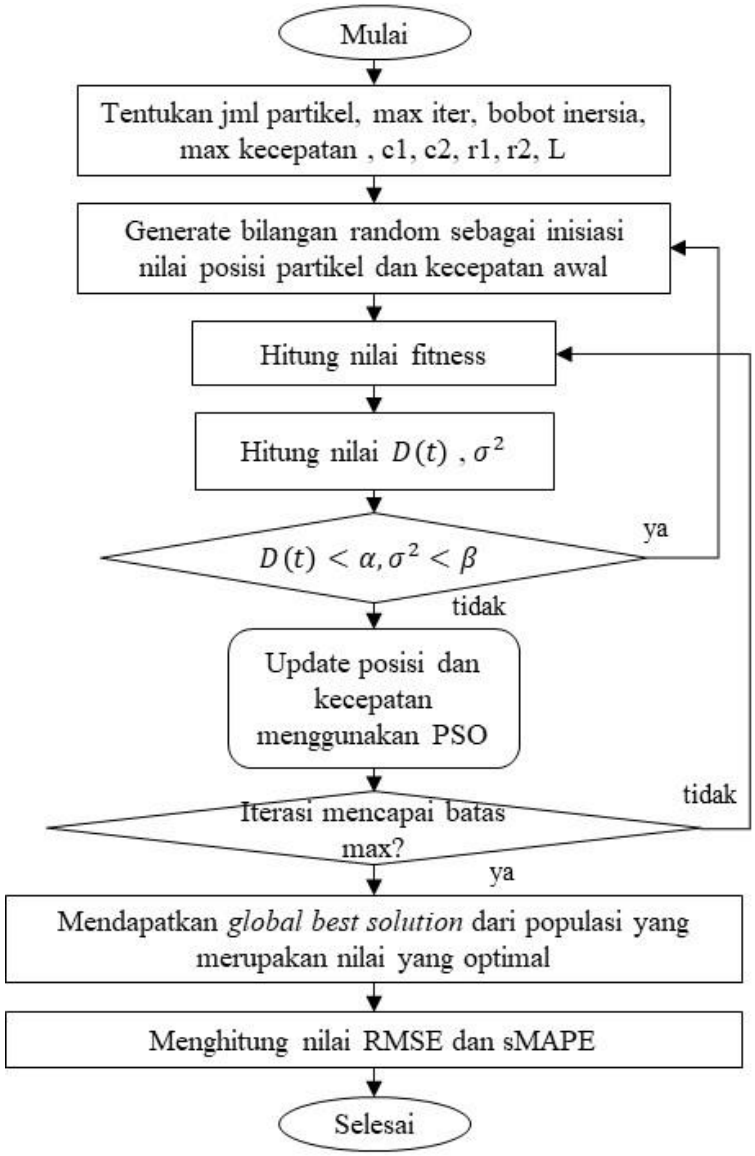
Gambar 3.4 Diagram Alir Peramalan LSSVR



Gambar 3.5 Diagram Alir Peramalan SVR-GA dan LSSVR-GA



Gambar 3.6 Diagram Alir Peramalan SVR-PSO dan LSSVR-PSO



Gambar 3.7 Diagram Alir Peramalan SVR-MPSO dan LSSVR-MPSO

BAB IV

ANALISIS DAN PEMBAHASAN

Pada bab ini dilakukan pembahasan meliputi kinerja dari metode LSSVR dengan beberapa metode optimasi, diantaranya *Grid Search*, *Genetic Algorithm* (GA), *Particle Swarm Optimization* (PSO), dan *Modified Particle Swarm Optimization* (MPSO) pada data beban listrik jangka pendek per setengah jam area Jawa Timur. Selain itu, pada bab ini juga membahas mengenai perbandingan kinerja metode LSSVR dengan metode SVR beserta metode optimasi. Kajian teoritis mengenai metode LSSVR dengan algoritma GA, PSO, dan MPSO akan dijelaskan terlebih dahulu. Percobaan dilakukan menggunakan bahasa program Matlab pada Intel Core i5 yang berjalan dengan RAM 8.00 GB dengan sistem operasi 64-bit.

4.1 Kajian Algoritma LSSVR dengan Optimasi GA, PSO, dan MPSO

A. Desain Algoritma LSSVR dengan Optimasi GA, PSO, dan MPSO

Pada penelitian ini menggunakan metode Least Square Support Vector Regression dan optimasi paramater GA, PSO, MPSO. Berikut ini merupakan algoritma hybrid LSSVR-GA dan LSSVR-PSO.

Algoritma 1. LSSVR

<u>Input</u> : Variabel $\boldsymbol{x}_i = \{x_{i1}, x_{i2}, \dots, x_{ip}\} \in \mathbb{R}^{tp}, i = 1,2, \dots, t$

Data $y_i = \{y_1, y_2, \dots, y_t\}$

Parameter LSSVR

Parameter fungsi kernel

<u>Output</u> : $a, b,$ RMSE, parameter LSSVR optimal, parameter fungsi kernel yang optimal

<u>Begin</u>

1. Menentukan range parameter LSSVR serta *increament* range
2. Menentukan range parameter fungsi kernel serta *increament* range

FOR $i = 1$ in seq_along parameter LSSVR

FOR $i = 1$ in seq_along parameter fungsi kernel

3. Menghitung matriks kernel RBF (K)

4. Menentukan fungsi objektif LSSVR, sesuai persamaan (2.32)
5. Menentukan parameter LSSVR (parameter penalti C)
6. Meminimumkan fungsi *lagrange primal*, sesuai persamaan (2.34)
7. Merubah bentuk ke sistem linier sebagai solusi dari *Quadratic Programming*, seperti persamaan (2.35)
8. Hitung nilai a dan b menggunakan Karush Kuhn Tucker (KKT)
9. Dapatkan nilai prediksi berdasarkan model yang didapat
10. Hitung nilai RMSE

    END FOR

END FOR

11. Dapatkan parameter LSSVR dan fungsi kernel optimal berdasarkan nilai RMSE terkecil
12. Input parameter LSSVR dan fungsi kernel optimal
13. Lakukan kembali langkah 3 hingga 8 sehingga didapat model LSSVR

<u>End</u>

Algoritma 2. LSSVR-GA

Input : Variabel $\boldsymbol{x}_i = \{x_{i1}, x_{i2}, \dots, x_{ip}\} \in \mathbb{R}^{tp}, i = 1,2, \dots, t$

Data $y_i = \{y_1, y_2, \dots, y_t\}$

N : banyak kromosom

MNI : batas iterasi maksimum

$L_{b,C}$ L_{b,σ^2} : batas bawah nilai parameter penalti dan fungsi kernel

$U_{b,C}$ U_{b,σ^2} : batas atas nilai parameter parameter penalti dan fungsi kernel

P_c : peluang pindah silang

P_m : peluang mutasi

P_e : peluang ellitism

Output : a, b, RMSE, parameter LSSVR optimal, parameter fungsi kernel yang optimal

Begin

1. Tentukan N, MNI, P_c, P_m, P_e, $L_{b,C}$, $U_{b,C}$, L_{b,σ^2}, U_{b,σ^2}

FOR $i = 1$ to MNI

2. Inisiasi kromosom menggunakan *good initial value*. Kromosom yang terbentuk merupakan vektor berukuran $1x2$ yang berisi parameter dari fungsi kernel dan parameter penalti LSSVR. Selanjutnya, generate sebanyak N kromosom dengan kriteria nilai paramater $lb_C < C_i < ub_C$ dan $lb_{\sigma^2} < \sigma^2 < ub_{\sigma^2}$
3. Hitung nilai *fitness function*, $F(x) = -f(x)$ dengan $f(x) = RMSE$ berdasarkan metode LSSVR di tiap kromosom
4. Seleksi kromosom untuk dijadikan orang tua menggunakan *Roulette Wheel*, dengan prosedur sebagai berikut:
   i. Hitung nilai proporsi *fitness*
   ii. Hitung nilai *fitness* kumulatif (F_c)
   iii. Generate bilangan random sebanyak N yang berdistribusi uniform diantara 0 dan 1. $RU \sim Unif(0,1)$
   iv. Jika $RU > F_c$, maka kromosom tersebut terpilih sebagai orang tua
5. Generate bilangan random, $RU \sim Unif(0,1)$ sebanyak kromosom terpilih. Jika $RU < P_c$ maka kromosom tersebut akan terpilih sebagai orang tua dan dilakukan proses *crossover* sehingga akan menghasilkan kromosom baru
6. Melakukan proses *crossover singlepoint* yang dilakukan dengan menukarsilangkan salah satu gen secara acak pada kromosom bapak dan ibu
7. Generate bilangan random, $RU \sim Unif(0,1)$ sebanyak gen atau parameter yang tersedia. Jika $RU < P_m$ maka gen tersebut akan dilakukan proses *mutation* dengan mengubah salah satu parameter menjadi suatu bilangan random
8. Urutkan nilai fitness dari semua kromosom yang terbentuk, baik dari kromosom awal, hasil *crossover*, dan *mutation* berurutan dari yang terkecil. Sebanyak $P_e \times$ kromosom sisa dengan nilai fitness tertinggi akan disimpan dan digunakan sebagai orang tua pada generasi berikutnya.
9. Generate N kromosom yang memiliki nilai fitness terendah, selanjutnya lakukan dengan prosedur yang sama

END FOR

10. Pilih kromosom yang memiliki nilai fitness terendah, selanjutnya dapatkan model LSSVR dari parameter tesebut

End

Algoritma 3. LSSVR-PSO

<u>Input</u> : Variabel $\boldsymbol{x}_i = \{x_{i1}, x_{i2}, \dots, x_{ip}\} \in \mathbb{R}^{tp}, i = 1,2, \dots, t$

Data $y_i = \{y_1, y_2, \dots, y_t\}$

N : Jumlah partikel

MNI : batas iterasi maksimum

k : koefisien dalam interval 0.1 dan 1

v_{max} : Batas maksimum kecepatan tiap partikel

$L_{b,C}$ L_{b,σ^2} : batas bawah nilai parameter penalti dan fungsi kernel

$U_{b,C}$ U_{b,σ^2} : batas atas nilai parameter parameter penalti dan fungsi kernel

c_1 : koefisien komponen tiap partikel dalam interval 0 dan 1

c_2 : koefisien komponen semua partikel dalam interval 0 dan 1

w : bobot inersia dalam interval 0.8 dan 1.2

r_1, r_2 : bilangan random berdistribusi uniform dalam interval 0 dan 1

<u>Output</u> : a, b, RMSE, parameter LSSVR optimal, parameter fungsi kernel yang optimal

<u>Begin</u>

1. Tentukan nilai jumlah partikel, MNI, w, $v_{max,C} = k \times U_{b,C}$, $v_{max,\sigma^2} = k \times U_{b,\sigma^2}$, c_1, c_2
2. Inisiasi posisi awal sejumlah N partikel x_{id}^k yang memenuhi kriteria nilai $lb_C < C_i < ub_C$ dan $lb_{\sigma^2} < \sigma^2 < ub_{\sigma^2}$
3. Inisiasi kecepatan partikel v_{id}^k dengan generate bilangan random sebanyak N partikel yang memenuhi kriteria $-v_{max} < v_{id}^k < v_{max}$
4. Hitung nilai *fitness function*, $F(x) = -f(x)$ dengan , $f(x) = RMSE$ berdasarkan metode LSSVR di tiap partikel

5. Tentukan nilai $gbest_{id}^{k}$ yang merupakan posisi dengan *fitness* maksimum dan $pbest_{id}^{k}$ tiap partikel sama dengan x_{id}^{k}

FOR $i = 1$ to MNI

6. Hitung nilai *fitness* dari tiap partikel $F(x)_i$
7. Jika nilai $F(x)_i >$ max $F(x)$, maka update nilai $gbest_{id}^{k+1}$
8. Jika nilai $gbest_{id}^{k+1} > gbest_{id}^{k}$, maka $gbest = gbest_{id}^{k+1}$ dan $pbest = pbest_{id}^{k+1}$
9. Generate r_1, r_2, selanjutnya hitung nilai kecepatan $\left(v_{id}^{k+1}\right)$ seperti pada persamaan (2.40)
10. Beri konstrain jika nilai $v_{id}^{k+1} > v_{max}$ atau $v_{id}^{k+1} \leq -v_{max}$ maka $v_{id}^{k+1} = v_{max}$
11. Hitung nilai pergeseran tiap partikel $\left(x_{id}^{k+1}\right)$ seperti pada persamaan (2.39)
12. Beri konstrain pada nilai pergeseran tiap partikel

    if $x_{id}^{k+1} > U_{b,C} \parallel x_{id}^{k+1} > U_{b,\sigma^2} \parallel x_{id}^{k+1} > L_{b,C} \parallel x_{id}^{k+1} > L_{b,C}$

    $x_{id}^{k+1} = x_{id}^{k}$

    end
13. Ulangi langkah lima hingga 11 sampai memenuhi kriteria kekonvergenan

END FOR

14. Pilih partikel dengan nilai fitness tertinggi, selanjutnya dapatkan model LSSVR berdasarkan parameter dari partikel tersebut

End

Algoritma 4. LSSVR-Modified PSO

<u>Input</u> : Variabel $\boldsymbol{x}_i = \{x_{i1}, x_{i2}, \dots, x_{ip}\} \in \mathbb{R}^{tp}, i = 1,2, \dots, t$

Data $y_i = \{y_1, y_2, \dots, y_t\}$

N : Jumlah partikel

MNI : batas iterasi maksimum

k : koefisien dalam interval 0.1 dan 1

v_{max} : Batas maksimum kecepatan tiap partikel

$L_{b,C}$ L_{b,σ^2} : batas bawah nilai parameter penalti dan fungsi kernel

$U_{b,C}$ U_{b,σ^2} : batas atas nilai parameter parameter penalti dan fungsi kernel

c_1 : koefisien komponen tiap partikel

c_2 : koefisien komponen semua partikel

w: bobot inersia

r_1, r_2 : bilangan random berdistribusi uniform dalam interval 0 dan 1

L : panjang diagonal dari ruang pencarian

$\propto, \beta$: suatu nilai threshold

<u>Output</u> : $a, b,$ RMSE, parameter LSSVR optimal, parameter fungsi kernel yang optimal

<u>Begin</u>

1. Tentukan nilai jumlah partikel, MNI, w, $v_{max,C} = k \times U_{b,C}$, $v_{max,\sigma^2} = k \times U_{b,\sigma^2}$, c_1, c_2
2. Inisiasi posisi awal sejumlah banyak partikel x_{id}^k sekaligus kecepatan awalnya x_{id}^k dengan generate bilangan random sebanyak N partikel yang memenuhi kriteria kriteria $lb_C < C_i < ub_C$ dan $lb_{\sigma^2} < \sigma^2 < ub_{\sigma^2}$
3. Hitung nilai *fitness function*, $F(x) = -f(x)$ dengan , $f(x) = RMSE$ berdasarkan metode LSSVR di tiap partikel
4. Tentukan nilai $gbest_{id}^k$ yang merupakan posisi dengan *fitness* maksimum dan $pbest_{id}^k$ tiap partikel sama dengan x^0
5. Hitung nilai rata-rata *fitness* $(\bar{f})$
6. Dapatkan nilai koefisien scaling normalisasi dengan kriteria sesuai persamaan (2.41)
7. Hitung nilai $D(t)$ dan σ^2 seperti pada persamaan (2.39)

WHILE $D(t) > \alpha \parallel \sigma^2 > \beta$

8. Generate x_{id}^k sebanyak N partikel yang memenuhi kriteria $lb_C < C_i < ub_C$ dan $lb_{\sigma^2} < \sigma^2 < ub_{\sigma^2}$
9. Evaluasi nilai $D(t)$ dan σ^2

END WHILE

WHILE $D(t) < \alpha \parallel \sigma^2 < \beta$

10. Update posisi dan pergerakan partikel berdasarkan metode PSO pada langkah 4 hingga 12
11. Evaluasi nilai $D(t)$ dan σ^2

    WHILE $D(t) < \alpha \parallel \sigma^2 < \beta$

    12. Generate x_{id}^k sebanyak N partikel yang memenuhi kriteria $lb_C < C_i < ub_C$ dan $lb_{\sigma^2} < \sigma^2 < ub_{\sigma^2}$
    13. Evaluasi nilai $D(t)$ dan σ^2

    END WHILE

END WHILE

14. Pilih partikel dengan nilai fitness tertinggi, selanjutnya dapatkan model LSSVR berdasarkan parameter dari partikel tersebut

End

B. Illustrasi Algoritma LSSVR dengan Optimasi GA dan PSO

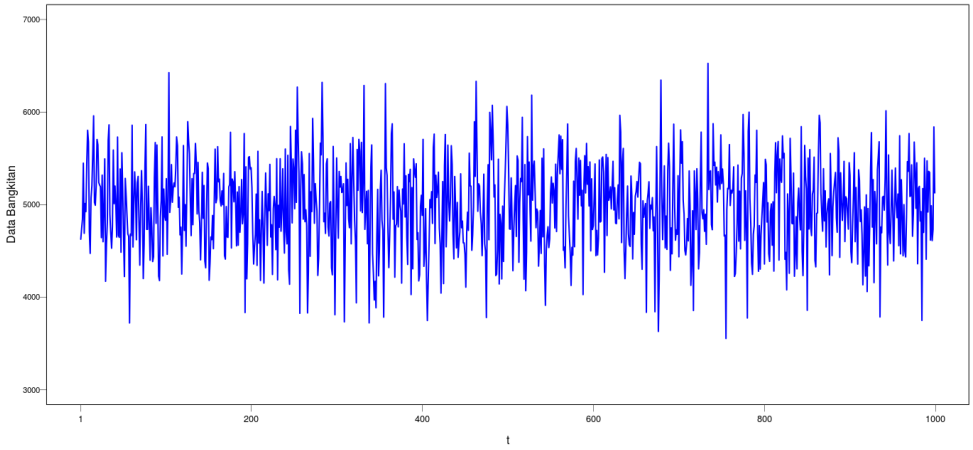
Desain algoritma LSSVR dengan optimasi GA dan PSO akan diilustrasikan menggunakan data AR(1). Sebanyak 1000 data dibangkitkan dengan karakteristik sebagai berikut:

Tabel 4.1. Karakteristik Data Bangkitan AR(1)

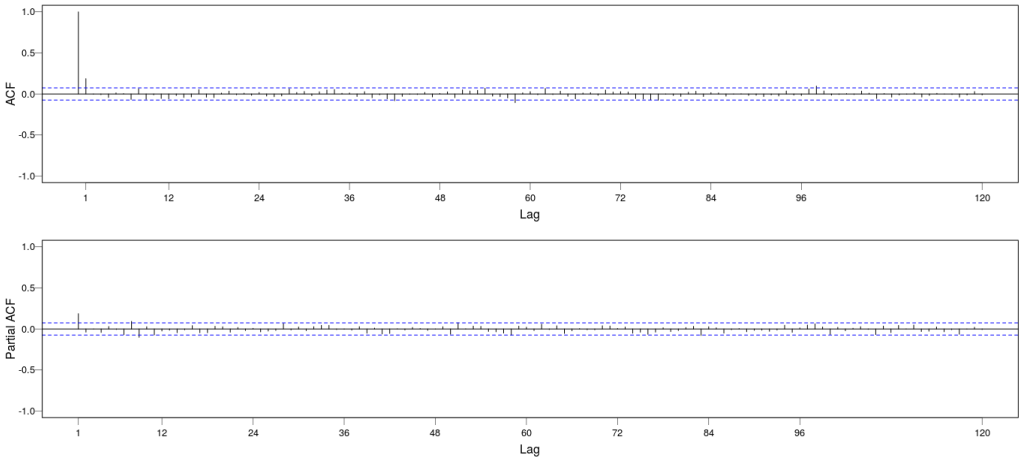
Minimum	Mean	Median	Maximum	Std
3795	5011	5009	6490	482,498

Penggunaan metode LSSVR membutuhkan *support vector* yang digunakan untuk membentuk model. *Support vector* yang digunakan merupakan input dari lag-lag yang berpengaruh signifikan berdasarkan model ARIMA. Berikut merupakan pola data AR(1) yang didapat seperti pada Gambar 4.1. Dari Gambar 4.1 diketahui bahwa secara visual pola data bangkitan AR(1) yang didapat memiliki pola yang stasioner baik dalam mean ataupun varians, sebab data tersebar disekitar nilai rata-ratanya. Pemodelan ARIMA akan dilakukan dengan data *training* yaitu sebanyak 700 data pertama, sedangkan sisanya yaitu sebanyak 300 data digunakan sebagai data *testing* untuk validasi kebaikan model yang dihasilkan. Pengecekan stasioneritas data *training* akan

dilakukan menggunakan pola ACF dan PACF data yang ditunjukkan seperti pada Gambar 4.2



Gambar 4.1 Pola Data AR(1)



Gambar 4.2 Plot ACF dan PACF Data AR(1)

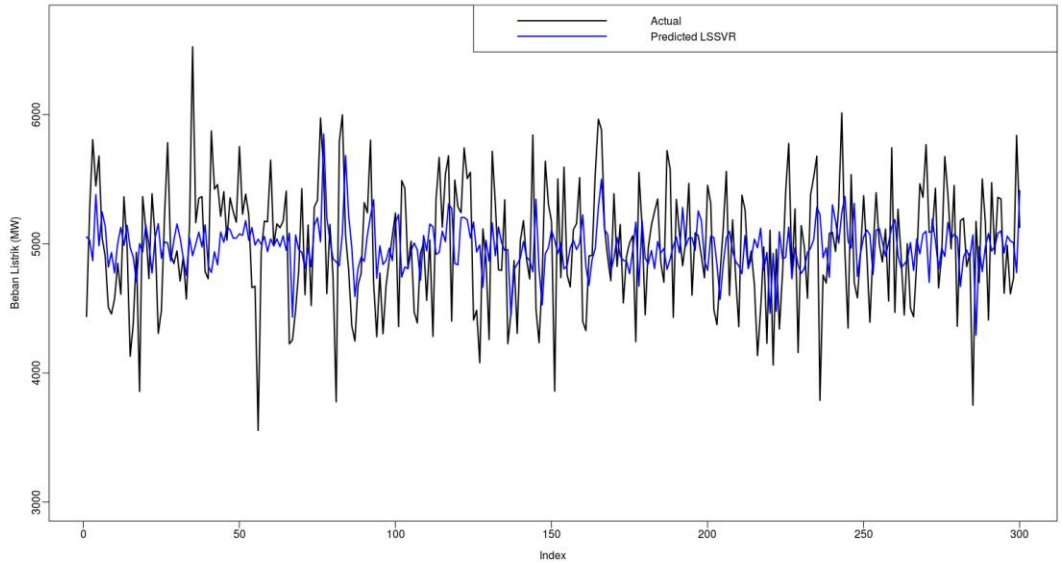
Berdasarkan Gambar 4.2 terlihat bahwa lag yang signifikan yaitu lag 1 sehingga model yang terbentuk yaitu model AR(1) dengan hasil estimasi seperti pada Tabel 4.2

Tabel 4.2 Uji Signifikansi Model ARIMA Data Bangkitan

Model	Parameter		p-value
ARIMA(1,0,0)	θ_1	$1{,}8664 \times 10^{-1}$	$5{,}69 \times 10^{-7}$

Lag yang signifikan dari model ARIMA akan dijadikan sebagai input variabel pada model LSSVR. Sebab data yang digunakan merupakan model AR(1), maka sudah pasti input lag yang akan digunakan pada model LSSVR adalah Z_{t-1}. Selanjutnya data pertama sebanyak lag terbesar akan dihapus, pada data bangkitan terdapat satu data awal yang dihapus sehingga akan bernilai (*) pada satu data pertama tersebut.

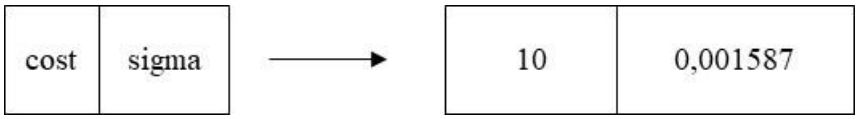
Dalam model LSSVR, terdapat dua inputan parameter yaitu parameter cost (C) dan parameter fungsi kernel RBF (σ^2). Untuk mendapatkan hasil estimasi yang optimal diperlukan pemilihan inputan parameter yang tepat. Salah satu metode yang dapat digunakan untuk memilih inputan parameter yaitu Grid Search. Metode Grid Search membagi range parameter yang dioptimalkan ke dalam grid dan melintasi semua titik untuk mendapatkan parameter yang optimal. Range parameter yang digunakan dalam penelitian ini adalah $\{(1e^{-5}: 1e^{-2}), (10: 50)\}$ untuk σ^2 dan C. Setelah didapat model, maka akan dilakukan pemodelan selama satu bulan kedepan. Hasil ramalan dibandingkan dengan data *testing* dan dihitung nilai akurasinya. Kriteria akurasi yang digunakan yaitu RMSE minimum dan parameter optimal yang didapat untuk σ^2 dan c yaitu sebesar 0,1587 dan 10 dengan nilai RMSE sebesar 488,8115. Berdasarkan Gambar 4.3 terlihat bahwa perbandingan plot data *testing* dengan hasil prediksi dari model LSSVR menunjukkan hasil yang tidak baik karena nilai ramalan yang didapat tidak mendekati nilai asli. Nilai sMAPE dan RMSE yang dihasilkan adalah sebesar 7,92% dan 488,8115.



Gambar 4.3 Plot *Actual* vs *Predicted* LSSVR Data Testing AR(1)

Selanjutnya, data akan diramalkan menggunakan metode optimasi GA untuk mendapatkan nilai parameter yang optimum pada model LSSVR dan diharapkan model yang didapat dapat lebih baik dibandingkan menggunakan metode *Grid Search*. Adapun berikut merupakan prosedur yang dilakukan pada optimasi GA

1. Mendefinisikan variabel-variabel yang diperlukan dalam optimasi GA. Variabel tersebut adalah jumlah populasi, banyak iterasi, peluang pindah silang, peluang mutasi, banyak kromosom untuk *ellitism*, batas atas dan batas bawah masing-masing parameter inputan. Adapun besaran yang ditentukan yaitu N=50, $MNI = 25$, $P_c = 0{,}8$, $P_m = 0{,}1$, $n_e = 1$, $L_{b,C} = 1e-2$, $U_{b,C} = 50$, $L_{b,\sigma^2} = 1e-5$, $U_{b,\sigma^2} = 10$
2. Melakukan inisiasi kromosom. Optimasi dilakukan pada parameter model LSSVR dengan fungsi kernel Gaussian sehingga kromosom yang terbentuk yaitu vektor berukuran 2×1. Vektor tersebut berisi nilai parameter dari LSSVR yakni C dan σ^2. Nilai pertama dari pembentukan kromosom akan diisi dengan *good initial value* yaitu hasil dari pemodelan LSSVR sebelumnya



Gambar 4.4 Ilustrasi Bentuk Kromosom

Selanjutnya akan dibangkitkan data sejumlah banyak populasi sehinga terbentuk 50 kromosom dengan nilai parameter sesuai dengan batas yang telah ditentukan seperti yang ditunjukkan pada Tabel 4.3

3. Hitung nilai fitness di tiap kromosom. Nilai fitness dihitung menggunakan , $F(x) = \frac{1}{f(x)}$ dengan , $f(x) = sMAPE$. Hasil perhitungan disajikan pada Tabel 4.3

Tabel 4.3 Ilustrasi Pembentukan Kromosom

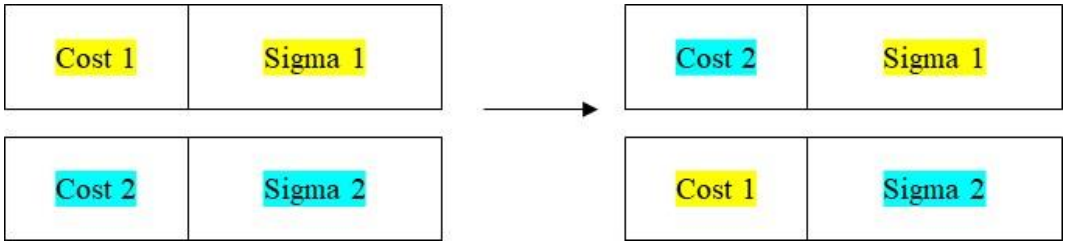
Kromosom	Cost	Sigma	Fitness
1	10	0,001587	85,52372
2	50	0,001062	75,89574
⋮	⋮	⋮	⋮
49	45,5556	0,003165	85,89475
50	27,7778	0,004742	87,14027

Selanjutnya tiap kromosom akan diseleksi untuk dijadikan sebagai calon orang tua. Seleksi dilakukan dengan menggunakan metode *roulette wheel*. Tahapan metode *roulette wheel* dilakukan dengan menghitung proporsi nilai *fitness*, menghitung nilai kumulatif dari *fitness*, membangkitkan bilangan random berdistribusi uniform antara 0 dan 1, kemudian membandingkan nilai kumulatif dengan bilangan random tersebut. Jika nilai kumulatif ke-*i* lebih besar dari bilangan random, maka kromosom ke-*i* tersebut akan terpilih sebagai orang tua untuk generasi berikutnya. Ilutrasi perhitungan metode *roullete wheel* ditunjukkan pada Tabel 4.4. Berdasarkan Tabel 4.4 dapat diketahui bahwa kromosom ke-1 dan 49 memenuhi kriteria nilai kumulatif *fitness* lebih besar dari bilangan random, sehingga kromosom tersebut terpilih sebagai orangtua.

Tabel 4.4 Ilustrasi Proses *Roulette Wheel*

Kromosom	Fitness	Proporsi Nilai *fitness*	Kumulatif *fitness*	Bilangan random
1	85,52372	0,019638	0,019638	0,00993
2	75,89574	0,017425	0,037061	0,8675
⋮	⋮	⋮	⋮	⋮
49	85,89475	0,019721	0,979993	0,5425
50	87,14027	0,020007	1	0,9923

4. Proses berikutnya yaitu melakukan pindah silang yang dimaksudkan untuk menghasilkan kromosom baru dari perpaduan dua kromosom orang tua. Pindah silang dilakukan dengan menggunakan metode *crossover single point* atau *crossover* sederhana. Prosedur yang dilakukan yaitu dengan menukarsilangkan salah satu gen pada kromosom bapak dan ibu. Titik potong untuk penukaran gen dapat dilakukan secara acak, akan tetapi pada penelitian ini tidak perlu menentukan titik potong sebab hanya terdapat dua gen di tiap kromosom. Selanjutnya, akan dibangkitkan bilangan random berdistribusi uniform (0,1), jika bilangan tersebut kurang dari peluang pindah silang ($P_c = 0{,}8$) maka kromosom tersebut akan terpilih sebagai orangtua yang melakukan proses crossover. Ilustrasi pindah silang disajikan seperti pada Gambar 4.5



Gambar 4.5 Ilustrasi *Crossover* Kromosom

Kromosom 1 melakukan penukaran gen dengan kromosom 2, sehingga menghasilkan kromosom baru pada bagian kanan. Selanjutnya kromosom baru tersebut juga dihitung nilai *fitness*.

5. Tahap selanjutnya yaitu mutasi menggunakan mutasi uniform. Proses mutasi dimaksudkan untuk mendapatkan keberagaman gen atau parameter dalam kasus

ini. Mutasi dilakukan dengan mengubah salah satu nilai parameter dengan suatu bilangan random. Pemilihan kromosom yang akan melakukan proses mutasi yaitu dengan membangkitkan berdistribusi uniform $(0,1)$ dan membandingkan dengan peluang mutasi $(P_m = 0{,}1)$, jika bilangan yang dibangkitkan bernilai lebih kecil maka kromosom tersebut merupakan kromosom terpilih untuk melakukan mutasi. Ilustrasi metode mutasi uniform disajikan pada Gambar 4.6

Cost	*Sigma*	*U*
24,649	9,965	0,967
9,457	3,699	**0,017**
9,457	9,965	**0,082**
24,649	3,699	0,283

→

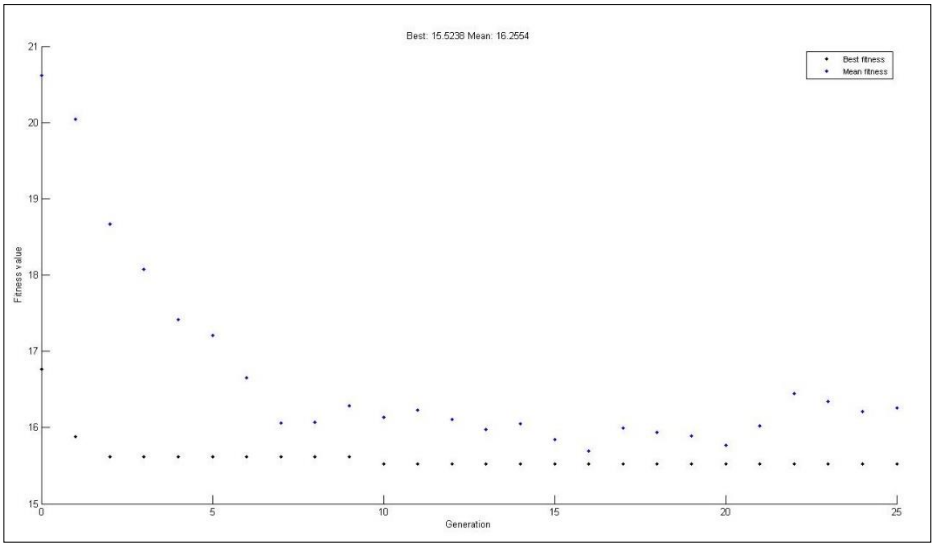
Cost	*Sigma*
24,649	9,965
0,804	3,699
0,326	9,965
24,649	3,699

Gambar 4.6 Ilustrasi Mutasi Uniform (kiri sebelum mutasi, kanan sesudah mutasi)

Berdasarkan Gambar 4.6 terlihat bahwa kromosom ke-2 dan ke-3 memiliki bilangan random kurang dari peluang yang ditentukan sehingga nilai salah satu gen kromosom tersebut diganti dengan bilangan berdistribusi uniform $(0,1)$.

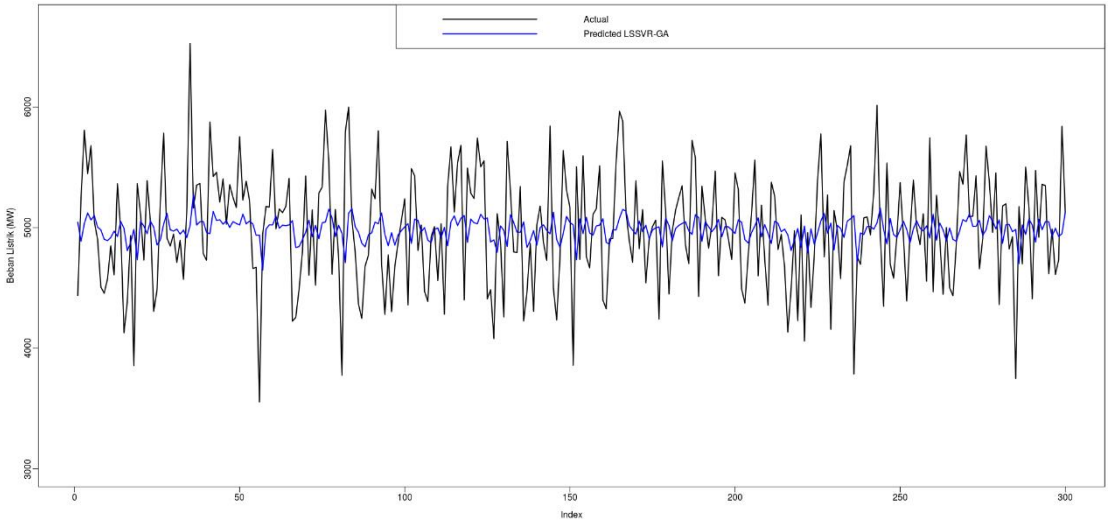
6. Proses berikutnya yaitu mengurutkan nilai *fitness* dari semua kromosom yang terbentuk, baik dari 50 kromosom awal, hasil pindah silang, dan mutasi dari paling kecil hingga paling besar. Sebanyak n_e kromosom dengan nilai *fitness* tertinggi dipertahankan dan digunakan sebagai orang tua pada generasi berikutnya. Selanjutnya adalah membangkitkan generasi baru sebanyak N kromosom dan dilakukan iterasi hingga iterasi ketiga/kekonvergenan terpenuhi.

Setelah dilakukan ketujuh tahapan diatas maka akan dihasilkan nilai ramalan yang dapat dibandingkan dengan data *testing*. Adapun performa dari optimasi parameter menggunakan algoritma GA disajikan pada Gambar 4.7



Gambar 4.7 Performansi *Genetic Algorithm* Data AR(1)

Pada Gambar 4.7 menunjukkan konvergensi dari *Genetic Algorithm* dicapai pada iterasi ke-25 dengan nilai rata-rata *fitness* sebesar 16,2554 dan *fitness* tertinggi sebesar 15,5238. Berdasarkan nilai *fitness* maksimum, didapatkan parameter optimal sebesar 9,816 dan 2,701 untuk σ^2 dan C dengan nilai RMSE sebesar 475,2527. Berdasarkan Gambar 4.8 terlihat bahwa perbandingan plot data *testing* dengan hasil prediksi dari model LSSVR-GA menunjukkan hasil yang tidak baik karena nilai ramalan yang didapat tidak mendekati nilai asli. Nilai sMAPE dan RMSE yang dihasilkan adalah sebesar 7,73% dan 475,2527.



Gambar 4.8 Plot *Actual* vs *Predicted* LSSVR-*Genetic Algorithm* Data Testing AR(1)

Selain itu, data juga akan diramalkan menggunakan metode optimasi PSO untuk mendapatkan nilai parameter yang optimum pada model LSSVR dan diharapkan model yang didapat dapat lebih baik dibandingkan menggunakan metode *Grid Search*. Adapun berikut merupakan prosedur yang dilakukan pada optimasi PSO.

1. Mendefinisikan parameter yang digunakan dalam PSO, yaitu jumlah partikel, batas iterasi maksimum, bobot inersia, batas maksimum kecepatan, koefisien komponen tiap partikel dan koefisien komponen semua partikel. Adapun besaran yang ditentukan yaitu nilai $N = 50$, $MNI = 50$, $w = [0.8, 0.9, 1.1. 0.1. 1.1]$, $k = .2,\ , v_{max,C} = k \times U_{b,C}$, $v_{max,\sigma^2} = k \times U_{b,\sigma^2}$, $c_1 = 2$, $c_2 = 2$. Penentuan nilai w, c_1, c_2 sebagaimana disebutkan oleh Bai (2010) bahwa $c_1 = c_2$ biasanya bernilai 2 dan nilai w optimum berada disekitar nilai 0,8 hingga 1,2 sehingga akan dicari berapa nilai w yang optimum secara empiris.
2. Inisiasi vektor posisi dan vektor kecepatan dari tiap partikel. Dimensi dari partikel ditentukan berdasarkan banyak parameter yang akan dioptimasi. sehingga didapat vektor berukuran 2×1. Inisiasi kecepatan partikel v_{id}^k merupakan bilangan bangkitan yang memenuhi kriteria $-v_{max} < v_{id}^k < v_{max}$. Adapun untuk vektor posisi x_{id}^k berisi nilai parameter dari LSSVR yakni C dan σ^2. Nilai pertama dari

pembentukan posisi partikel akan diisi dengan *good initial value* yaitu hasil dari pemodelan LSSVR sebelumnya. Range parameter yang digunakan secara berturut-turut untuk *cost* dan *sigma* adalah {(1e-2:50),(1e-5:10)}.

3. Hitung nilai *fitness* di tiap partikel. Nilai fitness dihitung menggunakan , $F(x) = \frac{1}{f(x)}$ dengan , $f(x) = sMAPE$ berdasarkan perhitungan data *testing*. Hasil perhitungan disajikan pada Tabel 4.5

Tabel 4.5 Ilustrasi Populasi Awal PSO

Partikel	x_{i1}^{k}	x_{i2}^{k}	Fitness
1	10	0,001587	85,52372
2	50	0.01	88,8801
⋮	⋮	⋮	⋮
49	32,22	0,0089	88,65062
50	18,889	0,008423	89,29925

4. Hitung nilai *local best/Pbest* dan *global best/Gbest.* Pada iterasi awal, semua inisiasi posisi partikel merupakan *Pbest*, sedangkan *Gbest* merupakan nilai *fitness* terbaik. Pada iterasi pertama sebagaimana Tabel 4.5, didapat nilai *fitness* terbaik sebesar 89,29925 yang juga merupakan nilai *Gbest*.
5. Selanjutnya hitung nilai kecepatan partikel untuk bergerak pada iterasi berikutnya. Pergerakkan partikel didasarkan pada nilai *Pbest* dan *Gbest.* Kecepatan perpindahan dihitung berdasarkan persamaan (2.38), lalu partikel akan bergerak pada posisi baru sesuai dengan perhitungan kecepatan. Posisi pergerakkan partikel ditujukkan pada persamaan (2.37). Jika posisi partikel terbaru untuk x_{i1}^{k+1} tidak termasuk pada $\left[L_{b,C}\ ,\ U_{b,C}\ \right]$ atau x_{i2}^{k+1} tidak termasuk pada $\left[L_{b,\sigma^2}\ , U_{b,\sigma^2}\ \right]$ maka ubah nilai posisi partikel terbaru menjadi nilai posisi partikel sebelumnya, $x_{i1}^{k+1} = x_{i1}^{k}$. Sehingga pada iterasi kedua posisi partikel dapat diilustrasikan seperti Gambar 4.9

$x_{i1}^{(1)}$	$x_{i2}^{(1)}$
0,156	9.75
$x_{i1}^{(2)}$ ↓	$x_{i2}^{(2)}$
0,00053	5,442
$x_{i1}^{(2)}$ ↓	$x_{i2}^{(2)}$
0,156	5,442

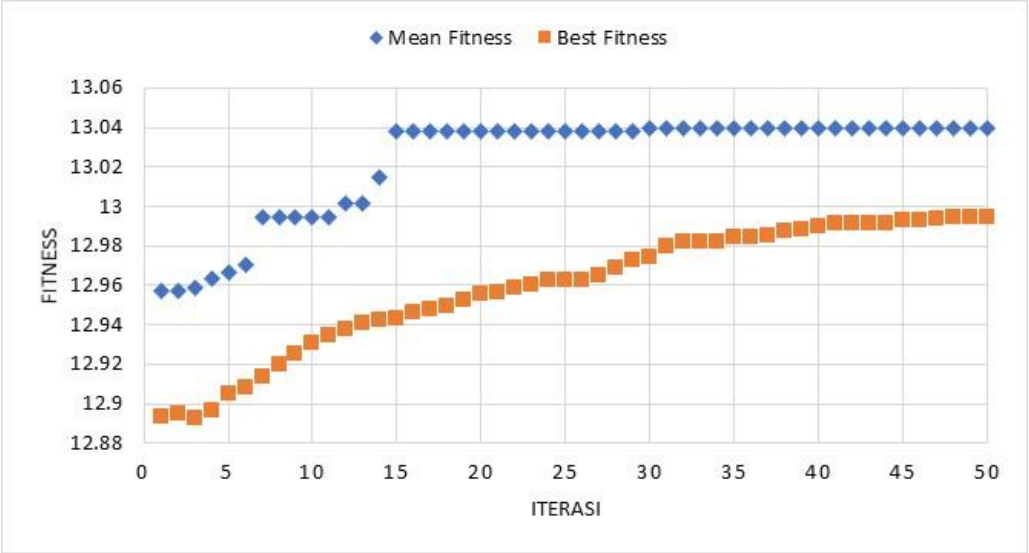
Gambar 4.9 Ilustrasi Pergerakkan Partikel pada Iterasi kedua Algoritma PSO

Setiap partikel akan berpindah sebanyak 50 iterasi. Pada setiap perpindahan, tiap partikel menyimpan nilai *fitness* terbaik yang disebut sebagai *Pbest* dan membawa informasi tersebut pada partikel lainnya sehingga nilai *fitness* terbaik untuk keseluruhan partikel dapat ditentukan yang disebut sebagai *Gbest*. Nilai *Pbest* dan *Gbest* akan selalu diperbarui dan menentukan kecepatan untuk mendapatkan posisi terbaru hingga konvergensi tercapai. Setelah kekonvergenan tercapai. akan dilakukan pemodelan dan dihasilkan nilai ramalan yang dapat dibandingkan dengan data *testing*. Pemodelan dilakukan dengan menjalankan beberapa kemungkinan nilai *inertia weight* w. Pemilihan w optimum akan didasarkan pada nilai RMSE seperti ditujukkan pada Tabel 4.6

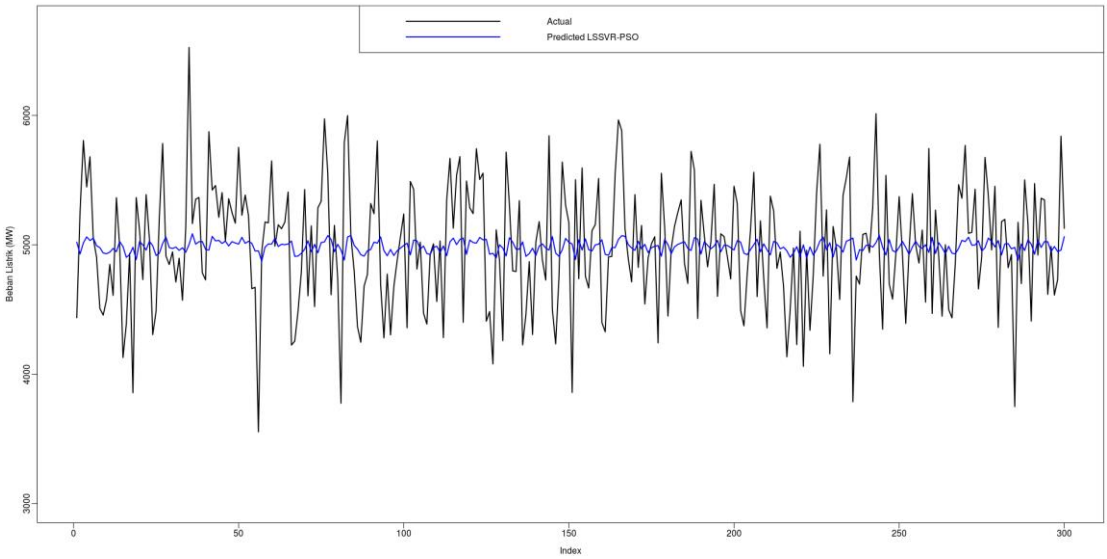
Tabel 4.6 Hasil Nilai RMSE pada Simulasi *Inertia Weight*

	$w = 0{,}8$	$w = 0{,}9$	$w = 1$	$w = 1{,}1$	$w = 1{,}2$
RMSE	471,9895	471,9883	473,3035	473,3157	475,2527

Berdasarkan Tabel 4.6 didapat nilai w optimum sebesar 0,9. Adapun hasil ringkasan nilai *Gbest* ditiap iterasi ditampilkan seperti Gambar 4.10



Gambar 4.10 Performa Algoritma PSO untuk Data AR(1)



Gambar 4.11 Plot *Actual* vs *Predicted* LSSVR-*Particle Swarm Optimization* Data Testing AR(1)

Pada Gambar 4.10 menunjukkan konvergensi dari *Particle Swarm Optimization* dicapai pada iterasi ke-50 dengan nilai rata-rata *fitness* sebesar 13,02322 dan *fitness* tertinggi sebesar 12,995. Berdasarkan nilai *fitness* maksimum, didapatkan parameter optimal sebesar 9,528 dan 3,9187 untuk σ^2 dan C dengan nilai RMSE

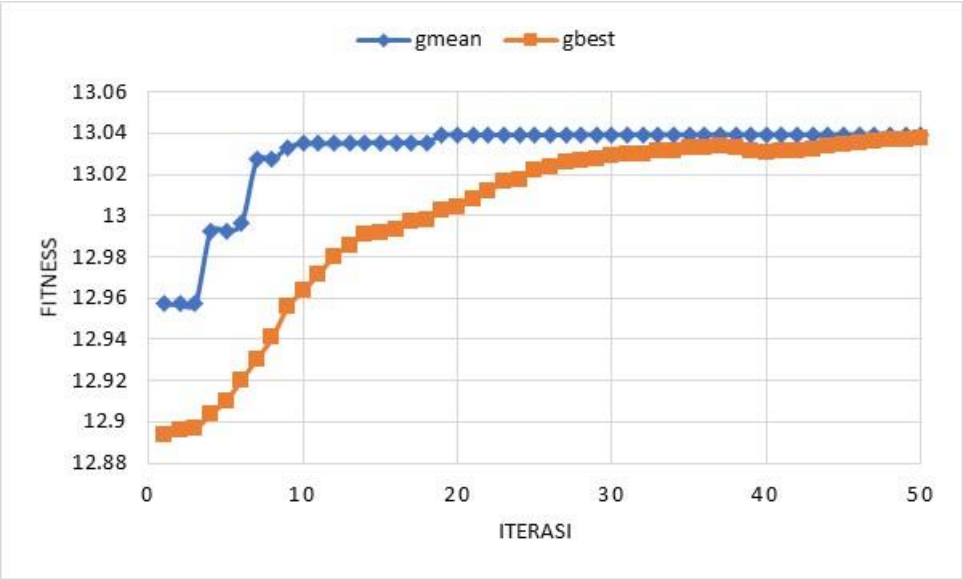
sebesar 471,9883. Berdasarkan Gambar 4.11 terlihat bahwa perbandingan plot data *testing* dengan hasil prediksi dari model LSSVR-PSO menunjukkan hasil yang tidak baik karena nilai ramalan yang didapat tidak mendekati nilai asli. Nilai sMAPE dan RMSE yang dihasilkan adalah sebesar 7,67% dan 471,9883.

Selain itu, data juga akan diramalkan menggunakan metode optimasi MPSO untuk mendapatkan nilai parameter yang optimum pada model LSSVR dan diharapkan model yang didapat dapat lebih baik dibandingkan menggunakan metode PSO, sebab algoritma PSO rawan terhadap terjadinya kekonvergenan prematur. Algoritma MPSO menambahkan parameter *discrete degree* $D(t)$ dan σ^2 yang mampu meningkatkan probabilitas penemuan solusi global optimal dan terhindar dari lokal optimal. Secara umum, kedua parameter tersebut mengontrol nilai posisi partikel agar sesuai dengan kriteria yang ditentukan dan akan membangkitkan dengan nilai lain sedemikan hingga kriteria tersebut terpenuhi. Adapun langkah yang dilakukan yaitu sebagai berikut

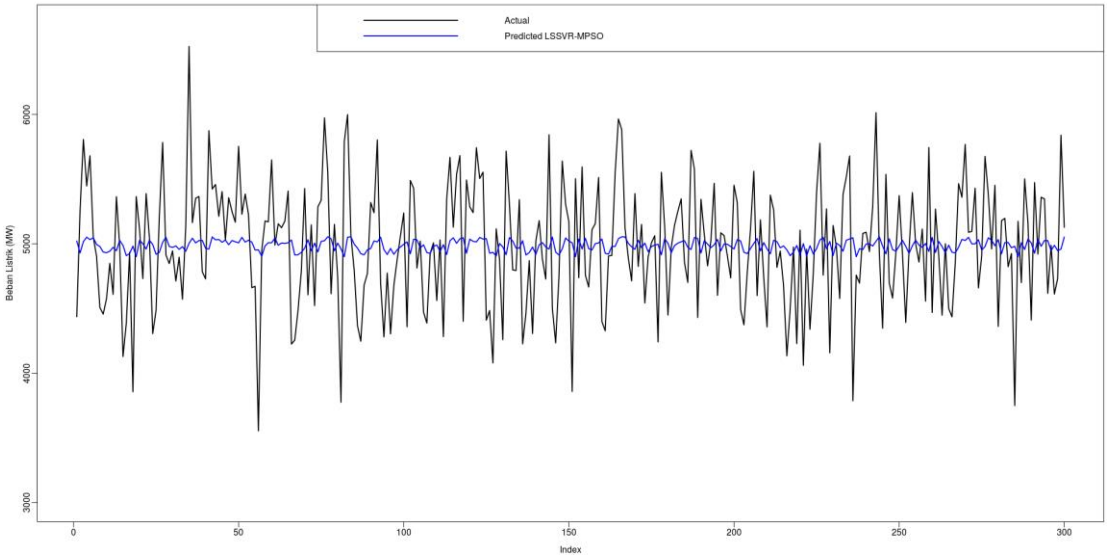
1. Definisikan parameter MPSO seperti definisi parameter PSO dimana nilai parameter $w = 0{,}9$ sesuai dengan hasil dari simulasi PSO, inisiasi vektor posisi dan vektor kecepatan seperti inisiasi keduanya pada PSO, hitung nilai *fitness*, *Pbest*, dan *Gbest* sebagaimana langkah pada algoritma PSO.
2. Hitung nilai $D(t)$ dan σ^2 seperti pada persamaan (2.39) dan (2.40)
3. Menentukan nilai $\alpha = 0{,}5,\ L = 10,\ \beta = 0{,}05$
4. Evaluasi nilai $D(t)$ dan σ^2 dengan kriteria jika $D(t) < \alpha$ atau $\sigma^2 < \beta$ maka inisiasi kembali nilai vektor posisi, jika tidak maka lakukan proses PSO untuk memperbarui nilai posisi dan kecepatan partikel.
5. Iterasi akan terus dilakukan dengan mengevaluasi kembali nilai $D(t)$ dan σ^2. Jika $D(t) < \alpha$ atau $\sigma^2 < \beta$ maka bangkitkan data vektor posisi terbaru dan evaluasi kembali hingga didapat nilai $D(t) > \alpha$ dan $\sigma^2 > \beta$
6. Proses perhitungan PSO terkoreksi nilai $D(t)$ dan σ^2 akan terus dilakukan hingga kekonvergenan terpenuhi.

Performa hasil MPSO disajikan pada Gambar 4.12, terlihat bahwa kekonvergenan dari *Modified Particle Swarm Optimization* dicapai pada iterasi ke-50 dengan nilai rata-rata

fitness sebesar 13,3036 dan *fitness* tertinggi sebesar 13,03736. Berdasarkan nilai *fitness* maksimum, didapatkan parameter optimal sebesar 9,999 dan 0,02415 untuk σ^2 dan C dengan nilai RMSE sebesar 471,9486.



Gambar 4.12 Performa Algoritma MPSO untuk Data AR(1)



Gambar 4.13 Plot *Actual* vs *Predicted* LSSVR-*Modified Particle Swarm Optimization* Data Testing AR(1)

Berdasarkan Gambar 4.13 terlihat bahwa perbandingan plot data *testing* dengan hasil prediksi dari model LSSVR-MPSO menunjukkan hasil yang tidak baik karena nilai ramalan yang didapat tidak mendekati nilai asli. Nilai sMAPE dan RMSE yang dihasilkan adalah sebesar 7,67% dan 471,9486. Penjabaran ilustrasi algoritma optimasi untuk data AR(1) diatas dapat diringkas seperti pada Tabel 4.7

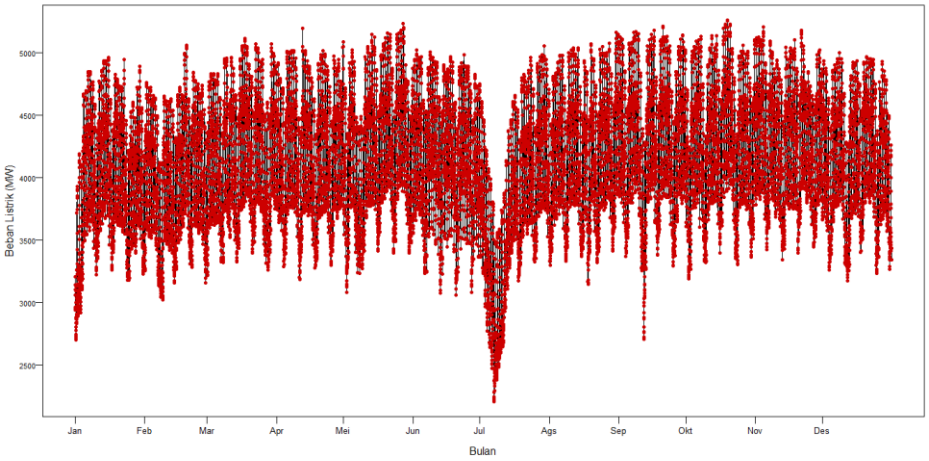
Tabel 4.7 Ringkasan Hasil Performa LSSVR pada Data AR(1)

	LSSVR	LSSVR-GA	LSSVR-PSO	LSSVR-MPSO
RMSE	488,8115	475,2527	471,9883	471,9486
sMAPE	7,915%	7,731%	7,669%	7,6711%

Secara umum, keempat metode tidak menghasilkan hasil peramalan yang baik sebab nilai yang didapat jauh dari data aslinya, terlihat pada plot data ramalan masing-masing metode. Tetapi, jika dilihat berdasarkan RMSE minimum maka dapat disimpulkan metode optimasi MPSO mampu mengoptimalkan hasil peramalan metode LSSVR lebih baik dibandingkan dengan metode optimasi GA dan PSO. Hal tersebut terlihat dari pola data pada MPSO mampu menangkan tren naik turun seperti data aslinya. Hasil peramalan LSSVR yang tidak baik untuk meramalkan data AR(1) mendukung pernyataan Wijaya (2017) dimana ada beberapa kasus data yang berpola linier ketika dimodelkan dengan metode nonlinier berbasis kecerdasan buatan justru menghasilkan tidak menghasilkan hasil yang baik, dimana metode berbasis kecerdasan buatan terkenal dengan fleksibilitasnya.

4.2 Analisis Karakteristik Beban Listrik Jangka Pendek Area Jawa Timur

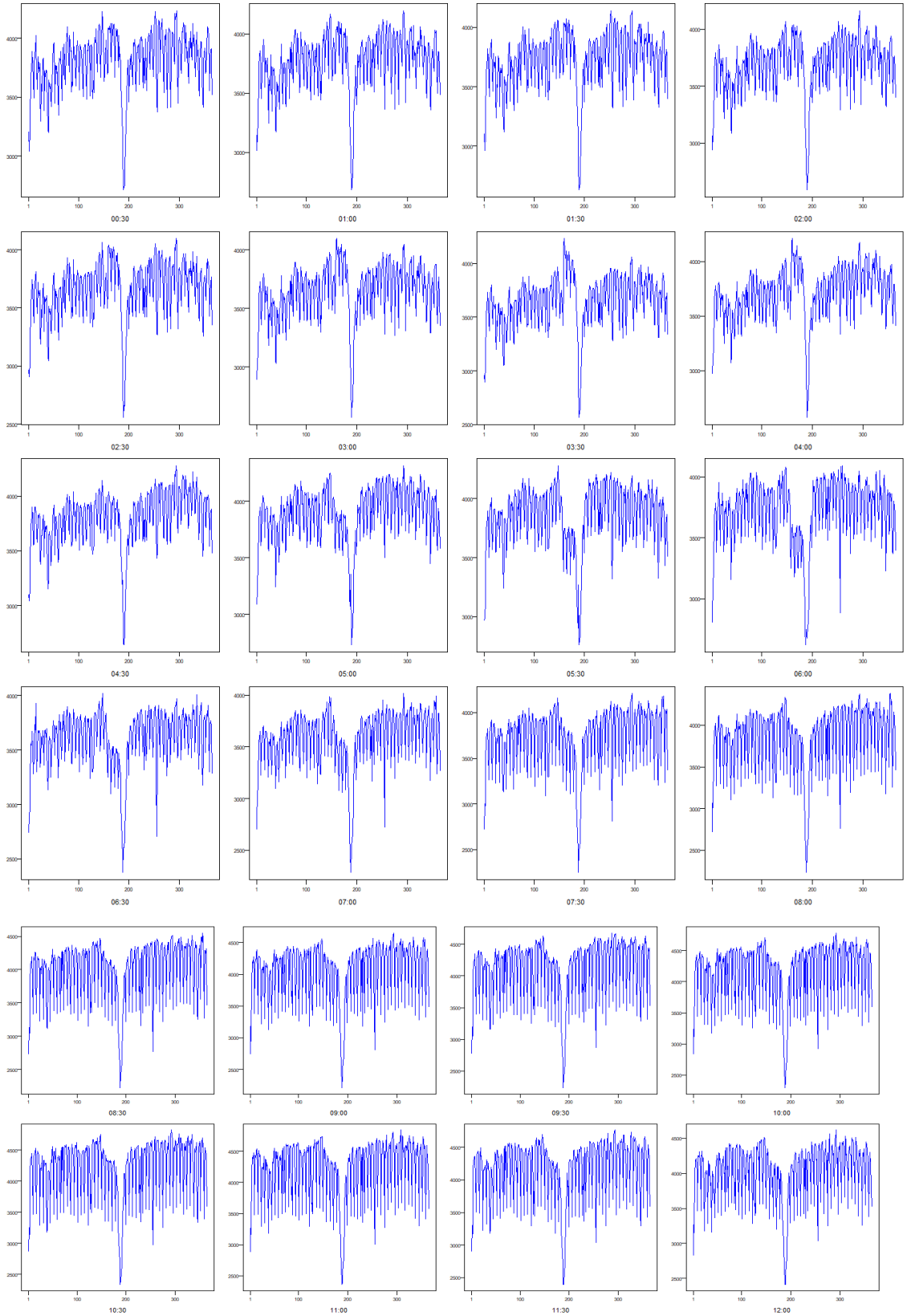
Karakteristik permintaan beban listrik regional Jawa Timur didapat melalui analisis statistika deskriptif dengan mengeksplorasi informasi yang terkandung dalam data tanpa melakukan inferensi. Melalui tampilan *time series plot*, secara visual dapat diketahui pola *seasonal*, tren, ataupun stasioneritas data dalam mean dan varians.

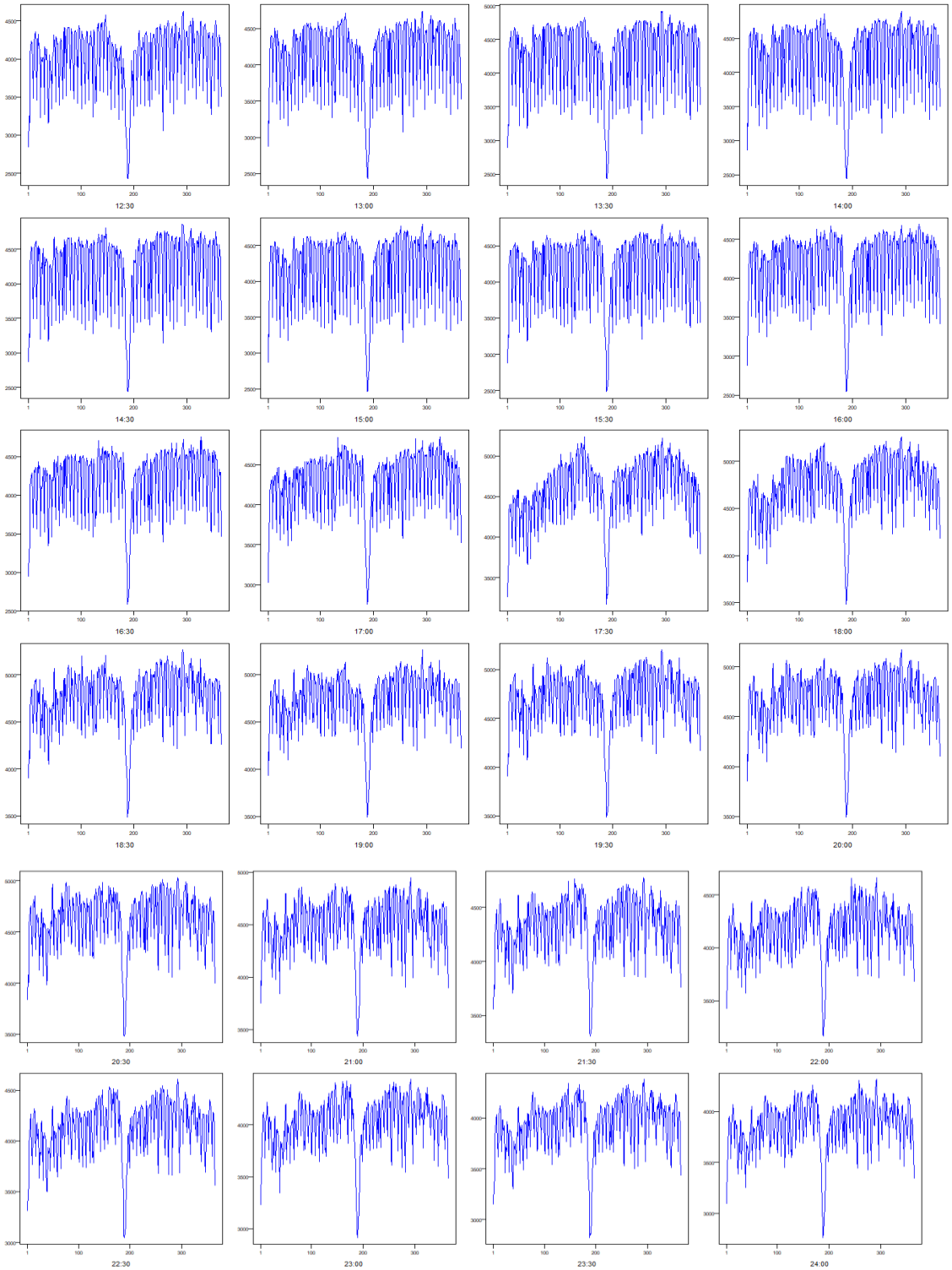


Gambar 4.14 Time Series Plot Beban Konsumsi Listrik

Gambar 4.14 menunjukkan plot permintaan beban listrik yang dicatat per setengah jam dari tanggal 1 Januari hingga 31 Desember 2016. Secara visual terlihat bahwa terjadi penurunan beban listrik yang cukup signifikan pada akhir Juni hingga pertengahan Juli. Fenomena tersebut terjadi sekitar 10 hari sebelum hingga sesudah Idul Fitri, mengingat pada momen tersebut merupakan hari cuti bersama sehingga sebagian industri tidak beroperasi. Akibatnya, beban konsumsi listrik mencapai puncak terendah. Selanjutnya, tampilan *time series plot* tiap setengah jam diperlukan untuk mengeksplorasi pola konsumsi listrik masyarakat.

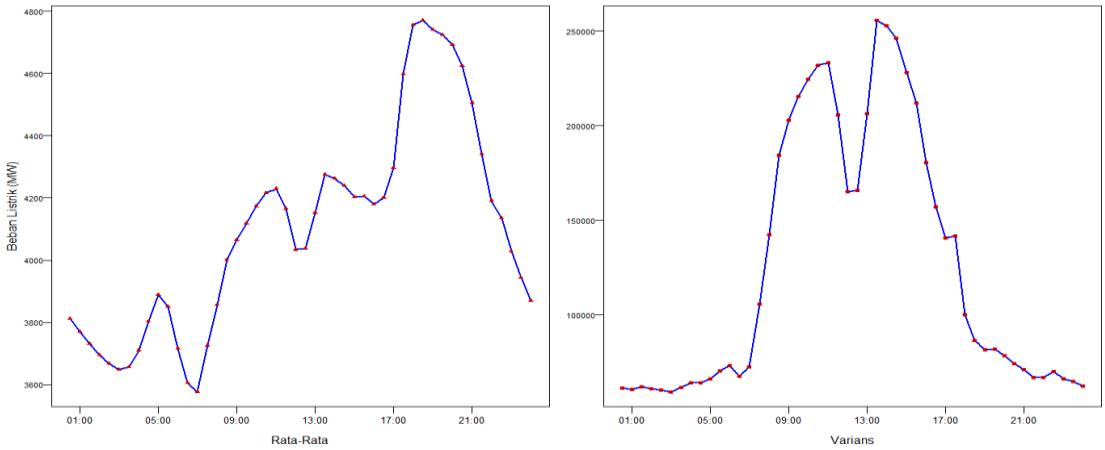
Berdasarkan Gambar 4.15 terlihat bahwa beban listrik memiliki fluktuasi yang tinggi dimulai dari pukul 08:00 seiring dengan dimulainya aktivitas masyarakat hingga 18:00 seiring berakhirnya aktivitas utama untuk bekerja. Fluktuasi konsumsi beban listrik yang tinggi di siang hari disebabkan oleh mayoritas konsumsi dari sektor industri yang beroperasi. Adapun pada malam hingga dini hari, fluktuasi beban listrik yang terjadi tidak begitu tinggi sebab penggunaan beban listrik pada waktu tersebut didominasi oleh sektor rumah tangga.


00:30
01:00
01:30
02:00
02:30
03:00
03:30
04:00
04:30
05:00
05:30
06:00
06:30
07:00
07:30
08:00
08:30
09:00
09:30
10:00
10:30
11:00
11:30
12:00



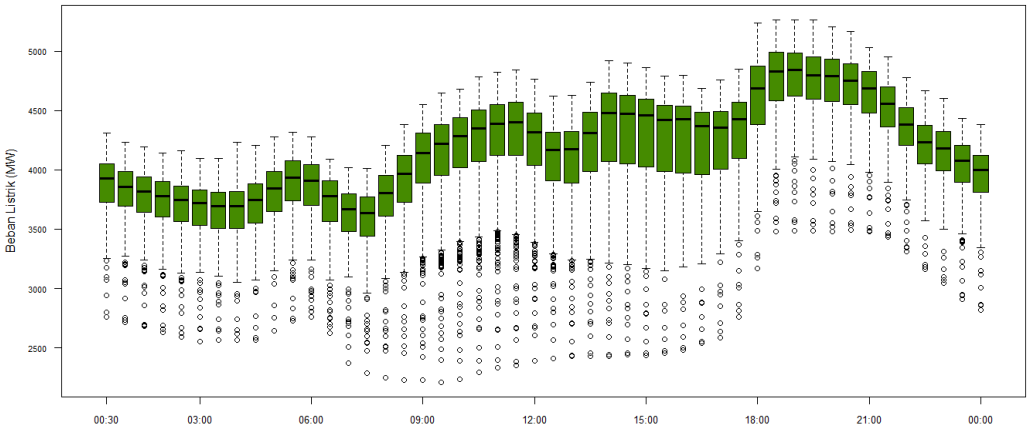
Gambar 4.15 Pola Konsumsi Beban Listrik Per Setengah Jam

Pola konsumsi beban listrik masyarakat secara ringkas dapat divisualisasikan berdasarkan rata-rata dan varians di tiap setengah jam seperti pada Gambar 4.16.



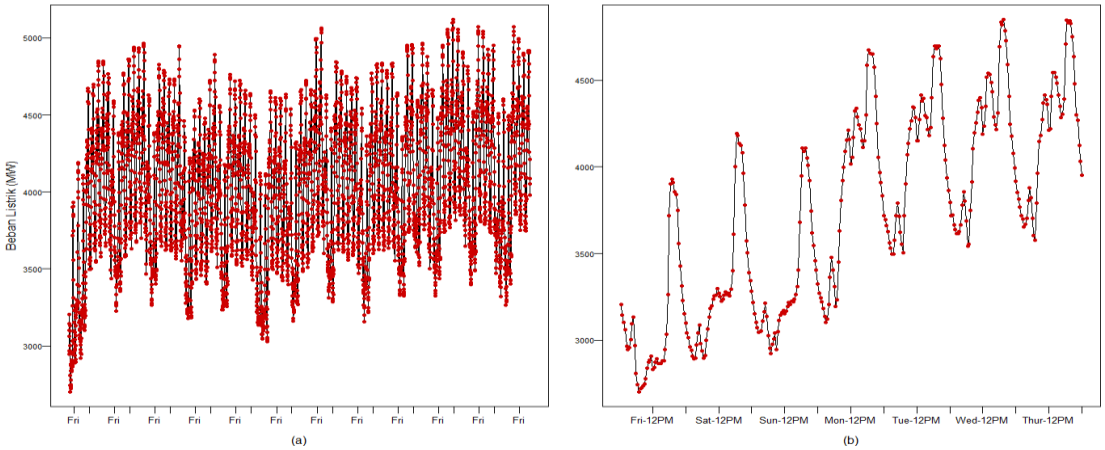
Gambar 4.16 Plot Rata-Rata dan Varians Beban Listrik Per Setengah Jam

Sebagaimana yang dapat diamati pada Gambar 4.16, rata-rata konsumsi beban listrik terendah terjadi pada pukul 07:00 seiring dengan dimulainya aktivitas masyarakat untuk bekerja di luar rumah. Sebagian besar alat elektronik pada sektor rumah tangga dimatikan sebab masyarakat bersiap untuk bekerja. Pada pukul 12:00 hingga 13:00, rata-rata beban listrik terlihat turun yang menandakan sebagian alat produksi tidak beroperasi di waktu istirahat kerja kemudian kembali naik seiring dengan dimulainya kembali aktivitas bekerja. Rata-rata konsumsi beban listrik mencapai puncak tertinggi pada pukul 18:30 seiring dengan berakhirnya kegiatan industri dan aktivitas bekerja di luar rumah. Setelahnya, rata-rata beban listrik berangsur turun seiring dengan masuknya jam istirahat masyarakat serta instansi yang tidak beroperasi di malam hari. Selain itu, berdasarkan Gambar 4.3 diketahui bahwa pada siang hari konsumsi beban listrik memiliki variansi yang tinggi. Hal tersebut berkaitan dengan pola konsumsi yang didominasi oleh sektor industri. Sedangkan pada malam hari, tidak terjadi fluktuasi konsumsi beban listrik yang tinggi sebab penggunaan beban listrik didominasi oleh sektor rumah tangga.



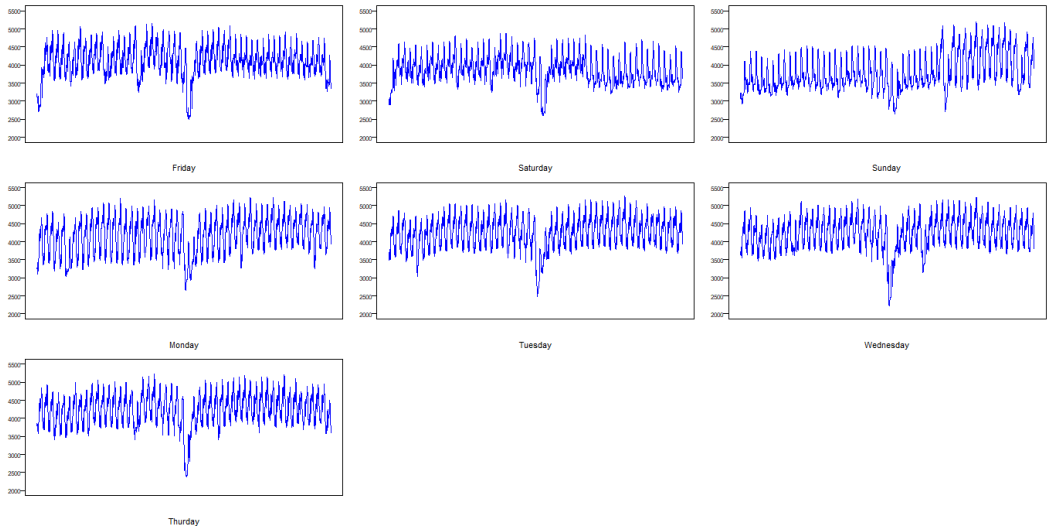
Gambar 4.17 Boxplot Beban Listrik Per Setengah Jam

Gambar 4.17 menunjukkan bahwa data beban konsumsi listrik per setengah jam memiliki banyak *outlier*, terutama bernilai kurang dari interval kepercayaan. Fenomena ini mengindikasikan beban konsumsi masyarakat lebih sering mengalami beban puncak terendah dibandingkan dengan beban puncak tertinggi. Sebagaimana Gambar 4.16, adanya variansi konsumsi beban listrik yang tinggi yang dimulai pukul 09:00 hingga 14:00 tidak hanya disebabkan oleh fluktuasi data yang tinggi, melainkan turut disumbang oleh banyaknya *outlier*. Hal tersebut diduga disebabkan oleh adanya kesalahan pada pembangkit listrik sehingga jumlah beban listrik yang didistribusikan menjadi berkurang/bertambah jauh dari jumlah beban listrik yang sewajarnya.



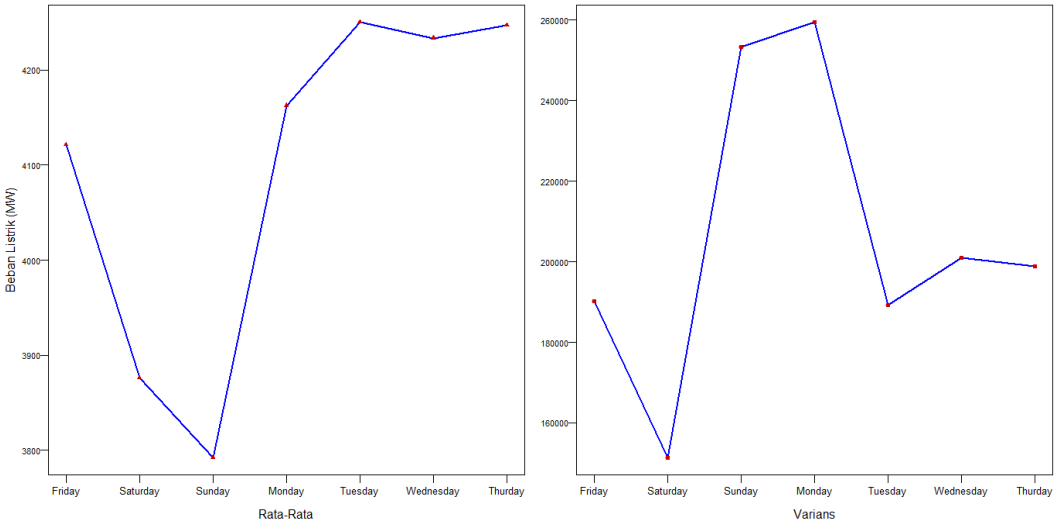
Gambar 4.18 Plot Musiman Beban Listrik Regional Jawa Timur

Adapun karakteristik pola *seasonal* dari data ditampilkan pada Gambar 4.18(a) yang menunjukkan besarnya konsumsi listrik dari tanggal 1 Januari hingga 31 Maret 2016. Hal tersebut menggambarkan pola *seasonal* mingguan dimana pada hari sabtu dan minggu konsumsi beban listrik lebih rendah dibandingkan dengan saat hari aktif, sebab disumbang oleh konsumsi industri yang beroperasi pada hari tersebut. Sementara itu, Gambar 4.18(b) menunjukkan besarnya konsumsi beban listrik pada tanggal 1 Januari hingga 7 Januari 2016 yang menggambarkan adanya pola konsumsi listrik yang rendah pada malam hingga dini hari dan kemudian terjadi peningkatan pada pagi hingga sore hari. Fenomena tersebut menunjukkan dugaan adanya pola *seasonal* harian pada data.



Gambar 4.19 Time Series Plot Konsumsi Beban Listrik Per Hari

Konsumsi beban listrik berdasarkan hari aktif ataupun hari libur kerja ditampilkan pada Gambar 4.19. Secara umum, plot tersebut menunjukkan beban konsumsi listrik yang dicatat tiap setengah jam menunjukkan data tidak stasioner dalam mean dan varians. Pada hari aktif, terlihat beban listrik memiliki fluktuasi yang tinggi, sedangkan pada hari libur kerja seperti Sabtu dan Minggu, plot beban listrik terlihat tidak berfluktuasi tinggi.

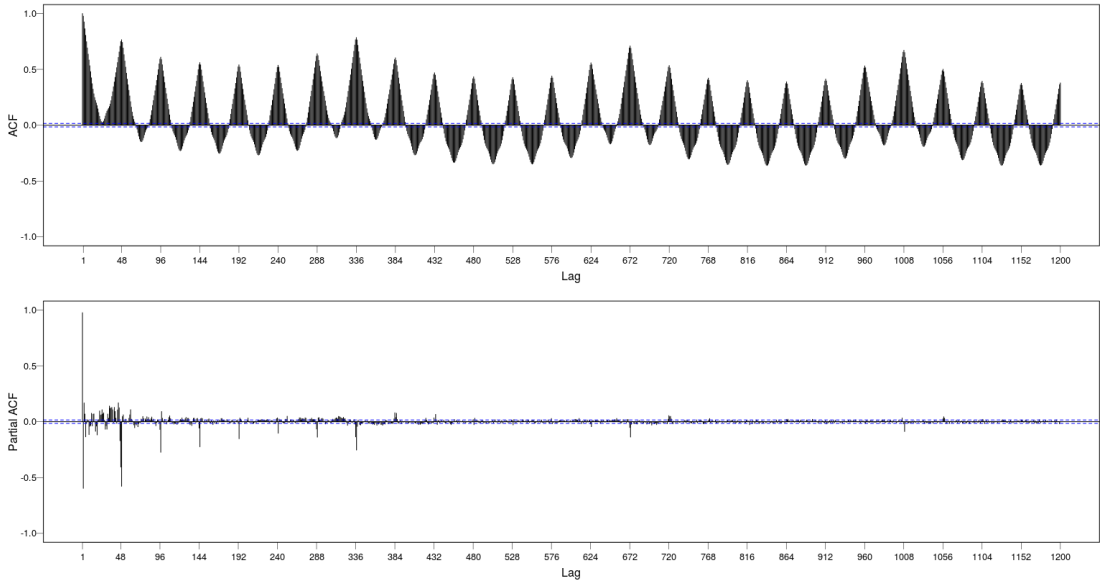


Gambar 4.20 Plot Rata-Rata dan Varians Konsumsi Beban Listrik Per Hari

Sebagaimana dapat diamati pada Gambar 4.20, rata-rata beban konsumsi listrik terendah terjadi pada hari Sabtu dan Minggu. Pada hari libur kerja, pola konsumsi listrik didominasi oleh sektor rumah tangga dimana pola konsumsi energi listrik cenderung stabil sesuai dengan gaya hidup masing-masing keluarga. Sedangkan pada hari aktif kerja, pola konsumsi beban listrik disumbang oleh sektor industri, instansi, dan fasilitas umum yang beroperasi sehingga pola konsumsi beban listrik cukup bervariasi. Selain itu, terlihat bahwa varians beban listrik mencapai puncaknya saat hari Senin yang merupakan hari pertama masuk kerja. Namun, pada hari Minggu terjadi anomali dimana rata-rata konsumsi listrik merupakan yang terendah tetapi memiliki variansi yang cukup besar. Seperti pada Gambar 4.19, pada hari Minggu terlihat pola fluktuasi yang tinggi di sekitar akhir tahun, dimana pada momen tersebut mayoritas masyarakat cenderung berpergian pada tempat wisata ataupun tempat hiburan sehingga terjadi variasi konsumsi beban listrik yang berfluktuatif pada sektor tersebut. Berdasarkan informasi diatas diperkirakan terdapat variasi harian (per 48 jam) dan variasi mingguan (per 336 jam) yang selanjutnya akan digunakan pada pemodelan.

4.3 Estimasi Parameter ARIMA

Pemodelan dengan ARIMA memerlukan identifikasi kestasioneran data dalam mean dan varians. Identifikasi stasioneritas mean dapat dilihat secara visual berdasarkan time series plot data seperti pada Gambar 4.14 yang terlihat bahwa data tidak stasioner dalam *mean*, sebab terdapat pola tren naik turun. Identifikasi juga dapat dilakukan melalui plot ACF. Plot ACF yang memiliki pola turun sangat lambat mengindikasikan bahwa data tidak stasioner dalam mean sehingga perlu dilakukan *differencing*.

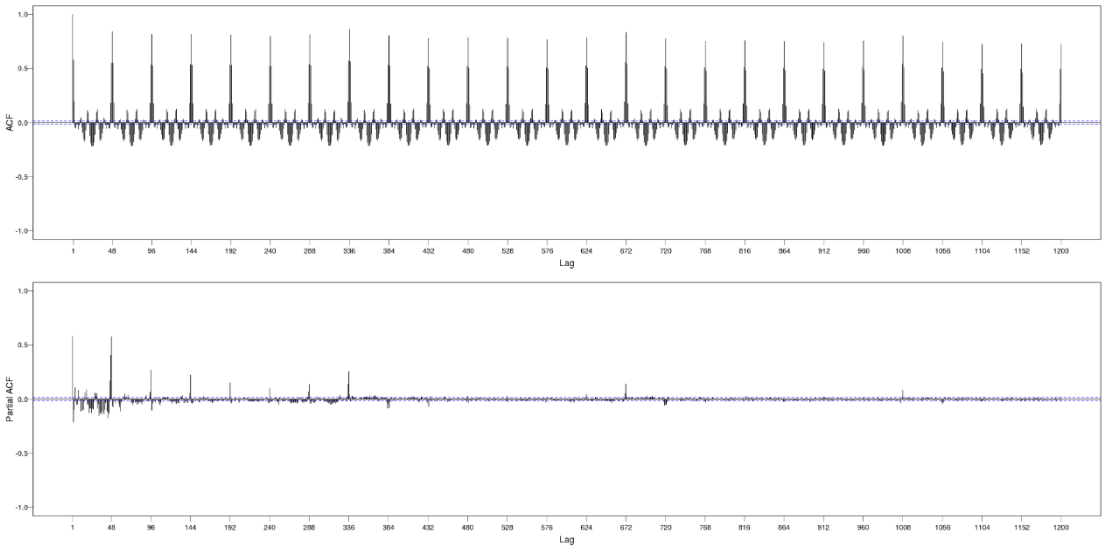


Gambar 4.21 Plot ACF dan PACF Data Beban Listrik

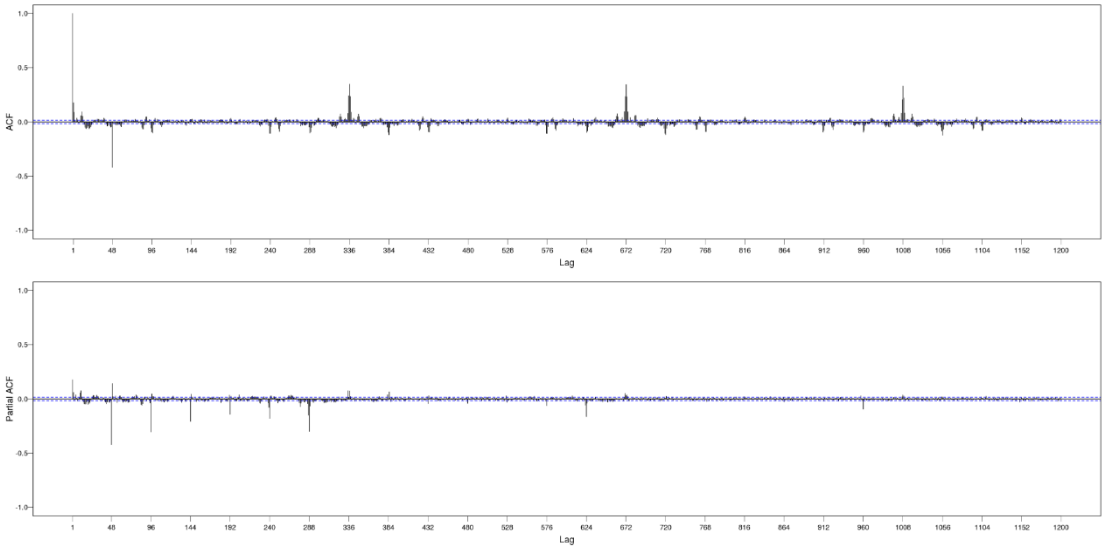
Berdasarkan Gambar 4.21 diketahui bahwa nilai ACF lag pertama bernilai mendekati satu dan turun sangat lambat sehingga mengindikasikan data beban listrik per setengah jam area Jawa Timur tidak stasioner dalam mean, sehingga data harus dilakukan *differencing*. Proses *differencing* akan dilakukan pada lag 1, hal tersebut diperkuat dengan pola PACF yang bernilai mendekati 1 pada lag 1.

Setelah data dilakukan *differencing* pada lag 1, terlihat bahwa pada Gambar 4.22 plot ACF berulang pada kelipatan ke-48. Terlebih plot PACF pada Gambar 4.22 juga menunjukkan lag signifikan tertinggi berada pada kelipatan 48. Hal tersebut mengindikasikan bahwa data beban listrik per setengah jam mempunyai pola musiman

harian atau per 48 jam. Selanjutnya, data akan di *differencing* pada lag ke-48 dan plot ACF PACF yang didapat disajikan pada Gambar 4.23



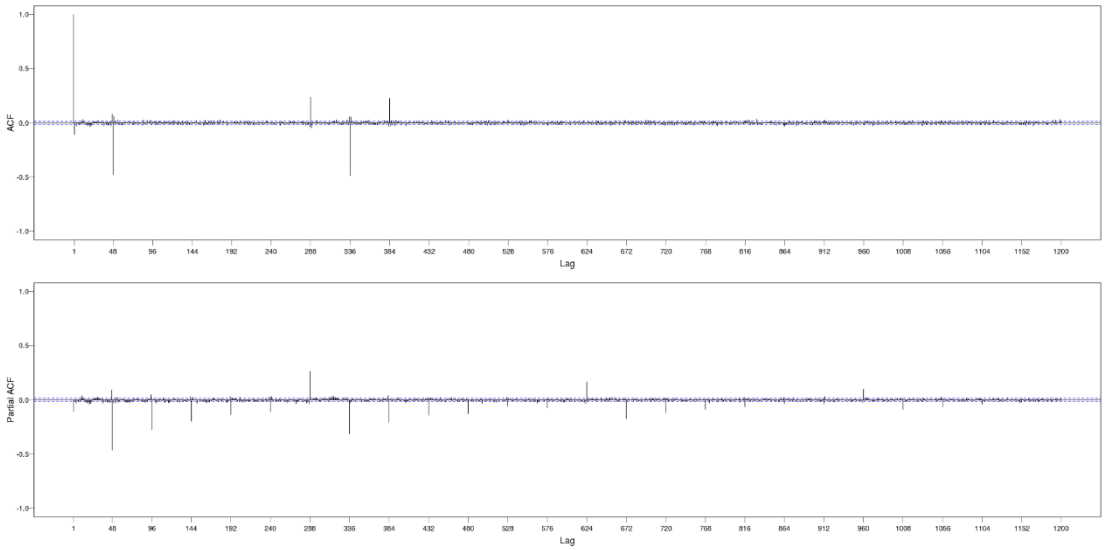
Gambar 4.22 Plot ACF dan PACF Data Beban Listrik *Differencing* 1



Gambar 4.23 Plot ACF dan PACF Data Beban Listrik *Differencing* 48

Data beban listrik jangka pendek area Jawa Timur yang telah dilakukan *differencing* pada lag 1 dan lag 48 selanjutnya akan dilakukan *differencing* pada lag 336. Hal tersebut mengacu sebagaimana Gambar 4.23 terlihat bahwa masih terdapat

pola berulang pada plot ACF yaitu pada kelipatan 336. Selain itu, pada plot PACF terlihat nilai signifikansi yang tinggi pada kelipatan ke-336. Hal tersebut mengindikasikan data beban listrik jangka pendek area Jawa Timur juga mempunyai pola musiman mingguan.



Gambar 4.24 Plot ACF dan PACF PACF Data Beban Listrik *Differencing* 336

Pada Gambar 4.24 terlihat bahwa plot ACF dan PACF telah mengalami *cut off* sehingga mengindikasikan bahwa data telah stasioner. Hal tersebut sesuai dengan kriteria seperti pada Tabel 2.1. Adanya dua pola musiman menandakan bahwa model yang terbentuk merupakan *double seasonal* ARIMA (DSARIMA). Berdasarkan plot ACF dan PACF data yang telah stasioner, diperoleh dua model dugaan yaitu

$$\begin{aligned} &ARIMA([28,31,47],1,[11,12,31])(0,1,1)^{48}(0,1,2)^{336} \\ &ARIMA([28,47],1,[11,12])(0,1,1)^{48}(0,1,2)^{336} \end{aligned} \tag{4.1}$$

Adapun hasil estimasi parameter ketiga model DSARIMA dapat diketahui pada Tabel 2.1. Semua nilai *p-value* bernilai kurang dari taraf signifikansi 0,05 menunjukkan bahwa semua parameter telah signifikan. Penentuan model terbaik pada penelitian ini dibatasi tidak menguji asumsi residual pada model DSARIMA, sehingga model terbaik akan ditentukan berdasarkan nilai RMSE minimum.

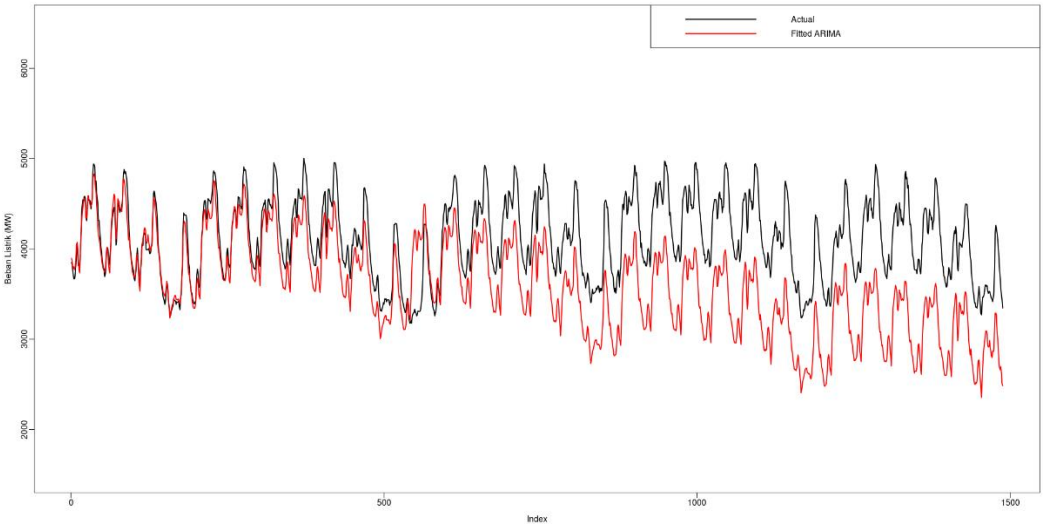
Tabel 4.8 Nilai Estimasi Parameter Model DSARIMA

	Model 1		Model 2	
	Est	p-value	Est	p-value
θ_{11}	-0,0519	<.0001	-0,0525	<.0001
θ_{12}	-0,0292	0.0002	-0,0274	0.0006
θ_{31}	-0,2297	0.0012	-	-
$\Theta_{1(48)}$	0,7072	<.0001	0.7057	<.0001
$\Theta_{2(336)}$	0,7862	<.0001	0.78644	<.0001
$\Theta_{3(672)}$	-0,0311	0.0002	-0.03102	0.0002
ϕ_{28}	-0,0199	0.0103	-0.0192	0.0167
ϕ_{31}	-0,2403	0.0006	-	-
ϕ_{47}	0,0783	<.0001	0.08006	<.0001

Kedua model memiliki parameter yang signifikan, sehingga langkah selanjutnya yaitu meramalkan hingga 1488 setengah jam kedepan. Hasil ramalan ini kemudian dibandingkan dengan data *testing* sehingga diperoleh nilai RMSE untuk model pertama dan model kedua masing-masing sebesar 618,1697 dan 646,4649 dengan nilai sMAPE masing-masing yaitu 13,76% dan 14,48%. Apabila data beban listrik jangka pendek area Jawa Timur dilambangkan sebagai Z_t maka model DSARIMA terbaik dapat dituliskan sebagai berikut

$$\begin{aligned}&\left(1-\phi_{28}B^{28}-\phi_{31}B^{31}-\phi_{47}B^{47}\right)\left(1-B\right)\left(1-B^{48}\right)\left(1-B^{336}\right)\left(1-B^{672}\right)Z_t\\&=\left(1-\theta_{11}B^{11}-\theta_{12}B^{12}-\theta_{31}B^{31}\right)\left(1-\Theta B^{48}\right)\left(1-\Theta B^{336}\right)\left(1-\Theta B^{672}\right)a_t\end{aligned} \tag{4.2}$$

Hasil peramalan selama 1488 setengah jam kedepan berdasarkan model pada persamaan (4.2) divisualisasikan seperti pada Gambar 4.25



Gambar 4.25 Plot *Actual* vs *Predicted* DSARIMA

Dari Gambar 4.25 terlihat bahwa hasil peramalan yang didapat tidak baik sebab plot data hasil peramalan jauh dari data asli, walaupun pola naik turun trend data sudah mampu diakomodasi oleh DSARIMA. Hal tersebut mengindikasikan bahwa pada data beban listrik jangka pendek area Jawa Timur diduga tidak hanya berpola musiman harian dan mingguan, tetapi juga terdapat pola nonlinier yang belum bisa ditangkap oleh metode DSARIMA.

4.4 Optimasi *Grid Search*

Langkah pertama yang perlu dilakukan untuk memodelkan SVR dan LSSVR adalah mendefinisikan variabel X sebagai variabel independen. Pada penelitian ini, inputan variabel yang digunakan merupakan lag-lag yang signifikan dari model ARIMA. Lag tersebut didapatkan dengan cara menjabarkan model ARIMA yang telah didapat. Adapun penjabaran model ARIMA pada persamaan (4.2) terlampir pada Lampiran 1. Pemilihan lag signifikan berfokus pada model AR sehingga akan didapat lag data. Dari persamaan model telah dijabarkan, diperoleh sebanyak 111 lag data signifikan yang kemudian data lag tersebut akan dijadikan sebagai variabel independen untuk analisis SVR dan LSSVR. Langkah selanjutnya, menyusun data variabel X agar siap dianalisis. Penyusunan dilakukan dengan menghapus beberapa data awal sejumlah

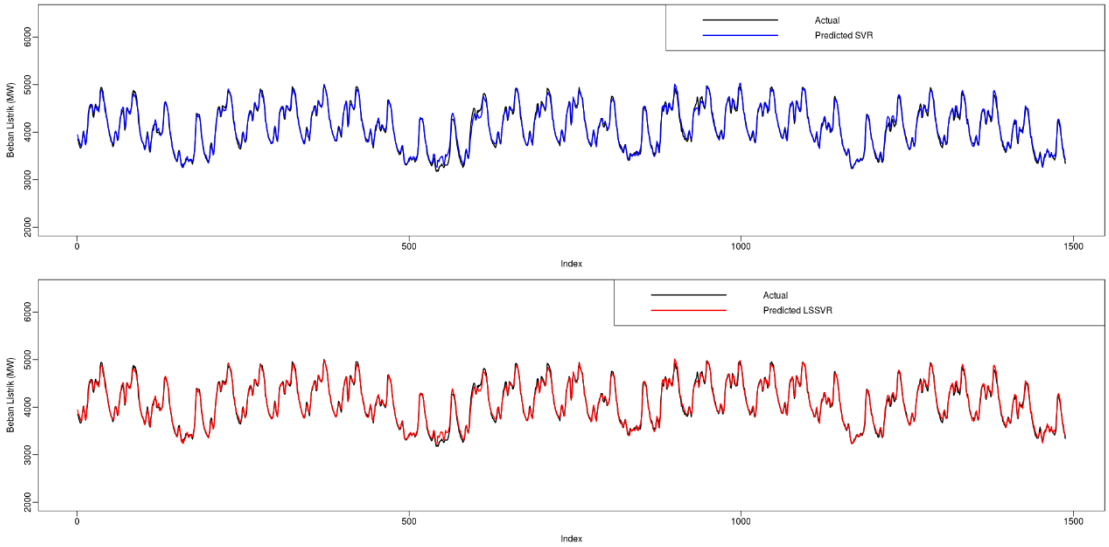
dengan nilai lag terbesar. Pada model persamaan (4.2) didapat lag yang terbesar yaitu 1163, sehingga pada 1163 data pertama akan bernilai kosong (NAN).

Pemilihan parameter optimal merupakan hal terpenting pada metode SVR dan LSSVR agar didapat hasil ramalan yang akurat. Salah satu metode optimasi yaitu *Grid Search* yang dilakukan dengan cara memembagi jangkauan parameter yang akan dioptimalkan kedalam grid dan melintasi semua titik untuk mendapatkan parameter yang optimal. Berbagai kemungkinan parameter tersebut akan dievaluasi menggunakan kriteria RMSE minimum.

Pada model SVR, terdapat tiga inputan parameter yaitu parameter cost (C), epsilon (ε) dan parameter fungsi kernel RBF (σ^2). Sedangkan pada model LSSVR, terdapat dua inputan parameter yaitu parameter cost (C) dan parameter fungsi kernel RBF (σ^2). Metode *Grid Search* membagi range parameter yang dioptimalkan ke dalam grid dan melintasi semua titik untuk mendapatkan parameter yang optimal. Range parameter yang digunakan untuk model SVR adalah $\{(1e-2,1),(1e-4,10),(1e-2,50)\}$ untuk ε, C, dan σ^2, sedangkan pada model LSSVR adalah $\{(50,200),(1,25.75,50.5,75.25,100)\}$ untuk σ^2 dan C. Setelah didapat model, maka akan dilakukan pemodelan selama satu bulan kedepan. Hasil ramalan dibandingkan dengan data *testing* dan dihitung nilai akurasinya terlampir pada Lampiran 13. Kriteria akurasi yang digunakan yaitu RMSE minimum berdasarkan hasil ramalan dan parameter SVR optimal yang didapat untuk ε, C, dan σ^2 yaitu 0.01, 10, dan 0.01, sedangkan parameter LSSVR optimal yang didapat untuk σ^2 dan C yaitu sebesar 50 dan 25.75. Adapun perbandingan akurasi pada data *testing* terlihat pada Tabel 4.9

Tabel 4.9 Perbandingan Akurasi Optimasi Grid Search

SVR		LSSVR	
RMSE	sMAPE	RMSE	sMAPE
58.11766134	0.01062289	**51.114845812**	0.009479884



Gambar 4.26 Plot *Actual* vs *Predicted* SVR dan LSSVR

Berdasarkan Gambar 4.26 terlihat bahwa perbandingan plot data *testing* dengan hasil prediksi dari model SVR dan LSSVR menunjukkan hasil yang baik karena nilai ramalan yang didapat mendekati nilai asli, begitu juga dengan pola tren naik turun data yang dapat ditangkap oleh hasil estimasi model. Hasil akurasi pada Tabel 4.9 menunjukkan bahwa berdasarkan RMSE minimum didapat metode LSSVR memiliki akurasi yang lebih baik dengan nilai sMAPE dan RMSE yang dihasilkan adalah sebesar 0,945% dan 51,115. Jadi dapat disimpulkan bahwa LSSVR mempunyai akurasi yang lebih baik dibandingkan dengan metode SVR ataupun DSARIMA dalam meramalkan data beban listrik jangka pendek area Jawa Timur.

4.5 Optimasi Genetic Algorithm

Strategi dalam pemodelan SVR ataupun LSSVR agar diperoleh hasil peramalan yang baik yaitu dengan mendapatkan nilai parameter optimal. Metode optimasi yang biasa digunakan untuk mendapatkan nilai parameter optimum yaitu *grid search*. Pada penelitian ini juga akan dilakukan optimasi menggunakan *Genetic Algorithm* untuk mendapatkan parameter optimum sehingga didapat nilai akurasi yang optimum juga. Optimasi dengan menggunakan *Genetic Algorithm* memiliki beberapa langkah utama yang harus dilakukan

1. Mendefinisikan variabel-variabel yang diperlukan dalam optimasi GA. Variabel tersebut adalah jumlah populasi, banyak iterasi, peluang pindah silang, peluang mutasi, banyak kromosom untuk *ellitism*, batas atas dan batas bawah masing-masing parameter inputan. Adapun besaran yang ditentukan yaitu N=15, $MNI = 5$, $P_c = 0{,}8$, $P_m = 0{,}1$, $n_e = 2$, $L_{b,C} = 49$, $U_{b,C} = 250$, $L_{b,\sigma^2} = 25$, $U_{b,\sigma^2} = 100$
2. Melakukan inisiasi kromosom. Optimasi dilakukan pada parameter model LSSVR dengan fungsi kernel Gaussian sehingga kromosom yang terbentuk yaitu vektor berukuran 2×1. Vektor tersebut berisi nilai parameter dari LSSVR yakni C dan σ^2. Nilai pertama dari pembentukan kromosom akan diisi dengan *good initial value* yaitu hasil dari pemodelan LSSVR sebelumnya

cost	sigma	→	50	25,75

Gambar 4.27 Bentuk Kromosom

Selanjutnya akan dibangkitkan data sejumlah banyak populasi sehinga terbentuk 15 kromosom dengan nilai parameter sesuai dengan batas yang telah ditentukan seperti yang ditunjukkan pada Tabel 4.10
3. Hitung nilai fitness di tiap kromosom. Nilai fitness dihitung menggunakan $RMSE$. Hasil perhitungan disajikan pada Tabel 4.10

Tabel 4.10 Pembentukan Kromosom

Kromosom	Cost	Sigma	Fitness
1	187.9772	48.75893	43.33881
2	92.60384	28.83976	45.43344
⋮	⋮	⋮	⋮
14	98.13115	74.94433	50.7528
15	98.00245	65.12841	51.9833

4. Selanjutnya tiap kromosom akan diseleksi untuk dijadikan sebagai calon orang tua. Seleksi dilakukan dengan menggunakan metode *roulette wheel*. Tahapan metode *roulette wheel* dilakukan dengan menghitung proporsi nilai *fitness*, menghitung

nilai kumulatif dari *fitness*, membangkitkan bilangan random berdistribusi uniform antara 0 dan 1, kemudian membandingkan nilai kumulatif dengan bilangan random tersebut. Jika nilai kumulatif ke-*i* lebih besar dari bilangan random, maka kromosom ke-*i* tersebut akan terpilih sebagai orang tua untuk generasi berikutnya. Ilutrasi perhitungan metode *roullete wheel* ditunjukkan pada Tabel 4.11. Berdasarkan Tabel 4.11 dapat diketahui bahwa bilangan random pada kromosom ke-2, 6 dan 7 memenuhi kriteria nilai kumulatif *fitness* lebih besar dari bilangan random, sehingga kromosom tersebut terpilih sebagai orangtua.

Tabel 4.11 Proses *Roulette Wheel*

Kromosom	Fitness	Proporsi Nilai *fitness*	Kumulatif *fitness*	Bilangan random
1	43.33881	0.064353	0.064353	0,769
2	45.43344	0.067175	0.135524	0,1432
⋮	⋮	⋮	⋮	⋮
14	50.7528	0.07504	0.923141	0,213
15	51.9833	0.076859	1	0,889

5. Proses berikutnya yaitu melakukan pindah silang yang dimaksudkan untuk menghasilkan kromosom baru dari perpaduan dua kromosom orang tua. Pindah silang dilakukan dengan menggunakan metode *crossover single point* atau *crossover* sederhana. Prosedur yang dilakukan yaitu dengan menukarsilangkan salah satu gen pada kromosom bapak dan ibu. Titik potong untuk penukaran gen dapat dilakukan secara acak, akan tetapi pada penelitian ini tidak perlu menentukan titik potong sebab hanya terdapat dua gen di tiap kromosom. Selanjutnya, akan dibangkitkan bilangan random berdistribusi uniform (0,1), jika bilangan tersebut kurang dari peluang pindah silang $(P_c = 0{,}8)$ maka kromosom tersebut akan terpilih sebagai orangtua yang melakukan proses crossover. Ilustrasi pindah silang disajikan seperti pada Gambar 4.28

Cost 1	Sigma 1	→	Cost 2	Sigma 1
Cost 2	Sigma 2		Cost 1	Sigma 2

Gambar 4.28 Proses *Crossover Singlepoint* Kromosom

Kromosom 1 melakukan penukaran gen dengan kromosom 2, sehingga menghasilkan kromosom baru pada bagian kanan. Selanjutnya kromosom baru tersebut juga dihitung nilai *fitness*.

6. Tahap selanjutnya yaitu mutasi menggunakan mutasi uniform. Proses mutasi dimaksudkan untuk mendapatkan keberagaman gen atau parameter dalam kasus ini. Mutasi dilakukan dengan mengubah salah satu nilai parameter dengan suatu bilangan random. Pemilihan kromosom yang akan melakukan proses mutasi yaitu dengan membangkitkan berdistribusi uniform $(0,1)$ dan membandingkan dengan peluang mutasi $(P_m = 0,1)$, jika bilangan yang dibangkitkan bernilai lebih kecil dari nilai peluang yang ditentukan maka kromosom tersebut merupakan kromosom terpilih untuk melakukan mutasi. Ilustrasi metode mutasi uniform disajikan pada Gambar 4.29

Cost	***Sigma***	***U***		***Cost***	***Sigma***
187,98	48,76	**0,025**		187,98	0,965
92,6	28,839	**0,217**	→	92,6	28,839
92,6	48,76	**0,082**		0,0658	28,839
187,98	28,839	**0,583**		187,98	28,839

Gambar 4.29 Mutasi Uniform Kromosom (kiri sebelum mutasi, kanan sesudah mutasi)

Berdasarkan Gambar 4.29 terlihat bahwa kromosom ke-2 dan ke-3 memiliki bilangan random kurang dari peluang yang ditentukan sehingga nilai salah satu gen kromosom tersebut diganti dengan bilangan berdistribusi uniform $(0,1)$.

7. Proses berikutnya yaitu mengurutkan nilai *fitness* dari semua kromosom yang terbentuk, baik dari 7 kromosom awal, hasil pindah silang, dan mutasi dari paling kecil hingga paling besar. Sebanyak n_e kromosom dengan nilai *fitness* tertinggi dipertahankan dan digunakan sebagai orang tua pada generasi berikutnya. Selanjutnya adalah membangkitkan generasi baru sebanyak N kromosom dan dilakukan iterasi hingga iterasi ketiga/kekonvergenan terpenuhi.

Setelah dilakukan ketujuh tahapan diatas maka akan dihasilkan nilai ramalan yang dapat dibandingkan dengan data *testing*. Adapun performa dari optimasi parameter menggunakan algoritma GA disajikan pada Gambar 4.30

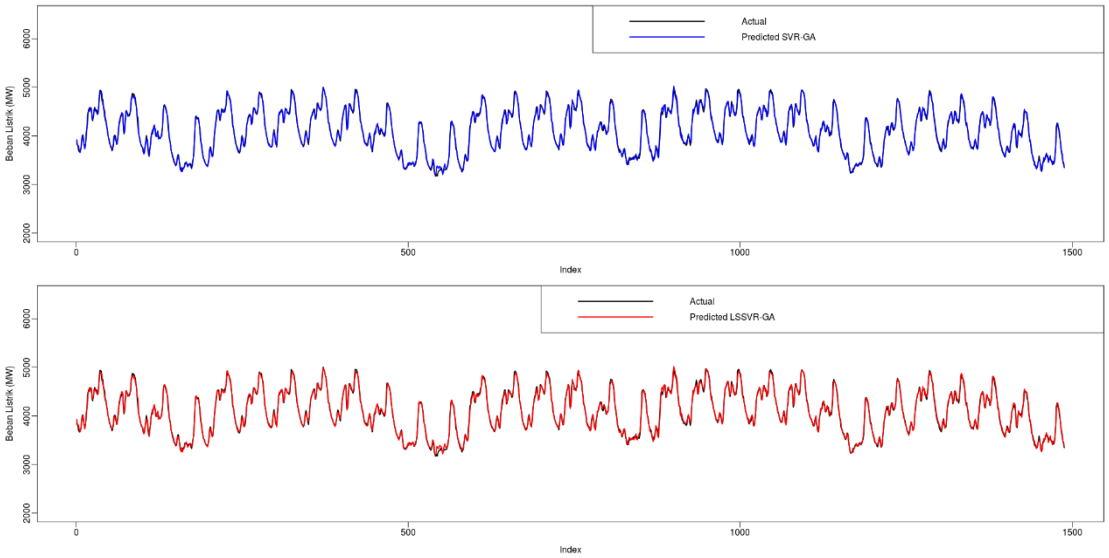
Gambar 4.30 Performansi Algoritma *Genetic Algorithm* LSSVR

Pada Gambar 4.30 menunjukkan konvergensi dari *Genetic Algorithm* dicapai pada iterasi ke-5 dengan nilai rata-rata *fitness* sebesar 43,4545 dan *fitness* terbaik sebesar 43,2587. Berdasarkan nilai *fitness* optimum, didapatkan parameter optimal sebesar 197,2273 dan 43,88546 untuk σ^2 dan C dengan nilai RMSE sebesar 43,25871.

Pada metode SVR, pemilihan parameter optimal dilakukan dengan prosedur yang sama, tetapi berbeda ukuran kromosom. Hal tersebut mengacu pada banyaknya parameter SVR yang dioptimalkan yaitu ε, C, dan σ^2. Range parameter yang digunakan yaitu $\{(1e-4:1e-2),(8:12),(1e-4:1e-2)\}$ untuk ε, C, dan σ^2. Kekonvergenan algoritma *Genetic Algorithm* pada SVR dicapai ketika mencapai tiga iterasi dengan nilai rata-rata *fitness* -43.62 dan *fitness* terbaik sebesar -42,901. Berdasarkan nilai *fitness* optimum, didapatkan parameter optimal sebesar 0,00839; 11,29062; 0,001119 untuk ε, C, dan σ^2, dengan nilai RMSE sebesar 42,901. Adapun perbandingan akurasi pada data *testing* terlihat pada Tabel 4.12

Tabel 4.12 Perbandingan Akurasi Optimasi Genetic Algorithm

SVR-GA		LSSVR-GA	
RMSE	sMAPE	RMSE	sMAPE
42.90561734	0.00783032	43.258705580	0.007993672



Gambar 4.31 Plot *Actual* vs *Predicted* SVR-GA dan LSSVR-GA

Berdasarkan Gambar 4.31 terlihat bahwa perbandingan plot data *testing* dengan hasil prediksi dari model SVR dan LSSVR menunjukkan hasil yang baik karena nilai ramalan yang didapat mendekati nilai asli, begitu juga dengan pola tren naik turun data yang dapat ditangkap oleh hasil estimasi model. Hasil akurasi pada Tabel 4.12 menunjukkan bahwa berdasarkan RMSE minimum didapat metode SVR-GA memiliki

akurasi yang lebih baik dengan nilai sMAPE dan RMSE sebesar 0,783% dan 42,906. Jadi dapat disimpulkan bahwa optimasi Genetic Algorithm pada SVR mempunyai akurasi yang lebih baik dibandingkan dengan metode LSSVR-GA dalam meramalkan data beban listrik jangka pendek area Jawa Timur.

4.6 Optimasi Particle Swarm Optimization

Sama seperti halnya dengan algoritma *Genetic Algorithm*, algoritma *Particle Swarm Optimization* digunakan untuk mempermudah mendapatkan nilai parameter model LSSVR yang optimum. Algoritma PSO mengadaptasi perilaku sekelompok hewan untu menemukan wilayah yang memiliki bahan makanan mencukupi (dalam algoritma disebut *fitness*) berdasarkan insting mereka (dalam algoritma, kemampuan insting digunakan untuk mencari dua daerah yaitu *local best* dan *global best*). Hal tersebut identik dengan eksplorasi yang merupakan kemampuan untuk mendapatkan sebuah hasil yang optimum pada setiap daerah yang berbeda-beda. Berikut merupakan prosedur yang dilakukan pada optimasi PSO

1. Mendefinisikan parameter yang digunakan dalam PSO, yaitu jumlah partikel, batas iterasi maksimum, bobot inersia, batas maksimum kecepatan, koefisien komponen tiap partikel dan koefisien komponen semua partikel. Adapun besaran yang ditentukan yaitu nilai $N = 15$, $MNI = 5$, $w = 0.9$ $k = 0.2$, $, v_{max,C} = k \times U_{b,C}$, $v_{max,\sigma^2} = k \times U_{b,\sigma^2}$, $c_1 = 2$, $c_2 = 2$. Penentuan nilai w, c_1, c_2 sebagaimana disebutkan oleh Bai (2010) bahwa $c_1 = c_2$ biasanya bernilai 2 dan nilai w ditentukan berdasarkan hasil simulasi pada model AR(1)
2. Inisiasi vektor posisi dan vektor kecepatan dari tiap partikel. Dimensi dari partikel ditentukan berdasarkan banyak parameter yang akan dioptimasi. sehingga didapat vektor berukuran 2×1. Inisiasi kecepatan partikel v_{id}^k merupakan bilangan bangkitan yang memenuhi kriteria $-v_{max} < v_{id}^k < v_{max}$. Adapun untuk vektor posisi x_{id}^k berisi nilai parameter dari LSSVR yakni C dan σ^2. Nilai pertama dari pembentukan posisi partikel akan diisi dengan *good initial value* yaitu hasil dari

pemodelan LSSVR sebelumnya. Range parameter yang digunakan secara berturut-turut untuk *cost* dan *sigma* adalah {(49:200),(25:100)}.

3. Hitung nilai *fitness* di tiap partikel. Nilai fitness dihitung menggunakan , $F(x) = -f(x)$ dengan , $f(x) = RMSE$ berdasarkan perhitungan data *testing*. Hasil perhitungan disajikan pada Tabel 4.13

Tabel 4.13 Populasi Awal PSO

Partikel	x_{i1}^{k}	x_{i2}^{k}	Fitness
1	187.9772	48.75893	-43.33881
2	92.60384	28.83976	-45.43344
⋮	⋮	⋮	⋮
14	98.13115	74.94433	-50.7528
15	98.00245	65.12841	-51.9833

4. Hitung nilai *local best/Pbest* dan *global best/Gbest.* Pada iterasi awal, semua inisiasi posisi partikel merupakan *Pbest*, sedangkan *Gbest* merupakan nilai *fitness* terbaik. Pada iterasi pertama sebagaimana Tabel 4.13, didapat nilai *fitness* terbaik sebesar -45.43344 yang juga merupakan nilai *Gbest*.
5. Selanjutnya hitung nilai kecepatan partikel untuk bergerak pada iterasi berikutnya. Pergerakkan partikel didasarkan pada nilai *Pbest* dan *Gbest.* Kecepatan perpindahan dihitung berdasarkan persamaan (2.38), lalu partikel akan bergerak pada posisi baru sesuai dengan perhitungan kecepatan. Posisi pergerakkan partikel ditujukkan pada persamaan (2.37). Jika posisi partikel terbaru untuk x_{i1}^{k+1} tidak termasuk pada $\left[L_{b,C}\ ,\ U_{b,C}\right]$ atau x_{i2}^{k+1} tidak termasuk pada $\left[L_{b,\sigma^2}\ ,U_{b,\sigma^2}\right]$ maka ubah nilai posisi partikel terbaru menjadi nilai posisi partikel sebelumnya, $x_{i1}^{k+1} = x_{i1}^{k}$. Sehingga pada iterasi kedua posisi partikel dapat diilustrasikan seperti Gambar 4.32

$x_{i1}^{(1)}$	$x_{i2}^{(1)}$
187.9772	48.75893

$x_{i1}^{(2)}$	$x_{i2}^{(2)}$
260,479	91,27844

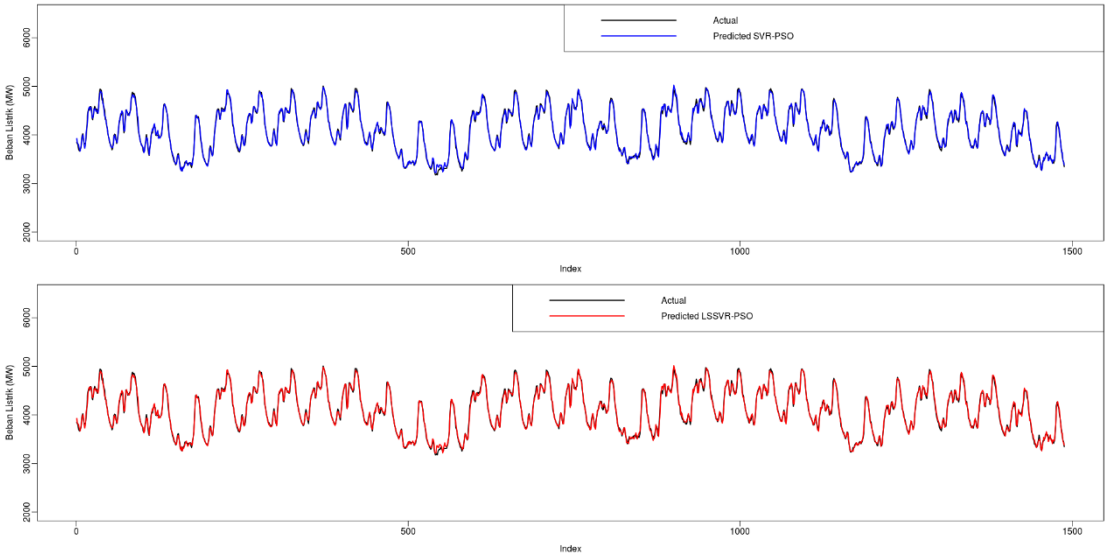
$x_{i1}^{(2)}$	$x_{i2}^{(2)}$
187.9772	91,27844

Gambar 4.32 Pergerakan Partikel pada Iterasi Kedua Algoritma PSO

Setiap partikel akan berpindah sebanyak 5 kali iterasi. Pada setiap perpindahan, tiap partikel menyimpan nilai *fitness* terbaik yang disebut sebagai *Pbest* dan membawa informasi tersebut pada partikel lainnya sehingga nilai *fitness* terbaik untuk keseluruhan partikel dapat ditentukan yang disebut sebagai *Gbest*. Nilai *Pbest* dan *Gbest* akan selalu diperbarui dan menentukan kecepatan untuk mendapatkan posisi terbaru hingga konvergensi tercapai. Setelah kekonvergenan tercapai. akan dilakukan pemodelan dan dihasilkan nilai ramalan yang dapat dibandingkan dengan data *testing*.

Gambar 4.33 Perfomansi Algoritma PSO LSSVR

Hasil ringkasan nilai *fitness* di tiap iterasi ditampilkan pada Gambar 4.33, terlihat bahwa konvergensi dari *Particle Swarm Optimization* dicapai pada iterasi ke-5 dengan nilai rata-rata *fitness* sebesar -43,65 dan *fitness* terbaik sebesar -43,31. Berdasarkan nilai *fitness* optimum, didapatkan parameter optimal sebesar 196,1141 dan 40,597 untuk σ^2 dan C dengan nilai RMSE sebesar 41,87447. Pada metode SVR, pemilihan parameter optimal dilakukan dengan prosedur yang sama, tetapi berbeda ukuran partikel. Hal tersebut mengacu pada banyaknya parameter SVR yang dioptimalkan yaitu ε, C, dan σ^2. Range parameter yang digunakan yaitu $\{(1e-4:1e-2),(8:12),(1e-4:1e-2)\}$ untuk ε, C, dan σ^2. Kekonvergenan algoritma *Particle Swarm Optimization* pada SVR dicapai ketika mencapai tiga iterasi. Berdasarkan nilai *fitness* optimum, didapatkan parameter optimal sebesar 0,005567992; 9,481282; 0,002875503 untuk ε, C, dan σ^2, dengan nilai RMSE sebesar 43,6688. Adapun perbandingan akurasi pada data *testing* terlihat pada Tabel 4.14



Gambar 4.34 Plot *Actual* vs *Predicted* SVR-PSO dan LSSVR-PSO

Tabel 4.14 Perbandingan Akurasi Optimasi Particle Swarm Optimization

SVR-PSO		LSSVR-PSO	
RMSE	sMAPE	RMSE	sMAPE
43,668757	0.008047856	**43.265549290**	0.007994164

Berdasarkan Gambar 4.34 terlihat bahwa perbandingan plot data *testing* dengan hasil prediksi dari model SVR dan LSSVR menunjukkan hasil yang baik karena nilai ramalan yang didapat mendekati nilai asli, begitu juga dengan pola tren naik turun data yang dapat ditangkap oleh hasil estimasi model. Hasil akurasi pada Tabel 4.12 menunjukkan bahwa berdasarkan RMSE minimum didapat metode LSSVR-PSO memiliki akurasi yang lebih baik dengan nilai sMAPE dan RMSE sebesar 0,0079% dan 43,266. Jadi dapat disimpulkan bahwa optimasi Particle Swarm Optimization pada LSSVR mempunyai akurasi yang lebih baik dibandingkan dengan metode LSSVR-GA dalam meramalkan data beban listrik jangka pendek area Jawa Timur.

4.7 Optimasi Modified Particle Swarm Optimization

Selain itu, data juga akan diramalkan menggunakan metode optimasi MPSO untuk mendapatkan nilai parameter yang optimum pada model LSSVR dan diharapkan model yang didapat dapat lebih baik dibandingkan menggunakan metode PSO, sebab algoritma PSO rawan terhadap terjadinya kekonvergenan prematur. Algoritma MPSO menambahkan parameter *discrete degree* $D(t)$ dan σ^2 yang mampu meningkatkan probabilitas penemuan solusi global optimal dan terhindar dari lokal optimal. Secara umum, kedua parameter tersebut mengontrol nilai posisi partikel agar sesuai dengan kriteria yang ditentukan dan akan membangkitkan dengan nilai lain sedemikan hingga kriteria tersebut terpenuhi. Adapun langkah yang dilakukan yaitu sebagai berikut

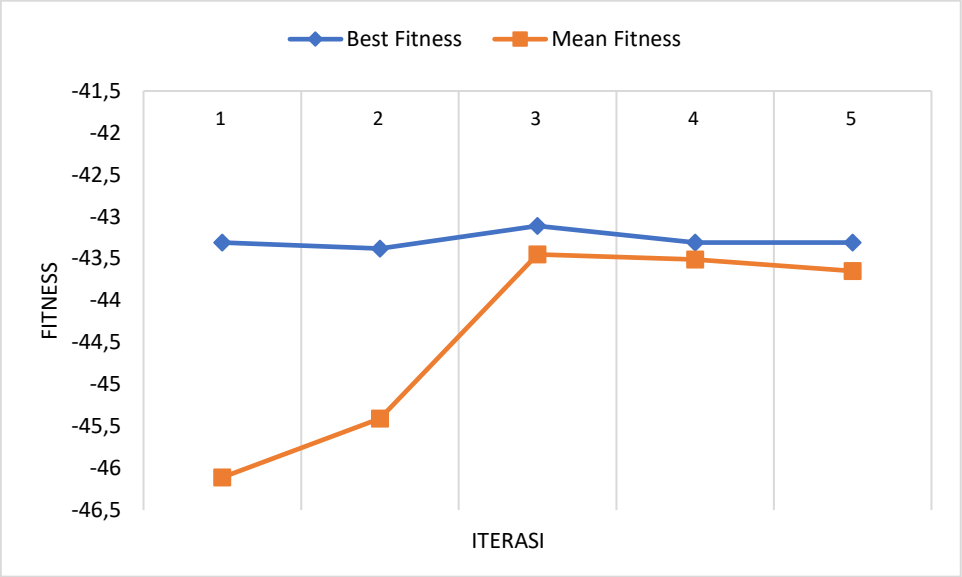
1. Definisikan parameter MPSO seperti definisi parameter PSO dimana nilai parameter $w = 0{,}9$ sesuai dengan hasil dari simulasi PSO, inisiasi vektor posisi dan vektor kecepatan seperti inisiasi keduanya pada PSO, hitung nilai *fitness*, *Pbest*, dan *Gbest* sebagaimana langkah pada algoritma PSO.
2. Hitung nilai $D(t)$ dan σ^2 seperti pada persamaan (2.39) dan (2.40)
3. Menentukan nilai $L = 100,\ \alpha = 0{,}5$, dan $\beta = 0{,}05$
4. Evaluasi nilai $D(t)$ dan σ^2 dengan kriteria jika $D(t) < \alpha$ atau $\sigma^2 < \beta$ maka inisiasi kembali nilai vektor posisi, jika tidak maka lakukan proses PSO untuk memperbarui nilai posisi dan kecepatan partikel.

5. Iterasi akan terus dilakukan dengan mengevaluasi kembali nilai $D(t)$ dan σ^2. Jika $D(t) < \alpha$ atau $\sigma^2 < \beta$ maka bangkitkan data vektor posisi terbaru dan evaluasi kembali hingga didapat nilai $D(t) > \alpha$ dan $\sigma^2 > \beta$
6. Proses perhitungan PSO terkoreksi nilai $D(t)$ dan σ^2 akan terus dilakukan hingga kekonvergenan terpenuhi.

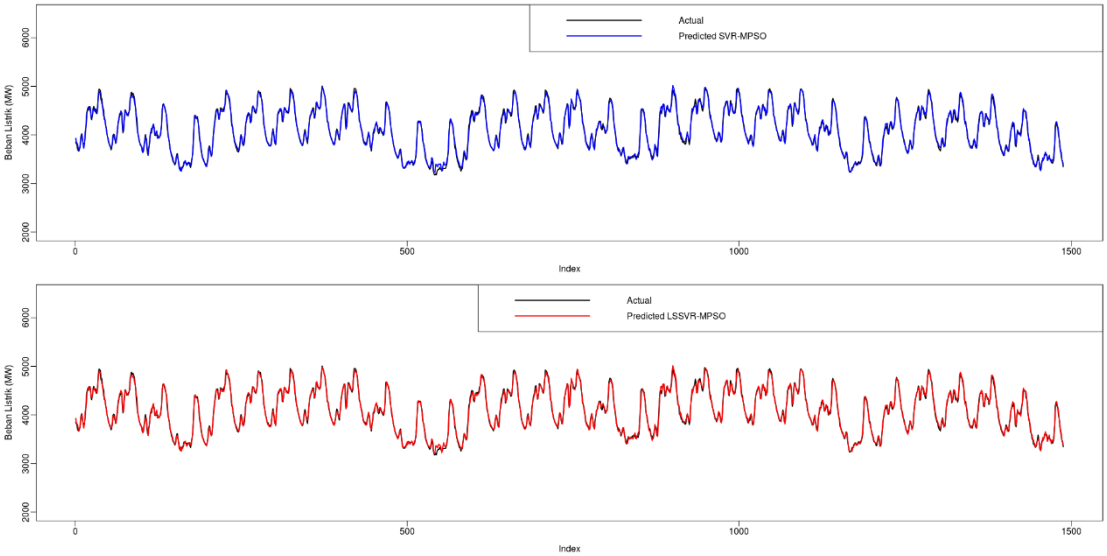
Performa hasil MPSO disajikan pada Gambar 4.35, terlihat bahwa kekonvergenan dari *Modified Particle Swarm Optimization* dicapai pada iterasi ke-3 dengan nilai rata-rata *fitness* sebesar -44,297 dan *fitness* tertinggi sebesar -43,337. Berdasarkan nilai *fitness* maksimum, didapatkan parameter optimal sebesar 188,0168 dan 47,67838 untuk σ^2 dan C dengan nilai RMSE sebesar 43,3354. Pada metode SVR, pemilihan parameter optimal dilakukan dengan prosedur yang sama, tetapi berbeda ukuran partikel. Hal tersebut mengacu pada banyaknya parameter SVR yang dioptimalkan yaitu ε, C, dan σ^2. Range parameter yang digunakan yaitu $\{(1e-4{:}\,1e-2), (8{:}\,12), (1e-4{:}\,1e-2)\}$ untuk ε, C, dan σ^2. Kekonvergenan algoritma *Modified Particle Swarm Optimization* pada SVR dicapai ketika mencapai tiga iterasi. Berdasarkan nilai *fitness* optimum, didapatkan parameter optimal sebesar 0,0001051; 8,3459835649; 0.0044634277 untuk ε, C, dan σ^2, dengan nilai RMSE sebesar 45,403822. Adapun perbandingan akurasi pada data *testing* terlihat pada Tabel 4.14

Tabel 4.15 Perbandingan Akurasi Optimasi Modified Particle Swarm Optimization

SVR-MPSO		LSSVR-MPSO	
RMSE	sMAPE	RMSE	sMAPE
45,40383	0,008459	**43,3354**	0,008014



Gambar 4.35 Performansi Algoritma MPSO LSSVR



Gambar 4.36 Plot *Actual* vs *Predicted* SVR-MPSO dan LSSVR-MPSO

Berdasarkan Gambar 4.34 terlihat bahwa perbandingan plot data *testing* dengan hasil prediksi dari model SVR dan LSSVR menunjukkan hasil yang baik karena nilai ramalan yang didapat mendekati nilai asli, begitu juga dengan pola tren naik turun data yang dapat ditangkap oleh hasil estimasi model. Hasil akurasi pada Tabel 4.12 menunjukkan bahwa berdasarkan RMSE minimum didapat metode LSSVR-MPSO

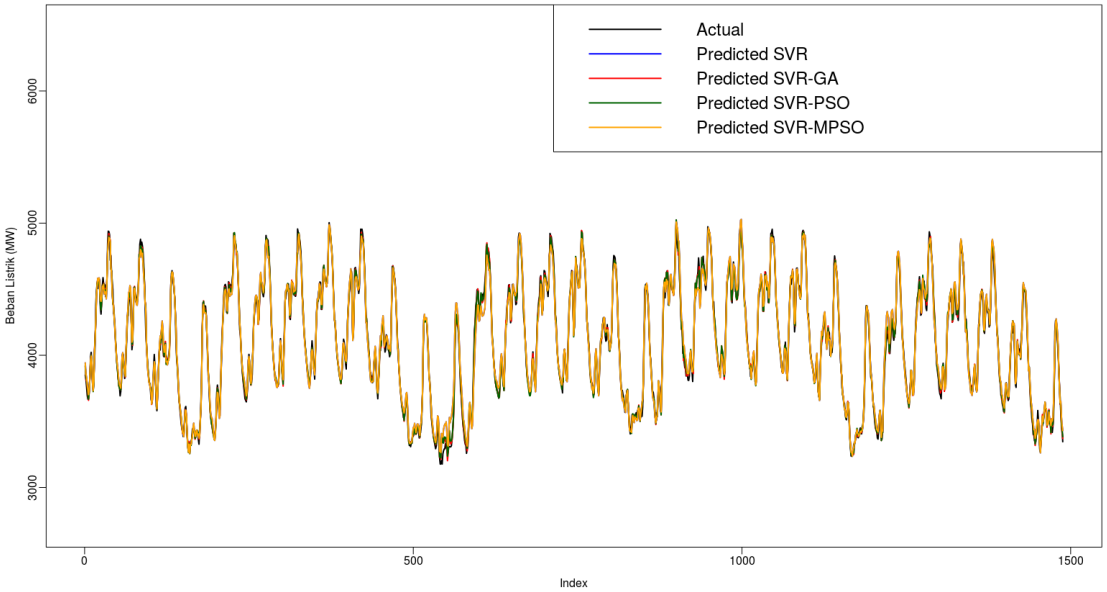
memiliki akurasi yang lebih baik dengan nilai sMAPE dan RMSE sebesar 0,08014% dan 43,3354. Jadi dapat disimpulkan bahwa optimasi Modified Particle Swarm Optimization pada LSSVR mempunyai akurasi yang lebih baik dibandingkan dengan metode LSSVR-GA dalam meramalkan data beban listrik jangka pendek area Jawa Timur.

4.8 Perbandingan Algoritma

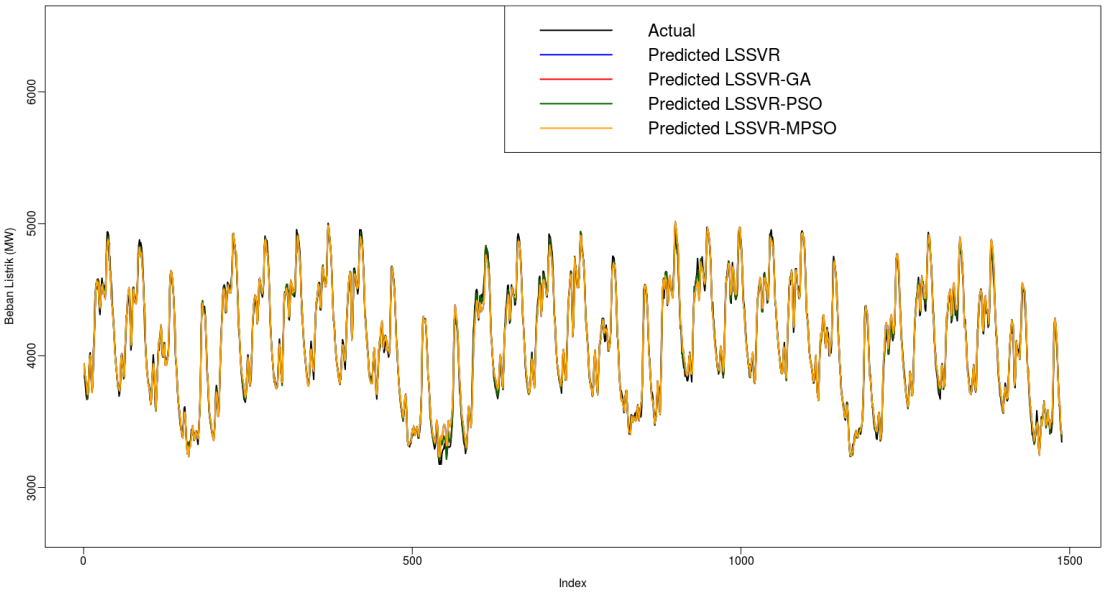
Setelah didapat hasil ramalan dan nilai kebaikan model dari tiap metode, maka langkah selanjutnya yaitu membandingkan nilai kebaikan model antar metode. Perbandingan akan dilakukan untuk mengetahui performansi dari metode optimasi juga untuk mengetahui performansi antara model LSSVR dan SVR. Sebagaimana pada penelitian Yan (2013) dengan adanya kontradiksi pada hasil akurasi yakni nilai akurasi pembentukan model LSSVR memiliki akurasi yang lebih tinggi tetapi akurasi pada hasil ramalan justru lebih rendah dibandingkan SVR, selain itu, pada penelitian Khosravi & Jouybari-Moghaddam (2017) disebutkan bahwa metode LSSVR memiki kemampuan komputasi yang lebih cepat dibandingkan dengan metode non linier lainnya, termasuk salah satunya SVR tetapi belum dijelaskan perbandingan akurasi dari kedua metode tersebut membuat penelitian ini menawarkan adanya penggunaan optimasi parameter dan membandingkannya untuk melihat performansi diantara kedua metode tersebut sekaligus antar metode optimasi diantara *Grid Search*, *Genetic Algorithm*, *Particle Swarm Optimization*, dan *Modified Particle Swarm Optimization*. Perbandingan performa masing-masing metode akan diukur berdasarkan nilai RMSE yang tercantum seperti pada Tabel 4.17 dan visualisasi perbandingan hasil akurasi pada SVR dan LSSVR ditampilkan pada Gambar dan Gambar

Tabel 4.16 Tabel Perbandingan Akurasi dan Lama Komputasi Tiap Metode

Metode	**RMSE**	**Ranking Keseluruhan**	**Waktu (jam)**
DSARIMA	618,1697	IX	0,167
LSSVR	51,114	VII	5
LSSVR-GA	43,2587	I	12
LSSVR-PSO	43,2655	II	16
LSSVR-MPSO	43,3354	IV	30
SVR	58,117	VIII	10
SVR-GA	42,906	II	20
SVR-PSO	43.668	V	24
SVR-MPSO	45,403	VI	45



Gambar 37. Perbandingan Plot Hasil Estimasi pada Model SVR



Gambar 38. Perbandingan Plot Hasil Estimasi pada Model LSSVR

Tabel 4.17 merupakan tabel perbandingan akurasi tiap metode sekaligus terdapat kolom rank yang menjelaskan urutan performansi model dimulai dari nilai RMSE terkecil hingga terbesar untuk membandingkan metode optimasi yang terbaik. Dapat diketahui bahwa pada metode LSSVR metode optimasi terbaik merupakan algoritma GA, kemudian disusul oleh PSO, MPSO, dan *Grid Search*. Sama halnya metode SVR, optimasi pada model SVR mempunyai urutan performansi yang sama yaitu GA, PSO, MPSO lalu *Grid Search*. Berbeda dengan (Niu & Dai, 2017) pada penelitian ini MPSO justru tidak memperbaiki akurasi dari PSO, hal tersebut dimungkinkan sebab adanya kesalahan pada pemilihan parameter pemandu. Selain itu, penelitian ini seperti yang disebutkan oleh Khosravi & Jouybari-Moghaddam (2017) bahwa komputasi LSSVR jauh lebih cepat dibandingkan dengan SVR. Jika dibandingkan dengan DSARIMA, metode berbasis kecerdasan buatan memiliki akurasi yang baik sebab adanya pola kompleks pada series waktu jangka pendek yang tidak dapat diakomodasi oleh model linier.

BAB V

KESIMPULAN DAN SARAN

5.1 Kesimpulan

Kesimpulan yang didapat pada penelitian ini yaitu:

1. Peramalan dengan metode LSSVR dilakukan berdasarkan hasil dari model ARIMA terlebih dahulu. Data beban listrik jangka pendek area Jawa Timur dengan menggunakan model DSARIMA diketahui memiliki sebanyak 111 lag signifikan. Peramalan menggunakan metode LSSVR menghasilkan akurasi hasil ramalan yang jauh lebih baik dibandingkan ARIMA.
2. Secara keseluruhan, secara berturut-turut metode optimasi yang memiliki akurasi terbaik yaitu GA, PSO, dan MPSO. Hal tersebut dikarenakan hasil optimasi GA menjadi peringkat pertama di tiap pemodelan. Pada penelitian ini, algoritma MPSO tidak terbukti meningkatkan akurasi dari PSO hal tersebut dimungkinkan karena kesalahan pemilihan parameter pemandu ataupun kurangnya iterasi. Adapun hasil akurasi pada perbandingan metode SVR dan LSSVR diperoleh kesimpulan metode SVR memiliki akurasi terbaik ketika dikombinasikan dengan metode optimasi, sedangkan pada metode LSSVR penambahan metode optimasi tidak memberi perubahan yang berarti pada hasil akurasi.

5.2 Saran

Berdasarkan hasil analisis yang dilakukan, terdapat beberapa kekurangan sehingga menjadi saran untuk penelitian selanjutnya yaitu sebagai berikut:

1. Perlu dilakukan pemodelan ARIMA yang memenuhi prosedur Box-Jenkins dimana semua asumsi residual terpenuhi sehingga didapatkan model time series berbasis parametrik yang ideal.
2. Perlu dilakukan studi simulasi yang lebih detail agar didapat penentuan parameter tambahan pada metode Modified PSO yang optimal, sebab nilai parameter tersebut bergantung pada data yang digun

3. Keterbatasan perangkat yang dimiliki peneliti, membuat iterasi dari metode optimasi tidak berjalan dengan seharusnya, terlebih data yang diolah begitu banyak dan metode SVR dalam menemukan hyperplane membutuhkan waktu yang begitu lama.
4. Menjadikan indeks data series beban listrik jangka pendek per setengah jam sebagai bukan sesuatu bernilai diskrit untuk menghindari proses running yang begitu lama. Dengan kata lain, jika memiliki kapasitas perangkat yang terbatas, data dengan series waktu yang banyak seharusnya tidak dimodelkan dengan ARIMA. Salah satu metode yang dapat digunakan yaitu Functional Data Analysis.

DAFTAR PUSTAKA

Amin, M., & Hoque, M. (2019). Comparison of ARIMA and SVM for Short-term Load Forecasting. 205-210.

Anonim. (2019). *ESDM: Kebutuhan Listrik Nasional Naik 6,9 Persen Tiap Tahun*. Retrieved from TEMPO.CO: https://www.msn.com/id-id/news/other/esdm-kebutuhan-listrik-nasional-naik-69-persen-tiap-tahun/ar-AAI6q2h

Arifin, J. (2018). *Listrik Padam 9 Jam, Eratex Pulangkan Karyawan*. Retrieved September 9, 2019, from Radar Bromo: https://radarbromo.co.id/2018/09/06/listrik-padam-9-jam-eratex-pulangkan-karyawan/

Arkeman, Y., Seminar, K., & Gunawan, H. (2012). *Algoritma Genetika, Teori dan Aplikasinya untuk Bisnis dan Industri.* (Elviana, & H. Baihaqi, Eds.) Bogor: PT Penerbit IPB Press.

Bai, Q. (2010). Analysis of Particle Swarm Optimization Algorithm. *Computer and Information Science, 3*(1), 180-184.

CNN Indonesia. (2018). *PLTU Paiton Alami Gangguan, Sebagian Jawa dan Bali Mati Lampu*. Retrieved September 9, 2019, from CNN Indonesia: https://www.cnnindonesia.com/nasional/20180905182510-20-327947/pltu-paiton-alami-gangguan-sebagian-jawa-dan-bali-mati-lampu

Cortez, P. (2014). Modern Optimization with R. *Gurmaraes*.

Dreo, J., Petrowski, A., Siarry, P., & Taillard, E. (2006). *Metaheuristics for Hard Optimization* . Berlin: Springer-Verlag Berlin Heidelberg.

Effendy, H. (2018). *Press Gathering PLN Distribusi Jawa Timur*. Retrieved September 10, 2019, from Berita Metro: http://www.beritametro.news/ekonomi/press-gathering-pln-distribusi-jawa-timur

Fahmi, F., & Sofyan, H. (2017). Forecasting Household Electricity Consumption in the Province of Aceh Using Combination Time Series Model. *International Conference on Electrical Engineering and Informatics (ICELTICs)*, 97-102.

Ghalia, M. (2008). Particle Swarm Optimization with An Improved Exploration-Exploitation Balance. *Midwest Symposiu on Circuit and Systems*, 759-762.

Gunn, S. (1998). *Support Vector Machines for Classification and Regression.* University of Southampton. Southampton: Image Speech and Intelligent Systems Group.

Handoko, B. (2010). Peramalan Beban Listrik Jangka Pendek pada Sistem Kelistrikan Jawa Timur dan Bali Menggunakan Fuzzy Time Series.

Hardle, W., Prastyo, D., & Hafner, C. (2014). *Support Vector Machines with Evolutionary Model Selecton for Default Prediction.* (J. Racine, L. Su, & A. Ullah, Eds.) Oxford University Press.

Hassan, R., Cohanim, B., & Weck, O. d. (2005). A Comparison of Particle Swam Optimization and The Genetic Algorithm. 1-13.

Hong, W.-C. (2009). Electric Load Forecasting by Support Vector Model. *Applied Mathematical Modelling, 33*, 2444-2454.

Hu, W., Yan, L., Liu, K., & Wang, H. (2015). PSO-SVR: A Hybrid Short-Term Traffic Flow Forecasting Method. *International Conference on Parallel and Distributed Systems*, 553-561.

Ismail, S., Shabri, A., & Samsudin, R. (2011). A Hybrid Model of Self-Organizing Maps (SOM) and Least Square Support Vector Machine (LSSVM) for Time-Series Forecasting. *Expert Systems with Applications, 38*, 10574-10578.

Ismail, Z., & Irhamah. (2008). Solving the Vehicle Routing Problem with Stochastic Demands via Hybrid Genetic Algorithm-Tabu Search. *Journal of Mathematics and Statistics, 4*(3), 161-167.

Khair, A. (2011). *Peramalan Beban Listrik Jangka Pendek Menggunakan Kombinasi Metode Autoregressive Integrated Moving Average (ARIMA) dengan Regresi Linear Antara Suhu dan Daya Listrik.* Skripsi, Universitas Indonesia, Depok.

Khosravi, I., & Jouybari-Moghaddam, Y. (2017). The Comparison of NN, SVR, LSSVR, and ANFIS at Modelling Meteorological and Remotely Sensed Drought Indices Over the Eastern District of Isfahan, Iran. *Nat Hazards, 87*, 1507-1522.

Khusna, H., & Suhartono. (2015). Pendekatan Percentile Error Bootstrap pada Model Double Seasonal Holt-Winters, Double Seasonal ARIMA, dan Naive untuk Peramalan Beban Listrik Jangka Pendek Area Jawa Timur-Bali. *Jurnal Sains dan Seni ITS, 4*(1), 2337-3520.

Marsudi, D. (2006). *Operasi Sistem Tenaga Listrik.* Yogyakarta: Graha Ilmu.

Min, Z., & Huanqi, T. (2011). Short Term Load Forecasting with Least Square Support Vector Regression and PSO. *Communication in Computer and Information Science, 228*, 124-132.

Niu, D., & Dai, S. (2017). A Short-Term Load Forecasting Model with A Modified Particle Swarm Optimization Algorithm and Least Squares Support Vector Machine Based on the Denoising Method of Empirical Mode Decomposition and Grey Relational Analysis. *Energies, 10*(408), 1-20.

Pangestu, A., Arie, B., Jovi, E., S, R. D., & Prasetyo, H. (2017). Optimizing Neural Network Classifier for Diabetes Data USing Metaheuristic Algorithms. *ITSMART : Jurnal Ilmiah Teknologi dan Informasi, 6*(2), 85-91.

Paramita, N., & Irhamah. (2012). Peramalan Beban Listrik Menggunakan Genetic Algorithm-Support Vector Machine (GA-SVM) di PT PLN (Persero) Sub Unit Penyaluran dan Pusat Pengaturan Beban (P3B) Jawa Timur-Bali. *ITS Library*.

Park, S., Han, S., & Son, Y. (2015). Demand Power Forecasting with Data Mining Method in Smart Grid. *Innovative Smart Grid Technologies - Asia (ISGT-Asia)*.

Pei-you, C., & Lu, L. (2013). Study on Coal Logistics Demand Forecast Based on PSO-SVR. *International Conference on Service System and Service Management*.

PLN. (2018). *Statistik PLN 2017.* Jakarta: Sekretariat Perusahaan PT PLN (Persero).

Prangga, S. (2017). *Optimasi Parameter pada Support Vector Machine Menggunakan Pendekatan Metode Taguchi untuk Data High-Dimensional.* Institut Teknologi Sepuluh Nopember, Statistics, Surabaya.

Purba, A. (2015, Desember 6). Perancangan Aplikasi Peramalan Jumlah Calon Mahasiswa Baru yang Mendaftar Menggunakan Metode Single Exponential

Smooting (Studi Kasus : Fakultas Agama Islam UISU). *Jurnal Riset Komputer, 2*(6), 8-12.

Rachmawati, A. (2019, Januari 1). *PLN Jatim Akan Perpanjang Program Super Power, Proyeksikan Penjualan Listrik 2019 Naik 6,97 Persen*. Retrieved November 4, 2019, from TribunJatim: https://jatim.tribunnews.com/2019/01/01/pln-jatim-akan-perpanjang-program-super-power-proyeksikan-penjualan-listrik-2019-naik-697-persen

Razak, I., Abidin, I., Siah, Y., & Abidin, A. (2017). An Optimization Method of Genetic Algorithm for LSSVM in Medium Term Electricity Price Forecasting. *Journal of Telecommuncation, Electronic and Computer Engineering, 10*(2-5), 99-`03.

Razak, I., Abidin, I., Siah, Y., & Abidin, A. (2017). Short Term Electricity Price Forecasting with Multistage Optimization Technique of LSSVM-GA. *Telecommunication, Electronic, and Computer Engineering, 9*(2-7), 117-122.

Rinanda, H. (2018). *Sejumlah Daerah di Jatim Alami Pemadaman Listrik, Ini Penyebabnya. Retrieved 9 September 2019*. Retrieved September 9, 2019, from Detik News: https://news.detik.com/berita-jawa-timur/d-4292460/sejumlah-daerah-di-jatim-alami-pemadaman-listrik-ini-penyebabnya

Santika, G. (2018). Enabling External Factors for Consumption Electricity Forecasting using Hybrid Genetic Algorithm and Fuzzy Neural System. *International Conference on Computer Applications and Information Processing Technology (CAIPT)*.

Septiningrum, L., Yasin, H., & Sugito. (2015). Prediksii Indeks Harga Saham Gabungan Menggunakan Support Vector Regression (SVR) dengan Algoritma Grid Search. *Jurnal Gaussian, 4*(2), 315-321.

Shabri, A., & Suhartono. (2012). Streamflow Forecasting Using Least-Squares Support Vector Machines. *Hydrological Sciences Journal, 57*(7), 1275-1293.

Singh, S., Hussain, S., & Bazaz, M. (2017). Short Term Load Forecasting Using Artificial Neural Network. *International Conference on Image Information Processing (ICIIP)*, 159-163.

Sivanandam, S., & Deepa, S. (2008). *Introduction to Genetic Algorithm.* Springer, Berlin, Heidelberg.

Smola, A., & Schölkopf, B. (2004). A Tutorial on Support Vector Regression. *Statistics and Computing, 14*, 199-222.

Suykens, J., & Vandewalle, J. (1999). Least Square Support Vector Machine Classifiers. *Neural Processing Letters, 9*, 293-300.

Syafii, & Noveri, E. (2013). Studi Peramalan (Forecasting) Kurva Beban Harian Listrik Jangka Pendek Menggunakan Metode Autoregressive Integrated Moving Average (ARIMA). *Jurnal Nasional Teknik Elektro, 2*, 65-73.

Trapsilasiwi, R., & Sutijo, B. (2011). *Peramalan Beban Listrik Jangka Pendek Menggunakan Hybrid Improved Particle Swarm Optimization-Support Vector Machine di PLN Region Jawa Timur-Bali.* Undergraduate Theses, Institut Teknologi Sepuluh Nopember, Surabaya.

Triboesono, A. (2018). *Statistik Ketenagalistrikan 2017*. Kementerian Energi dan Sumber Daya Mineral, Direktorat Jenderal Ketenagalistrikan, Jakarta.

Trisnanda, C. (2018). *Heboh Pemadaman Listrik di Wilayah Jawa Bali, Berlanjut hingga 6 September 2018?* Retrieved September 9, 2019, from Nakitaid: https://nakita.grid.id/read/02932592/heboh-pemadaman-listrik-di-wilayah-jawa-bali-berlanjut-hingga-6-september-2018?page=all#

Wang, H., & Hu, D. (2006). Comparison of SVM and LS-SVM for Regression. *International Conference on Neural Networks and Brain*, 279-283. doi:10.1109/ICNNB.2005.1614615

Wei, W. W. (2006). *Time Series Analysis Univariate and Multivariate Methods* (2nd ed.). New York: Pearson.

Wijaya, A. (2017). *Optimasi Parameter Model Support Vector Regression untuk Pemodelan Beban Listrik di Empat Belas Wilayah di Jawa Timur dengan Menggunakan Genetic Algorithm dan Particle Swarm Optimization.* Undergraduate Theses, Institut Teknologi Sepuluh Nopember, Surabaya.

Wiyanti, D., & Pulungan, R. (2012). Peramalan Deret Waktu Menggunakan Model Fungsi Basis Radial (RBF) dan Auto Regressive Integrated Moving Average (ARIMA). *Jurnal MIPA, 35*(2), 175-182.

Yan, X. (2013). A Comparison Between SVM and LSSVM in Mid-Term Electricity Market Clearing Price Forecasting. *Canadian Conference Of Electrical And Computer Engineering*.

Ye, T. (2017). Stock Forecasting Method Based on Wavelet Analysis and ARIMA-SVR Model. *International Conference on Information Management*, 102-106.

Zhang, G. (2003). Time Series Forecasting Using a Hybrid ARIMA and Neural Network Model. *Neurocomputing*, 159-175.

Zou, M., & Zhang, H. (2017). Load Forecast Based on Least Square SVR and HHT. *International Conference on Natural Computation, Fuzzy Systems and Knowloedge Discovery (ICNC-FSKD)*, 106-110.

LAMPIRAN

Lampiran 1. Model DSARIMA

$DSARIMA([28,31,47],1,[11,12,31])(0,1,1)^{48}(0,1,2)^{336}$

$$\left(1-\phi_{28}B^{28}-\phi_{31}B^{31}-\phi_{47}B^{47}\right)(1-B)\left(1-B^{48}\right)\left(1-B^{336}\right)\left(1-B^{672}\right)Z_t$$
$$=\left(1-\theta_{11}B^{11}-\theta_{12}B^{12}-\theta_{31}B^{31}\right)\left(1-\Theta B^{48}\right)\left(1-\Theta B^{336}\right)\left(1-\Theta B^{672}\right)a_t$$

$$\begin{aligned}
Z_t = {} & Z_{t-1}+Z_{t-48}-Z_{t-49}+Z_{t-336}-Z_{t-337}-Z_{t-384}+Z_{t-385} \\
& -0,02Z_{t-1}-0,02Z_{t-28}-0,02Z_{t-29}+0,005Z_{t-31}+0,005Z_{t-32}+0,000036Z_{t-47}-0.038Z_{t-48}-0.02Z_{t-49} \\
& +0.0046Z_{t-59}+0.0046Z_{t-60}-0.00036Z_{t-75}+0.0187Z_{t-76}+0.019Z_{t-77}-0.00036Z_{t-78}-0.000715Z_{t-79} \\
& +0.0046Z_{t-80}+0.00036Z_{t-95}+0.00036Z_{t-96}+0.00036Z_{t-106}+0.00072Z_{t-107} \\
& +0.0046Z_{t-108}+0.00036Z_{t-123}+0.00036Z_{t-124}+0.00036Z_{t-126}+0.00036Z_{t-127}+0.00036Z_{t-154} \\
& +0.00036Z_{t-155}-0.019Z_{t-336}-0.019Z_{t-337}-0.019Z_{t-364}-0.019Z_{t-365}+0.0046Z_{t-367}-0.0046Z_{t-368} \\
& +0.00036Z_{t-383}-0.019Z_{t-384}-0.019Z_{t-385}+0.0046Z_{t-395}+0.0046Z_{t-396}+0.000357Z_{t-411}-0.01864Z_{t-412} \\
& +0,02Z_{t-413}+0.00036Z_{t-414}+0.004923Z_{t-415}+0.0046Z_{t-416}+0.00036Z_{t-431}+0.00036Z_{t-432} \\
& +0.00036Z_{t-442}-0.01864Z_{t-443}+0.00036Z_{t-444}+0.00036Z_{t-459}+0.00036Z_{t-460}+0.00036Z_{t-462} \\
& +0.00036Z_{t-463}+0.00036Z_{t-490}+0.00036Z_{t-491}-0.019Z_{t-672}-0.019Z_{t-673}-0.019Z_{t-700} \\
& -0.019Z_{t-701}+0.0046Z_{t-703} \\
& +0,0046Z_{t-704}+0,00036Z_{t-719}-0,019Z_{t-720}-0.010Z_{t-721}+0.0045660Z_{t-731}+0.0045660Z_{t-732} \\
& +0.000357Z_{t-747}-0.01864Z_{t-748}-0.019Z_{t-749}+0.00036Z_{t-750}+0.005Z_{t-751}+0.004Z_{t-752} \\
& +0.000357Z_{t-767}+0.000357Z_{t-768}+0.000357Z_{t-778}+0.005Z_{t-779}+0.000357Z_{t-780} \\
& +0.00036Z_{t-795}+0.00036Z_{t-796}+0.00036Z_{t-798} \\
& +0.000357Z_{t-799}+0.000357Z_{t-826}+0.000357Z_{t-827}-0.019Z_{t-1008}-0.019Z_{t-1009}-0.019Z_{t-1036} \\
& -0.019Z_{t-1036}-0.019Z_{t-1037}+0.0046Z_{t-1039}+0.0046Z_{t-1040}+0.00036Z_{t-1055}-0.019Z_{t-1056} \\
& -0.019Z_{t-1057}+0.0046Z_{t-1067}+0.0046Z_{t-1068}+0.0004Z_{t-1083}-0.019Z_{t-1084}-0.019Z_{t-1085} \\
& +0.0004Z_{t-1086}+0.005Z_{t-1087}+0.005Z_{t-1088}+0.0004Z_{t-1103}+0.0004Z_{t-1104}+0.0004Z_{t-1114} \\
& +0.005Z_{t-1115}+0.005Z_{t-1116}+0.0004Z_{t-1131}+0.0004Z_{t-1132}+0.0004Z_{t-1134}+0.0004Z_{t-1135} \\
& +0.0004Z_{t-1162}+0.0004Z_{t-1163}\left(1-\theta_{11}B^{11}-\theta_{12}B^{12}-\theta_{31}B^{31}\right)\left(1-\Theta B^{48}\right)\left(1-\Theta B^{336}\right)\left(1-\Theta B^{672}\right)a_t
\end{aligned}$$

Lampiran 2. Syntax Pemodelan DSARIMA

```
data listriksas;
input Load;
datalines;
2740.4
2704.4
2636.8
2578.5
2544.3
2519.4
2479.4
.
.
.
4950.3
4756.4
4520.0
4547.1
4421.3
4336.6
;
proc arima data=listriksas;
        identify var=Load(1 48 336) nlag=48 noprint;
        run;
        estimate
        p=(28 31 47)
        q=(11 12 31) (48) (336 672)
        noconstant method=cls;
forecast lead=1488 out=arimaout;
run;

proc univariate data=arimaout normal;
var residual;
run;
```

Lampiran 3. Syntax Training Testing Data R

```
rm(list=ls())
library(pracma)
library(Metrics)
library(e1071)
library(tseries)

dat=read.csv(" ")
month=dat$month
test=dat[month==12,]
ntest=dim(test)[1]
train=dat[1:(dim(dat)[1]-ntest),]
brs=dim(train)[1]
ytr=train$gab

#training testing data
lagpad <- function(x, k) {
  if (k>0) {
    return (c(rep(NA, k), x)[1 : length(x)] );
  }
  else {
    return (c(x[(-k+1) : length(x)], rep(NA, -k)));
  }
}
lag=read.csv(" ")

nlag=length(lag)
maxlag=max(lag)
Xtr=matrix(ncol=length(lag),nrow=dim(train)[1])
for(i in 1:length(lag))
{
  Xtr[,i]=lagpad(ytr,lag[i])
}
Xtr=Xtr[c((maxlag+1):brs),]
ytr=ytr[c((maxlag+1):brs)]
data=data.frame(ytr,Xtr)
index=c(length(ytr)+1,dim(dat)[1])
test=dat[(index[1]-max(lag)):dim(dat)[1],]
Xts=matrix(ncol=nlag,nrow=dim(test)[1])
for(i in 1:nlag)
{
  Xts[,i]=lagpad(test$gab,lag[i])
}
Xts=Xts[(max(lag)+1):dim(test)[1],];yts=test$gab[(max(lag)+1):dim(test)[1]]
```

Lampiran 4. Syntax SVR

```
epsilon=linspace(1e-2,1,2)
cost=linspace(1e-4,10,2)
gamma=linspace(1e-2,50,2)

tune_result=tune(svm,ytr~.,data=data,kernel='radial',
            ranges=list(epsilon=epsilon,cost=cost,gamma=gamma))
tune_model=tune_result$best.model
ypred_in=predict(tune_model,Xtr)
ypred_out=predict(tune_model,Xts)

conf=matrix(0,2,2)
conf[1,1]=rmse(ytr,ypred_in)
conf[1,2]=rmse(yts,ypred_out)
conf[2,1]=smape(ytr,ypred_in)
conf[2,2]=smape(yts,ypred_out)
colnames(conf)=c("insample","outsample")
rownames(conf)=c("rmse","smape");conf
```

Lampiran 5. Syntax SVR-GA

```
fitness_ga=function(param) {
  eps=param[1]
  c=param[2]
  sigma=param[3]
  model=svm(ytr~.,data=data,kernel="radial",type="eps-regression",
          epsilon=eps,gamma=sigma,cost=c)
  ypred=predict(model,Xts)
  smape=smape(yts,ypred)
  smape=1/smape
  return(smape)
}

param_min=c(epsilon=1e-4,cost=8,gamma=1e-4)
param_max=c(epsilon=1e-2,cost=12,gamma=1e-2)
pop=5
iter=2
pcrossover=0.8
pmutation=0.1
elitism=2
result=ga(type="real-
valued",fitness=fitness_ga,names=names(param_min),popSize=pop,maxiter=iter,lower=para
m_min,upper=param_max,

population=gareal_Population,selection=gareal_rwSelection,crossover=gareal_spCrossover,
mutation=gareal_raMutation,
        pcrossover=pcrossover,pmutation=pmutation,elitism=elitism,monitor=plot)
hasil=summary(result)

eps=as.numeric(hasil$solution[1])
cost=as.numeric(hasil$solution[2])
sigma=as.numeric(hasil$solution[3])
mod=svm(ytr~.,data=data,
       epsilon=eps,gamma=sigma,cost=cost)
ypred_in=predict(mod,Xtr)
ypred_out=predict(mod,Xts)

conf=matrix(0,2,2)
conf[1,1]=rmse(ytr,ypred_in)
conf[1,2]=rmse(yts,ypred_out)
conf[2,1]=smape(ytr,ypred_in)
conf[2,2]=smape(yts,ypred_out)
colnames(conf)=c("insample","outsample"); rownames(conf)=c("rmse","smape");conf
```

Lampiran 6. Syntax SVR-PSO

```
fitness_pso=function(param) {
  eps=param[1]
  c=param[2]
  sigma=param[3]
  model=svm(ytr~.,data=data,kernel="radial",type="eps-regression",
          epsilon=eps,gamma=sigma,cost=c)
  ypred=predict(model,Xts)
  smape=smape(yts,ypred)
  smape=1/smape
  return(smape)
}

pso=function(FUN,n=5,max.loop=3,w=0.9,c1=0.2,c2=0.2,
           inisial,xmin,xmax,vmax=c(4,4),seed=10)
{
  d=length(xmin)
  set.seed(seed)
  x=matrix(nrow=n,ncol=d)
  wart.f=c()
  for(i in 1:d){
    x[,i]=runif(n,xmin[i],xmax[i])
  }
  for(j in 1:n){
    wart.f[j]=FUN(x[j,])
  }
  x.best.czastki=x
  x.best.czastki[1,]=inisial
  x.best.roju=matrix(x[which.max(wart.f),],ncol=d)

  v=matrix(runif(n*d,min=-vmax,max=vmax),ncol=d,nrow=n)
  g.mean=c()
  g.best=c()

  for(loop in 1:max.loop) {
    for(j in 1:n){
      wart.f[j]=FUN(x[j,])
    }
    g.mean=rbind(g.mean,mean(wart.f))
    idx=which(wart.f>FUN(x.best.czastki))
    x.best.czastki[idx,]=x[idx,]
    x.best.roju.nowe=matrix(x[which.max(FUN(x.best.czastki)),
    ],ncol=d)

    if(FUN(x.best.roju.nowe)>FUN(x.best.roju)) {
      x.best.roju=x.best.roju.nowe
```

```
    }
    g.best=rbind(g.best,FUN(x.best.roju))

    for(i in 1:n) {
      for(j in 1:d) {
        r1=runif(1)
        r2=runif(1)
        v[i,j]=w*v[i,j]+c1*r1*(x.best.czastki[i,j]-x[i,j])+c2*r2*(x.best.roju[j]-x[i,j])

        if(v[i,j]>vmax[j] || v[i,j]<(-vmax[j])) {
          v[i,j]=vmax[j]
        }
        x_prev=x[i,j]
        x[i,j]=x[i,j]+v[i,j]
        if(x[i,j]>xmax[j]) {
          x[i,j]<-x_prev
        }
        if(x[i,j]<xmin[j]) {
          x[i,j]<-x_prev
        }
      }
    }
    loop<-loop+1
  }
  res<-list(sol=x.best.roju,val=g.best[loop -
1],gbest=g.best,gmean=g.mean,xnew=x.best.czastki,fitness=wart.f)
  return(res)
}

param_min=c(epsilon=1e-4,cost=8,gamma=1e-4)
param_max=c(epsilon=1e-2,cost=12,gamma=1e-2)
inisial=c(1e-2,10,1e-2) #from gridsearch
N=3
max_iter=3
w=0.9
c1=2
c2=2
k=0.2
vmax=k*param_max
xmin=param_min;xmax=param_max
result=pso(FUN=fitness_pso(x),n=N,max.loop=max_iter,w=w,c1=c1,c2=c2,
           inisial=inisial,xmin=param_min,xmax=param_max,vmax=vmax,seed=100)

eps=result$sol[1]
cost=result$sol[2]
sigma=result$sol[3]
gbest=result$gbest
gmean=result$gmean
```

```
xnew=result$xnew
sum=data.frame(gbest,gmean)

mod=svm(ytr~.,data=data,kernel="radial",type="eps-regression",
        epsilon=eps,gamma=sigma,cost=cost)
ypred_in=predict(mod,Xtr)
ypred_out=predict(mod,Xts)

conf=matrix(0,2,2)
conf[1,1]=rmse(ytr,ypred_in)
conf[1,2]=rmse(yts,ypred_out)
conf[2,1]=smape(ytr,ypred_in)
conf[2,2]=smape(yts,ypred_out)
colnames(conf)=c("insample","outsample")
rownames(conf)=c("rmse","smape");conf
```

Lampiran 7. Syntax SVR-MPSO

```
dtsigma=function(x,xmin,xmax,seed=10) {
  d=length(xmin)
  set.seed(seed)
  mean.x=c()
  xd=matrix(ncol=d,nrow=n)
  summ=c()
  for(i in 1:d) {
    mean.x[i]=mean(x[,i])
    for(j in 1:n) {
      xd[j,i]=(x[j,i]-mean.x[i])^2
      summ[j]=sum(xd[j,])
    }
  }
  dt=sum(sqrt(summ))/(d*L)

  wart.f=c()
  for(j in 1:n){
    wart.f[j]=fitness_pso(x[j,])
  }
  ff=abs(wart.f-mean(wart.f))
  for(i in 1:n) {
    if(max(ff) > 1) f=max(ff)
    else f=1
  }
  sigma=sum((ff/f)^2)
  result=list(dt=dt,sigma=sigma)
  return(result)
}

param_min=c(epsilon=1e-4,cost=8,gamma=1e-4)
param_max=c(epsilon=1e-2,cost=12,gamma=1e-2)
inisial=c(1e-2,10,1e-2)
N=3
n=N
max_iter=3
w=0.9
c1=2
c2=2
k=0.2
vmax=k*param_max
xmin=param_min;xmax=param_max
alfa=0.1
beta=1.5
L=10
d=length(xmin)
```

```
set.seed(11)
x=matrix(nrow=n,ncol=d)
for(i in 1:d){
  x[,i]=runif(n,xmin[i],xmax[i])
}
x[1,]=inisial

abc=dtsigma(x,xmin,xmax,seed=10)
dt=abc$dt
sigma=abc$sigma

while(dt<alfa || sigma<beta) {
  for(i in 1:d){
    x[,i]=runif(n,xmin[i],xmax[i])
  }
  abc=dtsigma(x,xmin,xmax,seed=10)
  dt=abc$dt
  sigma=abc$sigma
}

while(dt>alfa || sigma>beta) {
  wart.f=rep(N)
  wart.czastki=rep(N)
  x.best.czastki=matrix(ncol=d,nrow=N)
  x.best.roju=rep(N)
  x.best.roju.nowe=rep(N)
  v=matrix(ncol=d,nrow=N)
  gmean=rep(max_iter)
  gbest=rep(max_iter)
  idx=rep()
  fit.czastki=rep(N)

  for(i in 1:N) {
    for(j in 1:d) {
      v[i,j]=-vmax[j]+runif(1)*(vmax[j]+vmax[j])
      x.best.czastki[i,j]=x[i,j]
    }
    wart.f[i]=fitness_pso(x[i,])
    if(wart.f[i]==max(wart.f)) {
      x.best.roju=abs(x.best.czastki[i,])
    }
  }

  for(loop in 1:max_iter) {
    for(i in 1:N) {
      #wart.f[i]=fitness_pso(x[i,])
      wart.czastki[i]=fitness_pso(x.best.czastki[i,])
```

```
  if(wart.f[i]>wart.czastki[i]) {
    x.best.czastki[i,]=abs(x[i,])
  } else {
    x.best.czastki[i,]=abs(x.best.czastki[i,])
  }
  if(wart.f[i] == max(wart.czastki)) {
    x.best.roju.nowe=abs(x[i,])
  }
}

fit.roju=fitness_pso(x.best.roju)
fit.roju.nowe=fitness_pso(x.best.roju.nowe)
if(fit.roju.nowe>fit.roju) {
  x.best.roju=abs(x.best.roju.nowe)
}
gmean[loop]=mean(wart.f)
gbest[loop]=fitness_pso(x.best.roju)

for(i in 1:N) {
  for(j in 1:d) {
    r1=runif(1)
    r2=runif(1)
    v[i,j]=w*v[i,j]+c1*r1*(x.best.czastki[i,j]-x[i,j])+c2*r2*(x.best.roju[j]-x[i,j])

    if(v[i,j]>vmax[j] || v[i,j]<(-vmax[j])) {
      v[i,j]=vmax[j]
    }
    x_prev=x[i,j]
    x[i,j]=x[i,j]+v[i,j]
    if(x[i,j]>xmax[j]) {
      x[i,j]<-x_prev
    }
    if(x[i,j]<xmin[j]) {
      x[i,j]<-x_prev
    }
    if(x[i,j]<param_min[j] || x[i,j]<0) {
      x[i,j]<-abs(param_min[j])
    }
  }
  wart.f[i]=fitness_pso(x[i,])
}
abc=dtsigma(x,xmin,xmax,seed=10)
dt=abc$dt
sigma=abc$sigma

while(dt<alfa || sigma<beta) {
  for(i in 1:d){
    x[,i]=runif(n,xmin[i],xmax[i])
```

```
      }
      abc=dtsigma(x,xmin,xmax,seed=10)
      dt=abc$dt
      sigma=abc$sigma
    }
  }
}
result<-list(sol=x.best.roju,val=gbest[loop-
1],gbest=gbest,gmean=gmean,xnew=x.best.czastki,fitness=wart.f)

eps=result$sol[1]
cost=result$sol[2]
sigma=result$sol[3]
gbest=result$gbest
gmean=result$gmean
xnew=result$xnew
sum=data.frame(gbest,gmean)

mod=svm(ytr~.,data=data,epsilon=eps,gamma=cost,cost=sigma)
ypred_in=predict(mod,Xtr)
ypred_out=predict(mod,Xts)

conf=matrix(0,2,2)
conf[1,1]=rmse(ytr,ypred_in)
conf[1,2]=rmse(yts,ypred_out)
conf[2,1]=smape(ytr,ypred_in)
conf[2,2]=smape(yts,ypred_out)
colnames(conf)=c("insample","outsample")
rownames(conf)=c("rmse","smape");conf
```

Lampiran 8. Syntax Training Testing Data Matlab

```
function [X,yt]=lagpad(data,lag)
   for i=1:length(lag)
      for j=lag(i)
         p=1;
         for k=1:length(data)
            if p<=j
               X(k,i)=nan;
            else
               X(k,i)=data(k-lag(i));
            end
            p=p+1;
         end
      end
   end
   mlag=max(lag)+1;
   X = X(mlag:length(data),:);
   yt = data(mlag:length(data),:);
end

dat=xlsread(' ');
bln=dat(:,1);
test=dat(bln==12,:);
n_test=size(test,1);
train=dat(1:(length(dat)-n_test),:);
ytr=train(:,2);

%training testing data
lag=xlsread(' ');
[X,yt]=lagpad(ytr,lag);
st=length(train);
index=[st+1 size(dat,1)];
tes_data=dat(index(1)-max(lag):size(dat,1),:);
yts=tes_data(:,2);
[Xtest,ytest]=lagpad(yts,lag);
```

Lampiran 9. Syntax LSSVR

```
function [P]=paramlssvr(sig,gam)
   nsig=length(sig);
   ngam=length(gam);
   k=1;
   for i=1:length(sig)
      for j=1:length(gam)
         P(k,:)=[sig(i) gam(j)];
         k=k+1;
      end
   end
end

sig=linspace(100,125,5); %kernel parameter
gam=linspace(50,100,5); %regularization parameter
P=paramlssvr(sig,gam);
rmse_lssvr=[];
for i=1:length(P)
   type='function estimation';
   [alpha,b]=trainlssvm({X,yt,type,P(i,2),P(i,1),'RBF_kernel'});
   ypred=simlssvm({X,yt,type,P(i,2),P(i,1), ...
      'RBF_kernel','preprocess'},{alpha,b},Xtest);
   rmse(i)=sqrt(mean((ytest-ypred).^2));
end

summary=[rmse' P]
for i=1:length(summary)
   if summary(i,1)==min(rmse)
      sig_opt=summary(i,2);
      gam_opt=summary(i,3);
   end
end

sig_opt=125; %result
gam_opt=50; %result
type='function estimation';
[alpha,b]=trainlssvm({X,yt,type,gam_opt,sig_opt,'RBF_kernel'});
ypred=simlssvm({X,yt,type,gam_opt,sig_opt, ...
   'RBF_kernel','preprocess'},{alpha,b},Xtest);
yfit=simlssvm({X,yt,type,gam_opt,sig_opt, ...
   'RBF_kernel','preprocess'},{alpha,b},X);
```

Lampiran 10. Syntax LSSVR-GA

```
inisial=[50 25.75];
n=length(inisial);
option = gaoptimset('PopulationSize',15,'InitialPopulation',inisial,'PopInitRange',[49 25;200
100]);
option =
gaoptimset(option,'MutationFcn',{@mutationuniform,0.1},'CrossoverFraction',[0.8],'Crossov
erFcn',@crossoversinglepoint);
option = gaoptimset(option,'EliteCount',2,'SelectionFcn',@selectionroulette);
option = gaoptimset(option,'Generations',5,'PlotFcns',@gaplotbestf);
[a,fval,exitflag,output,population,scores]=ga(@funcfitnessss,n,[],[],[],[],[],[],[],option)

sig_opt=a(1);
gam_opt=a(2);
summary=[population scores];

type='function estimation';
[alpha,b]=trainlssvm({X,yt,type,gam_opt,sig_opt,'RBF_kernel'});
ypred=simlssvm({X,yt,type,gam_opt,sig_opt, ...
   'RBF_kernel','preprocess'},{alpha,b},Xtest);
rmse=sqrt(mean((ytest-ypred).^2))
yfit=simlssvm({X,yt,type,gam_opt,sig_opt, ...
   'RBF_kernel','preprocess'},{alpha,b},X);
```

Lampiran 11. Syntax LSSVR-PSO

```
function
[summary1,summary2]=pso(N,maxloop,w,c1,c2,param_min,param_max,init,vmax,x_train,y
_train,x_test,y_test)
   d=length(param_min);
   rng default;
   x=[,];
   wart_f=[];
   wart_czastki=[];
   x_best_czastki=[,];
   x_best_roju=[];
   x_best_roju_nowe=[];
   v=[,];
   gmean=[];
   gbest=[];
   idx=[];
   fit_czastki=[];
   for i=1:N
      for j=1:d
         x(i,j)=param_min(j)+rand()*(param_max(j)-param_min(j));
         v(i,j)=-vmax(j)+rand()*(vmax(j)+vmax(j));
         x_best_czastki(i,j)=x(i,j);
      end
      wart_f(i)=funcfitness_pso(x_train,y_train,x_test,y_test,x(i,:));
      x_best_czastki(1,:)=init;
      if wart_f(i) == max(wart_f)
         x_best_roju=x_best_czastki(i,:);
      end
   end

   for loop=1:maxloop
      for i=1:N
         wart_czastki(i)=funcfitness_pso(x_train,y_train,x_test,y_test,x_best_czastki(i,:));
         if wart_f(i)>wart_czastki(i)
            x_best_czastki(i,:)=abs(x(i,:));
         else
            x_best_czastki(i,:)=abs(x_best_czastki(i,:));
         end
         if wart_czastki(i) == max(wart_czastki)
            x_best_roju_nowe=x(i,:);
         end
      end
      fit_roju=funcfitness_pso(x_train,y_train,x_test,y_test,x_best_roju);
      fit_roju_nowe=funcfitness_pso(x_train,y_train,x_test,y_test,x_best_roju_nowe);
      if fit_roju_nowe > fit_roju
         x_best_roju=abs(x_best_roju_nowe);
```

```
        end
        gmean(loop)=mean(wart_f);
        gbest(loop)=funcfitness_pso(x_train,y_train,x_test,y_test,x_best_roju);

        for i=1:N
            for j=1:d
                r1=rand();
                r2=rand();
                v(i,j)=w.*v(i,j)+c1*r1.*(x_best_czastki(i,j)-x(i,j))+c2*r2.*(x_best_roju(j)-x(i,j));

                if v(i,j)>vmax(j) || v(i,j) < -vmax(j)
                    v(i,j)=vmax(j);
                end
                x_prev=x(i,j);
                x(i,j)=x(i,j)+v(i,j);

                if x(i,j)>param_max(j)
                    x(i,j)=abs(x_prev);
                end

                if x(i,j)<param_min(j)
                    x(i,j)=abs((-x_prev));
                end
            end
            wart_f(i)=funcfitness_pso(x_train,y_train,x_test,y_test,x(i,:));
        end
    end
    summary1=[x_best_czastki wart_czastki']
    summary2=[gbest' gmean']
end

rng default
param_min=[49 25];
param_max=[200 100];
k=0.2;
w=0.9;
vmax=k*param_max;
inisial=[50 25.75];
m=length(param_min);
N=15;
c1=2;
c2=2;
maxloop=5;

[summary,summary_loop]=pso(N,maxloop,w,c1,c2,param_min,param_max,inisial,vmax,X,y
t,Xtest,ytest);
```

```
for i=1:N
   if summary(i,1) == max(summary(:,1))
      sigma=summary(i,2);
      gamma=summary(i,3);
   end
end
type='function estimation'
[alpha,b]=trainlssvm({X,yt,type,gamma,sigma,'RBF_kernel'});
ypred=simlssvm({X,yt,type,gamma,sigma, ...
   'RBF_kernel','preprocess'},{alpha,b},Xtest);
rmse=sqrt(mean((ytest-ypred).^2))
yfit=simlssvm({X,yt,type,gamma,sigma, ...
   'RBF_kernel','preprocess'},{alpha,b},X);
```

Lampiran 12. Syntax LSSVR-MPSO

```
function [dt,sigma]=dtsigma(x,xmin,n,L,x_train,y_train,x_test,y_test)
   d=length(xmin);
   mean_x=[];
   xd=[,];
   summ=[];
   for i=1:d
      mean_x(i)=mean(x(:,i));
      for j=1:n
         xd(j,i)=(x(j,i)-mean_x(i)).*(x(j,i)-mean_x(i));
         summ(j)=sum(xd(j,:));
      end
   end
   dt=sum(sqrt(summ))/(d*L);

   wart.f=[];
   for j=1:n
      wart.f(j)=funcfitness_pso(x_train,y_train,x_test,y_test,x(j,:));
   end
   ff=abs(wart.f-mean(wart.f));
   for i=1:n
      if max(ff)>1
         f=max(ff);
      else
         f=1;
      end
   end
   sigma=sum((ff/f).^2);
end

rng default
param_min=[49 25];
param_max=[200 100];
k=0.2;
w=0.9;
vmax=k*param_max;
inisial=[50 25.75];
m=length(param_min);
N=15;
c1=2;
c2=2;
maxloop=5;

L=100;
alfa=0.5;
beta=0.05;
```

```
xmin=param_min;
xmax=param_max;

d=length(xmin);
for i=1:N
   for j=1:d
      x(i,j)=abs(param_min(j)+rand()*(param_max(j)-param_min(j)));
   end
end
x(1,:)=init;
[dt,sigma]=dtsigma(x,param_min,n,L,X,yt,Xtest,ytest);

while dt<alfa || sigma<beta
   for i=1:N
      for j=1:d
         x(i,j)=param_min(j)+rand()*(param_max(j)-param_min(j));
      end
   end
   x
   [dt,sigma]=dtsigma(x,param_min,n,L,X,yt,Xtest,ytest);
end

x_train=X;
y_train=yt;
x_test=Xtest;
y_test=ytest;

if dt>alfa || sigma>beta
   wart_f=[];
   wart_czastki=[];
   x_best_czastki=[,];
   x_best_roju=[];
   x_best_roju_nowe=[];
   v=[,];
   gmean=[];
   gbest=[];
   idx=[];
   fit_czastki=[];
   for i=1:N
      for j=1:d
         %x(i,j)=param_min(j)+rand()*(param_max(j)-param_min(j));
         v(i,j)=-vmax(j)+rand()*(vmax(j)+vmax(j));
         x_best_czastki(i,j)=x(i,j);
      end
      wart_f(i)=funcfitness_pso(x_train,y_train,x_test,y_test,x(i,:));
      %x_best_czastki(1,:)=init;
      if wart_f(i) == max(wart_f)
         x_best_roju=abs(x_best_czastki(i,:));
```

```
    end
end

for loop=1:maxloop
    loop
    for i=1:N
        %wart_f(i)=funcfitness_pso(x_train,y_train,x_test,y_test,x(i,:));
        wart_czastki(i)=funcfitness_pso(x_train,y_train,x_test,y_test,x_best_czastki(i,:));
        if wart_f(i)>wart_czastki(i)
            x_best_czastki(i,:)=abs(x(i,:));
        else
            x_best_czastki(i,:)=abs(x_best_czastki(i,:));
        end
        %fit_czastki(i)=funcfitness_pso(x_train,y_train,x_test,y_test,x_best_czastki(i,:));
        if wart_czastki(i) == max(wart_czastki)
            x_best_roju_nowe=abs(x(i,:));
        end
    end
    fit_roju=funcfitness_pso(x_train,y_train,x_test,y_test,x_best_roju);
    fit_roju_nowe=funcfitness_pso(x_train,y_train,x_test,y_test,x_best_roju_nowe);
    if fit_roju_nowe > fit_roju
        x_best_roju=abs(x_best_roju_nowe);
    end
    loop
    gmean(loop)=mean(wart_f);
    gbest(loop)=funcfitness_pso(x_train,y_train,x_test,y_test,x_best_roju);

    for i=1:N
        for j=1:d
            r1=rand();
            r2=rand();
            v(i,j)=w.*v(i,j)+c1*r1.*(x_best_czastki(i,j)-x(i,j))+c2*r2.*(x_best_roju(j)-x(i,j));

            if v(i,j)>vmax(j) || v(i,j) < -vmax(j)
                v(i,j)=vmax(j);
            end
            x_prev=abs(x(i,j));
            x(i,j)=x(i,j)+v(i,j);

            if x(i,j)>param_max(j)
                x(i,j)=abs(x_prev);
            end

            if x(i,j)<param_min(j) || x(i,j)<0
                x(i,j)=abs(param_min(j));
            end
        end
        wart_f(i)=funcfitness_pso(x_train,y_train,x_test,y_test,x(i,:));
```

```
        end
        [dt,sigma]=dtsigma(x,param_min,n,L,X,yt,Xtest,ytest);
        loop
        while dt<alfa || sigma<beta
            for i=1:N
                for j=1:d
                    x(i,j)=abs(param_min(j)+rand()*(param_max(j)-param_min(j)));
                end
            end
            x
            [dt,sigma]=dtsigma(x,param_min,n,L,X,yt,Xtest,ytest);
        end
        loop
    end
end
%sol=x_best_roju;
gbest=gbest;
gmean=gmean;
param=x_best_czastki;
fit_param=wart_czastki';
summary=[param fit_param]

for i=1:N
    if summary(i,3) == min(summary(:,3))
        sigma=summary(i,1);
        gamma=summary(i,2);
    end
end
type='function estimation'
[alpha,b]=trainlssvm({X,yt,type,gamma,sigma,'RBF_kernel'});
ypred=simlssvm({X,yt,type,gamma,sigma, ...
    'RBF_kernel','preprocess'},{alpha,b },Xtest);
yfit=simlssvm({X,yt,type,gamma,sigma, ...
    'RBF_kernel','preprocess'},{alpha,b },X);
```

Lampiran 12.

```
dat=xlsread('listrik.csv');
bln=dat(:,1);
test=dat(bln==12,:);
n_test=size(test,1);
train=dat(1:(length(dat)-n_test),:);
ytr=train(:,2);

%training testing data
lag=xlsread('D:\\THE SIS\\_RUNNING NEW\\lagdsarima.csv');
[X,yt]=lagpad(ytr,lag);
st=length(train);
index=[st+1 size(dat,1)];
tes_data=dat(index(1)-max(lag):size(dat,1),:);
yts=tes_data(:,2);
[Xtest,ytest]=lagpad(yts,lag);

%%
%
rng default
param_min=[49 25];
param_max=[200 100];
k=0.2;
w=0.9;
vmax=k*param_max;
inisial=[50 25.75];
init=inisial;
m=length(param_min);
N=15;
c1=2;
c2=2;
maxloop=5;
n=N;

L=100;
alfa=2;
beta=2;
xmin=param_min;
xmax=param_max;

d=length(xmin);
for i=1:N
   for j=1:d
      x(i,j)=abs(param_min(j)+rand()*(param_max(j)-param_min(j)));
   end
end
[dt,sigma]=dtsigma(x,param_min,n,L,X,yt,Xtest,ytest);
```

```
while dt<alfa || sigma<beta
    for i=1:N
        for j=1:d
            x(i,j)=param_min(j)+rand()*(param_max(j)-param_min(j));
        end
    end
    x
    [dt,sigma]=dtsigma(x,param_min,n,L,X,yt,Xtest,ytest);
end

x_train=X;
y_train=yt;
x_test=Xtest;
y_test=ytest;

if dt>alfa && sigma>beta
    wart_f=[];
    wart_czastki=[];
    x_best_czastki=[,];
    x_best_roju=[];
    x_best_roju_nowe=[];
    v=[,];
    gmean=[];
    gbest=[];
    idx=[];
    fit_czastki=[];
    for i=1:N
        for j=1:d
            %x(i,j)=param_min(j)+rand()*(param_max(j)-param_min(j));
            v(i,j)=-vmax(j)+rand()*(vmax(j)+vmax(j));
            x_best_czastki(i,j)=x(i,j);
        end
        wart_f(i)=funcfitness_pso(x_train,y_train,x_test,y_test,x(i,:));
        x_best_czastki(1,:)=init;
        if wart_f(i) == max(wart_f)
            x_best_roju=abs(x_best_czastki(i,:));
        end
    end

    for loop=1:maxloop
        loop
        for i=1:N
            %wart_f(i)=funcfitness_pso(x_train,y_train,x_test,y_test,x(i,:));
            wart_czastki(i)=funcfitness_pso(x_train,y_train,x_test,y_test,x_best_czastki(i,:));
            if wart_f(i)>wart_czastki(i)
                x_best_czastki(i,:)=abs(x(i,:));
            else
                x_best_czastki(i,:)=abs(x_best_czastki(i,:));
```

```
    end
    %fit_czastki(i)=funcfitness_pso(x_train,y_train,x_test,y_test,x_best_czastki(i,:));
    if wart_czastki(i) == max(wart_czastki)
        x_best_roju_nowe=abs(x(i,:));
    end
end
fit_roju=funcfitness_pso(x_train,y_train,x_test,y_test,x_best_roju);
fit_roju_nowe=funcfitness_pso(x_train,y_train,x_test,y_test,x_best_roju_nowe);
if fit_roju_nowe > fit_roju
    x_best_roju=abs(x_best_roju_nowe);
end
loop
gmean(loop)=mean(wart_f);
gbest(loop)=funcfitness_pso(x_train,y_train,x_test,y_test,x_best_roju);

for i=1:N
    for j=1:d
        r1=rand();
        r2=rand();
        v(i,j)=w.*v(i,j)+c1*r1.*(x_best_czastki(i,j)-x(i,j))+c2*r2.*(x_best_roju(j)-x(i,j));

        if v(i,j)>vmax(j) || v(i,j) < -vmax(j)
            v(i,j)=vmax(j);
        end
        x_prev=abs(x(i,j));
        x(i,j)=x(i,j)+v(i,j);

        if x(i,j)>param_max(j)
            x(i,j)=abs(x_prev);
        end

        if x(i,j)<param_min(j) || x(i,j)<0
            x(i,j)=abs(param_min(j));
        end
    end
    wart_f(i)=funcfitness_pso(x_train,y_train,x_test,y_test,x(i,:));
end
[dt,sigma]=dtsigma(x,param_min,n,L,X,yt,Xtest,ytest);
[dt sigma]

loop
while dt<alfa || sigma<beta
    for i=1:N
        for j=1:d
            x(i,j)=abs(param_min(j)+rand()*(param_max(j)-param_min(j)));
        end
    end
    [dt,sigma]=dtsigma(x,param_min,n,L,X,yt,Xtest,ytest);
```

```
            x
            [loop dt sigma]
        end
        loop
    end
end
%sol=x_best_roju;
gbest=gbest;
gmean=gmean;
param=x_best_czastki;
fit_param=wart_czastki';
summary=[param fit_param]

for i=1:N
    if summary(i,3) == max(summary(:,3))
        sigma=summary(i,1);
        gamma=summary(i,2);
    end
end
%sigma=232.715067011541;
%gamma=240.606464799906
type='function estimation'
[alpha_lssvm,b_lssvm]=trainlssvm({X,yt,type,gamma,sigma,'RBF_kernel'});
ypred=simlssvm({X,yt,type,gamma,sigma, ...
    'RBF_kernel','preprocess'},{alpha_lssvm,b_lssvm},Xtest);
yfit=simlssvm({X,yt,type,gamma,sigma, ...
    'RBF_kernel','preprocess'},{alpha_lssvm,b_lssvm},X);

mse=mean((y_test-ypred).^2);
rmse=sqrt(mse)
```

BIODATA PENULIS

Penulis dengan nama lengkap Kinanti Hanugera Gusti, lahir di Surabaya, 3 September 1996. Anak kedua dari Bambang Wuryono Kartika dan Widiyani Setyaninigsih. Selama hidupnya, penulis pernah menempuh pendidikan di SDN Sambikerep II pada tahun 2002-2008, SMPN 26 Surabaya pada tahun 2008-2011 dan SMAN 21 Surabaya pada tahun 2011-2014. Setelah lulus SMA, penulis melanjutkan studi Sarjana di Program Studi Statistika Universitas Airlangga pada tahun 2014 dan lulus pada tahun 2018. Selanjutnya, penulis melanjutkan studi Magister di Jurusan Statistika ITS pada tahun 2018. Untuk informasi ataupun saran dari penelitian ini, pembaca dapat menghubungi penulis di kinantihanugera@gmail.com .