PERAMALAN KONSUMSI AIR DI PDAM KOTA MALANG MENGGUNAKAN METODE *SUPPORT VECTOR REGRESSION* DENGAN *IMPROVE-PARTICLE SWARM OPTIMIZATION*

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:
Mega Manfaati
NIM: 125150201111044

PROGRAM STUDI TEKNIK INFORMATIKA
JURUSAN TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2016

PENGESAHAN

PERAMALAN KONSUMSI AIR DI PDAM KOTA MALANG MENGGUNAKAN METODE *SUPPORT VECTOR REGRESSION* DENGAN *IMPROVE-PARTICLE SWARM OPTIMIZATION*

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh :
Mega Manfaati
NIM: 125150201111044

Skripsi ini telah diuji dan dinyatakan lulus pada
10 Agustus 2016
Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I	Dosen Pembimbing II
Imam Cholissodin, S.Si., M.Kom	Randy Cahya Wihandika, S.ST., M.Kom
NIK: 201201 850719 1 001	NIK: 201405 880206 1001

Mengetahui
Ketua Jurusan Informatika

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

PERNYATAAN ORISINALITAS

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

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

Malang, 17 Agustus 2016

Mega Manfaati

NIM: 125150201111044

KATA PENGANTAR

Puji dan syukur penulis panjatkan kepada Tuhan Yang Maha Esa atas berkat, rahmat, ridho dan karunia-Nya sehingga penulis dapat menyelesaikan tugas akhir yang berjudul "**Peramalan Konsumsi Air Di PDAM Kota Malang Menggunakan Metode *Support Vector Regression* Dengan *Improve-Particle Swarm Optimization***" sebagai salah satu persyaratan untuk menyelesaikan studi di Jurusan Informatika, Fakultas Ilmu Komputer Universitas Brawijaya.

Penulis menyadari bahwa tugas akhir ini dapat terselesaikan berkat bantuan, petunjuk, bimbingan dan dukungan dari berbagai pihak yang telah banyak membantu proses penyelesaian tugas akhir ini. Oleh karena itu penulis ingin menyampaikan terima kasih yang sebesar-besarnya kepada:

1. Bapak Imam Cholissodin, S.Si, M.Kom selaku pembimbing I dan Bapak Randy Cahya Wihandika, S.ST, M.Kom selaku pembimbing II yang telah banyak membimbing dan mengarahkan penulis sehingga dapat menyelesaikan skripsi ini.
2. Bapak Budi Darma Setiawan, S.Kom, M.Cs selaku penguji I dan Bapak M.Ali Fauzi, S.T, M.Kom selaku penguji II yang telah menguji dan memberikan saran untuk skripsi saya sehingga skripsi saya bisa terselesaikan dengan lebih baik.
3. Orang tua penulis, Bapak Koesmani dan Ibu Sujiyah yang tiada henti memberi dukungan baik moril dan materil.
4. Saudara penulis : Hudi Kun Rifai, Ruri Dwi Kustanti, Dian Viky Martiyani, Agatha Gofar Arisandhi yang telah memberikan motivasi dan semangat demi terselesainya skripsi ini.
5. Imam Achmad Hambali yang selalu memberikan dukungan, semangat, bantuan dan selalu menemani penulis selama masa studi dan penyusunan skripsi hingga skripsi ini selesai.
6. Yunastria Christine Irwanti yang sudah banyak memberi bantuan, masukan dan saran dalam penyusunan skripsi ini.
7. Seluruh dosen Fakultas Ilmu Komputer Universitas Brawijaya atas kesediaan membagi ilmunya kepada penulis.
8. Sahabat penulis: Yeny Lisa, Renyta, Anisa yang selalu memberi dukungan dan semangat untuk menyelesaikan skripsi.
9. Sahabat penulis : Akmilatul, Maria, Ainun, Tanti, Nindyari, Tya, Mahda, Taslis, Zahrin, Nana, Adila, yang selalu menemani, bertukar ilmu, semangat dan motivasi kepada penulis.
10. Teman-teman Informatika F terimakasih atas kebersamaan serta bantuan yang telah diberikan dari semester 1 hingga semester 8.

11. Teman-teman seperjuangan Fakultas Ilmu Komputer angkatan 2012 yang telah memberikan bantuan selama masa penyelesaian skripsi.
12. Dan semua pihak yang tidak bisa disebutkan satu per satu. Terima kasih atas segala bantuannya.

Penulis sadar bahwa skripsi ini masih banyak kekurangan, oleh karena itu kritik dan saran yang membangun dangat diharapkan untuk menyempurnakan skripsi ini. Penulis berharap skripsi ini dapat bermanfaat khususnya bagi diri sendiri dan bagi semua pihak.

Malang, 17 Agustus 2016

megamanfaati21@gmail.com

Abstrak

Air merupakan kebutuhan fundamental bagi kelangsungan hidup manusia. Semua makhluk hidup membutuhkan air di dalam melakukan aktifitas dan kegiatan sehari-hari. Manusia sangat bergantung dengan adanya sumber daya air. Badan penyedia dan produksi air Perusahaan Daerah Air Minum (PDAM) sudah didirikan di setiap Kota di Indonesia guna melayani kebutuhan air. Salah satunya adalah di Kota Malang. Banyak faktor yang mengakibatkan bertambahnya jumlah penduduk di Kota Malang yang otomatis akan mempengaruhi peningkatan kebutuhan konsumsi air penduduk Kota Malang. Banyak persoalan yang ada dan salah satunya adalah ketika jumlah air yang didistribusikan PDAM tidak sesuai dengan konsumsi penggunaan air yang mengakibatkan pemborosan air. Sebaliknya ketika yang didistribusikan tidak memenuhi konsumsi penduduk maka akan terjadi kurangnya air dari sisi konsumen tersebut. Oleh karena itu dengan melakukan peramalan konsumsi air ini diharapkan terjadi keseimbangan antara persediaan volume produksi untuk distribusi air dengan kebutuhan air konsumen. Cukup banyak metode peramalan yang sudah digunakan dengan sifat non linier dan menghasilkan peramalan yang cukup baik. Seperti penelitian "*Forecasting Agriculture Water Consumption Bases On PSO and SVM*" yang menghasilkan nilai MAPE sebesar 0.72. Kemudian untuk metode IPSO sendiri pernah digabungkan dengan metode SVR untuk meramalkan kesalahan pada proses *Tennessee Optimization* dan terbukti parameter IPSO menghasilkan tingkat error yang baik. Setelah melakukan pengujian dengan menggunakan data konsumsi air PDAM Kota Malang dengan rentang per bulan dari tahun 2008 sampai 2014, metode SVR yang dioptimasi menggunakan IPSO menghasilkan *Fitness* dengan nilai sebesar 0.719.

Kata Kunci : Air, Konsumsi Air,PDAM, SVR, IPSO, MAPE.

Abstract

Water is a fundamental requirement for humam survival. All human need water in the activity and daily activities. Humans are highly dependent on the existence of water resources. Agency providers and water production Regional Water Company (PDAM) has been set up in every town in Indonesia to serve the needs of the water. One is in the city of Malang. Many factors led to the increase of population in the city of Malang which will automatically affect the increase in water consumption needs of resident the city of Malang. Many problems thet exist, and one of them is when the amount of water distributed taps are not in accordance with the consumption of water use that result in waste water. Otherwise when distributed does not meet the consumption of the population, there will be a lack of water on the consumer. Therefore, by forecasting water consumption is expected to occur the balance between supply production volume for the distribution of water to the water needs of consumer. A lot of forecasting methods that have been used with non-linear and generate a good forecasting. As the study "Forecasting Agriculture Water Consumption Bases On PSO and SVM" which produces MAPE value of 0,72. Then for IPSO method have been combined with SVR methods to forecast errors in the process and proven parameters Tennessee Optimization IPSO produce good error rate. After a test using water consumption with data PDAM Malang per month range from 2008 to 2014, SVR optimized mehod using IPSO generate fitness with a value of 0,719.

Keywords : *Water, Water Consumption,PDAM,SVR,IPSO,MAPE*

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB 1 PENDAHULUAN

Pada bab 1 ini merupakan penjelasan mengenai latar belakang permasalahan untuk penelitian, rumusan masalah dalam penelitian, batasan masalah dalam penelitian, tujuan penelitian, manfaat penelitian, serta sistematika penulisan.

1.1 Latar Belakang

Air merupakan kebutuhan makhluk hidup di dunia yang tidak dapat dipisahkan. Air merupakan bagian terpenting. Air sangat berperan utama dalam kehidupan manusia. Kehidupan makhluk hidup dapat berlangsung juga karena adanya ketersediaan air. Untuk kehidupan sehari hari makhluk hidup menggunakan sumber daya air untuk berbagai aktifitas dan keperluan mulai dari makan, minum, mandi, cuci, bekerja dan berbagai kebutuhan lainnya. Hampir semua aktifitas makhluk hidup khususnya manusia sangat bergantung dengan tersedianya air bersih.

Badan yang menangani produksi air bersih di Indonesia adalah Perusahaan Daerah Air Minum (PDAM). PDAM didirikan di setiap Kota di Indonesia sebagai badan yang melayani kebutuhan air bersih untuk masyarakat. Salah satunya yang ada di Kota Malang. Kota Malang merupakan Kota yang berada di daratan tinggi dan dikelilingi oleh pegunungan. Kota Malang memiliki iklim yang sejuk. Selain itu Kota Malang memiliki kualitas dan fasilitas pendidikan yang memadai. Faktor-faktor inilah yang mengakibatkan meningkatnya jumlah penduduk di Kota Malang, karena banyaknya penduduk yang transmigrasi. Seiring dengan meningkatnya jumlah penduduk di Kota Malang maka otomatis kebutuhan konsumsi air penduduk di Kota Malang akan meningkat.

Berdasarkan uraian singkat di atas PDAM harus meningkatkan kebutuhan air setiap tahunnya. Karena dari beberapa persoalan yang ada salah satunya adalah apabila jumlah air yang didistribusikan PDAM dan disalurkan lebih besar daripada permintaan akan konsumsi penggunaan air maka hal ini akan menunjukkan adanya pemborosan air atau inefisiensi produksi. Sedangkan apabila jumlah air yang didistribusikan PDAM lebih sedikit atau bahkan tidak memenuhi kebutuhan konsumsi air, maka pada sisi konsumen akan terjadi kekurangan air atau rendahnya produktifitas. Oleh karena itu dibutuhkan suatu estimasi untuk memperkirakan dengan tepat seberapa besar volume distribusi air PDAM di Kota Malang yang diperlukan untuk melayani kebutuhan air para konsumen PDAM Kota Malang di tahun tahun berikutnya. Maka penulis mengusulkan sistem peramalan tingkat konsumsi air di PDAM Kota Malang agar terjadi keseimbangan antara persediaan volume produksi untuk distribusi air dengan kebutuhan air pada konsumen.

Terdapat beberapa penelitian mengenai peramalan dan prediksi konsumsi air. Salah satu diantaranya yaitu penelitian yang berjudul "*Forecasting Agriculture Water Consumption Based on PSO and SVM*". Penelitian tersebut bertujuan untuk meramalkan konsumsi air pertanian dengan menggunakan metode *Particle*

Swarm Optimization dan *Support Vector Machine.* Permasalahan tersebut akan ditentukan oleh konstansitas data yang diolah dan bagaimana hasil aliran data tersebut. Dalam penelitian ini PSO digunakan untuk mencari nilai optimal parameter SVM, dimana SVM merupakan metode yang digunakan untuk peramalan konsumsi air pertanian tersebut. Adapun parameter SVM yang dioptimasi diantaranya yaitu nilai C, ε dan σ. Hasil nilai MAPE dari peramalan menggunakan metode PSO-SVM ini lebih akurat dibandingkan dengan metode BPNN yaitu 0,72% : 3,34%. Dengan kesimpulan apabila nilai *forecasting* error suatu data semakin kecil, maka data peramalan tersebut dapat dinyatakan semakin akurat (Zhang et al., 2014).

Berdasarkan beberapa penelitian yang telah disebutkan di atas, maka pada penelitian ini penulis mengusulkan peramalan konsumsi air dengan menggunakan metode *Support Vector Regression* dengan metode optimasi *Improve Particle Swarm Optimization*. Penelitian ini bermaksud untuk menggabungkan metode peramalan untuk melakukan peramalan pada konsumsi air, dan metode IPSO yang digunakan sebagai metode untuk mencari koefisien atau parameter yang optimal terhadap SVR. Diharapkan dengan mengimplementasi kedua metode ini dapat memeberikan hasil peramalan yang bagus untuk konsumsi air di PDAM Kota Malang. Yang mungkin nantinya dapat digunakan sebagai bahan masukan atau pertimbangan bagi pihak PDAM dalam menentukan air yang harus di distribusikan ke seluruh kecamatan yang ada di wilayah Kota Malang. Dengan begitu, sistem pendistribusian akan menjadi lebih efektif dan efisien. Serta tingkat penggunaan air yang berlebihan oleh penduduk dapat menurun karena adanya keseimbangan antara peramalan konsumsi air dan jumlah penduduk.

1.2 Rumusan Masalah

Dengan latar belakang di atas, maka dirumuskan permasalahan sebagai berikut :

1. Bagaimana mengimplementasikan algoritma *Improve Particle Swarm Optimization* (IPSO) untuk optimasi peramalan konsumsi air menggunakan *Support Vector Regression* (SVR) ?
2. Bagaimana tingkat error rate peramalan konsumsi air di PDAM kota Malang menggunakan metode *Support Vector Regression* (SVR) dengan *Improve Particle Swarm Optimization* (IPSO)

1.3 Tujuan

Tujuan dalam penelitian ini adalah :

1. Mengimplementasikan algoritma *Improve Particle Swarm Optimization* (IPSO) untuk optimasi peramalan konsumsi air menggunakan metode *Support Vector Regression* (SVR)
2. Mengukur tingkat akurasi peramalan konsumsi air di PDAM Malang menggunakan metode *Support Vector Regression* (SVR) dengan *Improve Particle Swarm Optimization* (IPSO)

1.4 Manfaat

Penelitian ini diharapkan dapat bermanfaat untuk berbagai pihak. Manfaat dari penelitian ini adalah sebagai berikut:

- Bagi Penulis

  Penulis mendapatkan pemahaman dan pengetahuan tentang implementasi algoritma *Improve Particle Swarm Optimization* (IPSO) untuk optimasi peramalan konsumsi air menggunakan metode *Support Vector Regression* (SVR).

- Bagi Masyarakat

  1. Diharapkan tidak adanya keluhan atas kurangnya kesediaan air yang di di distribusikan.
  2. Tidak terjadi peledakan air atau pemborosan oleh masyarakat.

- Bagi PDAM

  1. Memberikan pertimbangan dalam penyediaan volume produksi air untuk pendistribusian sesuai dengan tingkat peramalan konsumsi air yang dibutuhkan.
  2. Memberikan kemudahan dalam membagi volume distribusi air sesuai dengan tingkat konsumsi air yang dibutuhkan masyarakat Kota Malang.

1.5 Batasan Masalah

Agar pembahasan penelitian ini tidak menyimpang dari apa yang telah dirumuskan, maka diperlukan batasan-batasan. Batasan-batasan dalam penelitian ini adalah:

1. Metode yang digunakan adalah *Support Vector Regression* (SVR) yang dioptimasikan dengan metode *Improve Particle Swarm Optimization* (IPSO).
2. Obyek yang digunakan adalah data pemakaian konsumsi air di PDAM Kota Malang.
3. Data yang dibuat acuan adalah pemakaian konsumsi air di PDAM Kota Malang 7 tahun terakhir, yaitu dari tahun 2008 – 2014 yang dibuat dalam bulanan. Data tersebut didapat dari pihak kantor PDAM Kota Malang.
4. Keluaran yang nantinya akan dihasilkan sistem adalah *value forecasting error*, dengan asumsi semakin kecil *value forecasting error* maka hasil peramalan data tersebut semakin baik.

1.6 Sistematika Pembahasan

Sistematika pembahasan pada penelitian ini tersusun sebagai berikut:

BAB I : Pendahuluan

Bab ini menjelaskan mengenai latar belakang permasalahan, rumusan masalah dalam penelitian, batasan masalah dalam penelitian, manfaat, tujuan, dan sistematika penulisan.

BAB II : Landasan Kepustakaan

Bagian ini menjelaskan mengenai kajian pustaka, dasar teori yang menjadi bahan acuan di dalam penelitian maupun penulisan skripsi yang diperoleh dari beberapa literatur, jurnal, dan halaman website.

BAB III : Metodologi

Pada Bab ini akan berisi metode yang digunakan dan langkah-langkah penelitian yang dimulai dengan tahapan-tahapan penelitian, kebutuhan dalam sistem,formulasi permasalahan, tahapan penyelesaian penelitian serta perhitungan yang masih dilakukan secara manual.

BAB IV : Perancangan

Bab ini berisi perancangan database, perancangan antarmuka, serta perancangan hasil pengujian dan evaluasi.

BAB V : Implementasi Sistem

Bab ini berisi penjelasan tentang teknis implementasi dari sistem, batasan-batasan sistem, serta algoritma untuk mengembangkan sistem.

BAB VI : Pengujian dan Analisis

Bab ini menjelaskan tentang tingkat akurasi dan analisa hasil terhadap metode yang digunakan

BAB VII : Penutup

Bab ini berisi kesimpulan dan saran atas kekurangan di dalamnya sehingga dapat dikembangkan menjadi penelitian yang lebih baik.

BAB 2 LANDASAN KEPUSTAKAAN

Secara umum pada bab ini berisi kajian pustaka dan dasar teori yang digunakan di dalam penelitian. Dasar teori ini terfokus pada data konsumsi air sebagai objek yang digunakan di dalam penelitian, *Improve Particle Swarm Optimization* sebagai metode optimasi dan *Support Vector Regression* sebagai algoritma peramalan yang akan digunakan di dalam penelitian.

2.1 Kajian Pustaka

Peramalan dan prediksi mengenai konsumsi air telah banyak dibahas pada penelitian sebelumnya. Salah satu diantaranya yaitu penelitian yang berjudul "*Forecasting Agriculture Water Consumption Based on PSO and SVM*". Penelitian tersebut bertujuan untuk meramalkan konsumsi air pertanian dengan menggunakan metode *Particle Swarm Optimization* dan *Support Vector Machine.* Permasalahan tersebut akan ditentukan oleh konstansitas data yang diolah dan bagaimana hasil aliran data tersebut. Dalam penelitian ini PSO digunakan untuk mencari nilai optimal parameter SVM, dimana SVM merupakan metode yang digunakan untuk peramalan konsumsi air pertanian tersebut. Adapun parameter SVM yang dioptimasi diantaranya yaitu nilai C, ε dan σ. Hasil nilai MAPE dari peramalan menggunakan metode PSO-SVM ini lebih akurat dibandingkan dengan metode BPNN yaitu 0,72% : 3,34%. Dengan diasumsikan bahwa apabila nilai *forecasting error* suatu data semakin kecil, maka peramalan tersebut semakin akurat (Zhang et al., 2014).

Sedangkan untuk metode *Support Vector Regression* dan *Improve Particle Swarm Optimization* pernah sebelumnya diterapkan dalam penelitian (Zou et al., 2015) sebagai solusi untuk membantu membangun mekanisme pendukung dalam mendirikan suatu pemodelan peramalan dalam sistem proyeksi kinerja. Penelitian yang berjudul "*Fault Prediction Method Based on SVR of Improved PSO*". Penelitian tersebut bertujuan untuk mengetahui cara kerja metode *Support Vector Regression* dengan *Improve Particle Swarm Optimization* dalam permasalahan *time series* dan *regresi non-linear*. Untuk mendapatkan optimasi global yang terbaik di dalam SVR, penelitian ini mempekerjakan metode IPSO sebagai pengoptimalan parameter. Proses dalam penelitian ini yaitu memprediksi 51 poin kesalahan dari suatu proses Tennessee-Eastman. Proses tersebut merupakan suatu simulasi yang dibuat oleh sebuah pabrik kimia untuk memberikan proses industri yang realistis. Dimana data tersebut telah terbentuk suatu pola kesalahan ruang, kemudian dibagi kembali menjadi sub ruang. Penelitian ini menghasilkan nilai prediksi kesalahan dari metode PSO dan IPSO, terbukti dengan metode IPSO nilai prediksi kesalahan yang dihasilkan mencapai 0.05.Dalam penelitian tersebut terbukti bahwa metode SVR-IPSO menjadi metode yang bisa dihandalkan dibanding dengan metode SVR-PSO.

SVR sendiri sebenarnya memiliki suatu konsep untuk membuat hyperplane yang maksimal dengan tujuan agar memperoleh data-data yang nantinya akan menjadi *support vector*. Dimana dengan metode *Support Vector Regression* juga

terbukti dapat dan mampu mengatasi masalah *overfitting*. Biasanya masalah yang sering terjadi dalam SVR ialah dalam menentukan parameter yang optimal. Sehingga untuk mengatasi maalah tersebut adalah denggan menggunakan metode *Improved Particle Swarm Optimization*. Metode IPSO ini mampu mendapatkan parameter SVR yang optimal. Bahkan IPSO mampu memberikan hasil optimum pada data uji berukuran kecil serta dengan waktu yang relatif cepat. Sehingga pada penelitian ini, penulis mengusulkan metode *Support Vector Regression* dan *Improved Particle Swarm Optimization* sebagai solusi untuk melakukan peramalan konsumsi air. Perbandingan metode penelitian yang akan dilakukan dengan penelitian dari masing-masing referensi studi literatur ditunjukkan pada Tabel 2.1.

Tabel 2.1 Perbandingan Objek dan Metode

No	Judul	Objek	Metode	Output
1.	*Forecasting Agriculture Water Consumption Based on PSO and SVM*	*Agriculture Water*	PSO and SVM	Hasil peramalan konsumsi air untuk pertanian menggunakan metode PSOSVM lebih baik dibandingkan dengan menggunakan metode BPNN, dengan nilai MAPE 0.72% : 3.34%.
2.	*Fault Prediction Method Based on SVR of Improved PSO*	Nilai jumlah poin kesalahan dari proses TE (Tennessee-Eastman)	*Support Vector Regression* (SVR) dan *Improved PSO*	- Hasil prediksi kesalahan menggunakan IPSO dan SVR menunjukkan bahwa algoritma ini tidak hanya menyediakan konvergensi cepat tetapi juga efisien yang baik - Nilai Peramalan
3.	Usulan Penulis: Peramalan Konsumsi Air Menggunakan Metode *Support Vector Regression* dengan *Improve Particle Swarm Optimization*	Konsumsi Air	*Support Vector Regression* dan *Improve Particle Swarm Optimization*	Koefisien Regresi yang dihasilkan oleh Algoritma SVR-IPSO yang optimum mampu menyeleseikan peramalan konsumsi air.

Tabel perbandingan input, proses dan output dari referensi-referensi penelitian sebelumnya dengan penelitian penulis ditunjukkan pada Tabel 2.2.

Tabel 2.2 Perbandingan Input, Proses dan Output

Judul	*Forecasting Agriculture Water Consumption Based on PSO and SVM*	*Fault Prediction Method Based on SVR of Improved PSO*	Usulan Penulis: Peramalan Konsumsi Air Menggunakan Metode *Support Vector Regression* dengan

			Improve Particle Swarm Optimization
Input	*Agriculture Water*	- Banyak Partikel - Nilai Bobot inersia - Koefisien akselerasi - Banyak iterasi - Nilai interval	Data Konsumsi Air PDAM Kota Malang
Proses	- Inisialisasi parameter PSO - Evaluasi Nilai *fitness* - Penghentian kondisi : - Proses evolusi akan berhenti hingga mencapai kondisi satisfied - Jika tidak, maka akan menghasilkan partikel baru atau terus mengulang	- Inisialisasi parameter dan nilai input - mengevaluasi nilai *fitness* - Pemrosesan data menggunakan *Transformasi Wavelet* - Determinasi parameter optimasi SVR - Pelatihan dan pengujian dengan SVR	- Inisialisasi parameter SVR dan IPSO - Evaluasi nilai *fitness* - pembaruan xPosition (posisi) dan pembaruan velocity (kecepatan) - Mencari nilai *pbest* dan nilai *gbest* kemudian melakukan perangkingan - Pelatihan dan pengujian metode SVR
Output	Hasil nilai MAPE dari peramalan menggunakan metode PSO-SVM ini lebih akurat dibandingkan dengan metode BPNN yaitu 0,72% : 3,34%	Metode SVR-IPSO merupakan metode yang dapat dihandalkan dibandingkan metode SVR-PSO	- Nilai parameter SVR yang optimal - Hasil peramalan tingkat konsumsi air - nilai hasil MAPE

2.2 Air

2.2.1 Definisi Air

Air merupakan unsur zat atau unsur materi yang menjadi kebutuhan utama (primer) dalam semua bentuk kehidupan atau aktifitas makhluk hidup. Permukaan bumi hampir 71% nya bahkan ditutupi oleh air. Air yang tersedia di bumi mencapai angka 1,4 triliun kubik (330 juta mil^3). Bahkan diperkirakan pada kutub utara dan selatan planet mars juga memiliki sejumlah besar air. Wujud air terbagi atas wujud padat (es), wujud cair (air) dan wujud gas (uap air). Air merupakan satu-satunya zat yang secara alami terdapat di permukaan bumi dalam ketiga wujudnya tersebut (Supijatno., Chozin Ah., Sopandie Di., Tri., Junaedi A., Lubis I., 2012).

Air sebagai materi esensial di dalam kehidupan. Hal ini ditunjukkan dengan begitu pentingnya dan begitu jelasnya kebutuhan makhluk hidup terhadap air. Namun keperluan air sehari-hari didalam kehidupan lingkungan atau kehidupan rumah tangga ternyata memiliki jangkauan yang berbeda di setiap tingkatannya. Disini dapat diasumsikan bahwa apabila taraf kehidupan seseorang semakin tinggi makan kebutuhan seseorang atau makhluk hidup akan air juga akan semakin

meningkat. Kemudian jumlah penduduk di dunia juga setiap harinya bertambah, dapat diasumsikan pula bahwa ketika penduduk bertambah otomatis kebutuhan akan air juga meningkat. Berdasarkan Keputusan Menteri Kesehatan Republik Indonesia Nomor 1405/menkes/sk/xi/2002 mengenai apa itu air bersih merupakan air yang masuk dalam syarat kualitas sesuai dengan UU yang dapat digunakan untuk melakukan aktifitas dan dapat digunakan untuk memasak atau minum (Wachjar A., Angga R., 2013).

Tubuh manusia sebagian besar terdiri dari air. Sekitar 73% tubuh kita itu terdiri dari air. Karena air merupakan zat pembentuk tubuh manusia, sebagai pengangkut dan pelarut bahan makanan penting. Kemudian untuk menjalankan kehidupan sehari-hari air juga digunakan untuk mencuci, mandi, memasak, dll. Air juga difungsikan sebagai pembangkit tenaga, alat transportasi, sebagai irigasi, dan sejenisnya. Dari sini kita tahu bahwa kebutuhan air untuk makhluk hidup memang sangat mutlak. Semakin maju tingkat kebudayaan masyarakat maka penggunaan air makin meningkat (Briawan D., Rachma P., Annisa K., 2011).

Jumlah sumber air minum yang telah memenuhi syarat sebagai air yang layak untuk diminum semakin lama semakin berkurang karena perbuatan manusia (Supijatno., Chozin Ah., Sopandie Di., Tri., Junaedi A., Lubis I., 2012). Pemenuhan kebutuhan air dapat dipenuhi dengan sumber air yang ada dari dalam tanah, air permukaan, dan air hujan. Namun dari 3 sumber air itu, air yang berasal dari tanah yang paling banyak digunakan karena relative kecil terkena pencemaran. Oleh karena itu terciptanya pihak PDAM dalam memenuhi kebutuhan air manusia.

2.2.2 Kualitas AIR

Standart air digunakan untuk mengukur kualitas air dengan berbagai jenis air yang ada. Setiap jenis air yang ada diukur konsentrasi kandungan unsur yang tercantum dalam standart kualitas. Dengan kata lain standart kualitas dapat digunakan sebagai tolak ukur (Wachjar A., Angga R., 2013).

Standart kualitas air bersih merupakan ketentuan dari Permenkes RI No. 416/MENKES/PER/IX/1990. Tujuan dari standart kualitas air ini adalah dalam rangka perlindungan, dan pemeliharaan kesehatan masyarakat. Kebutuhan air yang bersih sehari-hari sebaiknya memiliki kondisi yang jernih, tidak ada rasa, tidak bau, tidak berwarna, dan suhu yang sesuai standart. Apabila syarat-syarat yang telah disebutkan di atas atau satu dari syarat yang telah disebutkan tidak memenuhi standar kualitas air maka dapat dipastikan bahwa air itu tidak layak digunakan karena mengandung zat-zat berbahaya yang menyebabkan air menjadi bau, memiliki rasa, dan tidak jernih (Briawan D., Rachma P., Annisa K., 2011).

Sebagai badan organisasi kesehatan internasional, WHO mengeluarkan suatu auturan mengenai syarat kualitas air bersih yang meliputi beberapa bagian. Diantaranya adalah kualitas dari segi fisik, segi kimia, dan biologi. Hal ini dijadikan pedoman oleh Negara anggota lainnya. Namun setiap Negara juga dapat memiliki syarat kualitas air tersendiri yang disesuaikan dengan bagaimana keadaan dari Negara itu sendiri (Supijatno., Chozin Ah., Sopandie Di., Tri., Junaedi A., Lubis I., 2012).

Parameter air bersih yang dapat dipakai untuk kebutuhan dan aktifitas manusia ialah sebagai berikut, yaitu:

- Syarat fisik, adapun kriteria persyaratan fisik untuk air berkualitas adalah :
  a) Jernih atau air dalam kondisi tidak keruh.
  b) Tidak berwarna.
  c) Rasa yang tawar, apabila air ketika di minum memiliki rasa maka menunjukkan air tidak dalam kondisi yang sehat. Rasa tersebut disebabkan oleh kandungan garam yang masih larut dalam air atau bisa juga karena adanya kandungan asam, baik itu organik atau anorganik.
  d) Tidak memiliki bau.
  e) Memiliki temperatur yang pas (panas suhunya normal dan sejuk).
  f) Tidak mengandung bahan-bahan padat
- Syarat kimia
  a) pH (derajat kesamaan)

  digunakan dalam proses penyaringan air. Apabila terlalu banyak mengandung pH akan mengakibatkan terbentuknya racun di dalam air.
  b) Zat organik
  c) Aluminium

  Akan mengakibatkan air memiliki rasa apabila mengandung zat aluminium.
  d) Tingkat kesadahan air
  e) Besi

  Apabila air banyak mengandung besi akan mengakibatkan perubahan warna pada air menjadi kuning serta mengakibatkan rasa logam besi kepada air. Batas maksimal yang terkandung di dalam air adalah 1,0 mg/1.
  f) Nitrat dan nitrit
  g) Sulfat

  Apabila kandungan sulfat tinggi maka dapat menyebabkan kerak air saat direbus dalam panci. Juga mengakibatkan bau pada pipa.
  h) surfaktan
- Syarat biologis :
  a) Tidak terdapat bakteri-bakteri penyakit di dalamnya.

2.3 *Particle Swarm Optimization* (PSO)

Penulis akan membahas Landasan pustaka mengenai *Particle Swarm Optimization* pada sub bab 2.4.1 sampai 2.4.2.

2.3.1 Definisi *Particle Swarm Optimization*

Particle Swarm Optimization merupakan algoritma berbasis populasi. Algoritma PSO diperkenalkan pertama kalinya oleh ilmuwan Kennedy dan temannya Eberhart di tahun 1995. Algoritma *Particle Swarm Optimization* ini memiliki tingkat kemiripan yang cukup banyak dengan metode optimasi berbasis populasi lainnya seperti: *algoritma genetika* dan *evolutionary strategies*. Algoritma PSO dan GA diawali adanya suatu populasi yang terdiri dari berbagai jenis individu. Dimana jenis individu tersebut dibangkitkan secara acak atau random yang selanjutnya akan dilaukan pencarian solusi paling baik dengan memperbaiki beberapa individu untuk generasi tertentu. Sama halnya dengan GA, partikel yang ada di dalam PSO juga tidak dapat mati. Hal ini disebabkan karena kecepatan dan posisi partikelnya akan selalu mengalami perbaruan. Dan perbaruan ini akan terus dilakukan untuk setiap iterasinya. Diharapkan dengan ini akan menghasilkan pemecahan atau solusi baru yang lebih baik. PSO juga terkenal karena kesederhanaan implementasinya dan kecepatan kemampuan untuk kovergen (Ratnaweera et al., 2004).

Filosofi dari pengembangan algoritma dan teori ini terinspirasi dari perilaku sekelompok sosial burung yang sedang mencari makanan di satu daerah tertentu dengan secara random (Ning, 2010; Sengupta et.al, 2014), sedangkan makanan yang dicari oleh sekelompok burung tersebut hanya dimiliki oleh daerah itu namun sekelompok burung tersebut tidak tahu dimana makanan tersebut berada. Namun sekelompok burung tersebut tahu di setiap iterasi makanannya, sehinnga mereka memiliki strategi yang dapat digunakan yaitu mengikuti burung-burung yang posisi atau keberadaanya lebih dekat dengan makanan yang mereka cari (Kinoshita et al., 2014).

Keuntungan dari Algoritma PSO, (Liu et al., 2014):

1. Deskripsinya mudah
2. Mudah untuk diimplementasikan
3. Konvergensi dengan kecepatan tinggi
4. Memiliki tingkat komputasi yang rendah
5. Memori yang digunakan kecil

Pada PSO, setiap partikel mencari solusi yang optimal untuk mencari fungsi objektif dalam ruang pencarian. Posisi vektor dan kecepatan vektor partikel ke -*i* dan dimensi ke -*D* dalam ruang pencarian tersebut, direpresentasikan sebagai Vektor pertama yaitu : $X_i = x_{i1}, x_{i2}, x_{i3}, ..., x_{iD}$.

Vektor X : merupakan vektor yang menyimpan posisi dari solusi yang paling baik sejauh ini.

Sedangkan Vektor *P* $P_i = (p_{i1}, p_{i2}, p_{i3}, ..., p_{iD})$.

Vektor *P* : merupakan vektor untuk menyimpan posisi dari solusi terbaik sejauh yang ditemukan oleh partikel.

Vektor V : $V_i = v_{i1}, v_{i2,} v_{i3}, ..., v_{iD}$ merupakan vektor yang berisi arah yang menyatakan kemana partikel akan terbang, sedangkan solusi terbaik dalam sebuah populasi dinyatakan dengan $P_g = (P_{g1}, P_{g2}, P_{g3}, ..., P_{gD})$ (Hu et al., 2010).

2.3.2 Algoritma *Particle Swarm Optimization*

Algoritma PSO ini diawali dengan sejumlah partikel yang telah diinisialisasi sebelumnya. Selanjutnya untuk posisi setiap partikel akan dievaluasi dengan nilai *error rate* atau fungsi *fitness* yang sebelumnya telah ditentukan. Kecepatan dari setiap partikel akan berubah sesuai dengan posisi, dengan kata lain bahwa setiap partikel kecepatan terbangya berubah dengan mempertimbangkan posisi paling baik yang pernah dilalui. Kecepatan juga dipengaruhi posisi terbaik di antara semua partikel.

Berikut persamaan 2.4 untuk update kecepatan dan posisi :

$$v_{id} = w * v_{id} + c_1 * r(p_{il} - x_{id}) + c_2 * r * (p_{gl} - x_{id}) \quad (2.12)$$

$$x_{id} = x_{id} + v_{id} \quad (2.13)$$

Untuk r_1 dan r_2 merepresentasikan random number dengan batas [0,1], dan c_1 dan c_2 merepresentasikan akselerasi konstan. Berikut penjabaran untuk fungsi lainnya (Liu, 2013) :

i : merupakan inisialisasi untuk partikel keberapa

d : merupakan dimensi keberapa

c_1 : merupakan laju belajar untuk komponen *cognition* (kecerdasan individu)

c_2 : merupakan laju belajar untuk komponen *social* (hubungan antar individu)

p_{il} (pbest) : adalah nilai *fitness* terbaik yang dihasilkan sejauh ini

p_{gl} (gbest) : merupakan partikel dengan *fitness* terbaik di dalam populasi

r : adalah bilangan acak dalam interval [0,1].

Untuk *Velocity* pada setiap dimensi memiliki nilai batasan maksimum yang diinisialisasi dengan V_{max}. Nilai V_{max} sebelumnya telah diinisialisasi. Apabila percepatan velocity melebihi nilai V_{max}, maka hasil velocity tersebut nilainya dianggap sama dengan nilai V_{max}. untuk inisialisasi nilai-nilai vektor velocity, nilai nilai tersebut dibangkitkan secara random dengan batas range $[-V_{max}, V_{max}]$. Vmax merupakan nilai maksimun yang dapat diberikan pada V_i (Ratnaweera et al., 2004). Secara singkat algoritma PSO akan dijelaskan pada Gambar 2.3

```
Algoritma PSO
Inisialisasi Partikel
      Do
      For setiap partikel
      Hitung fitness value
            If fitness value > pbest
            Perbarui pbest dengan fitness value
      End
      Pilih partikel yang memiliki fitness value terbaik
      diantara semua partikel sebagai gbest
            For setiap partikel
            Hitung kecepatan(Persamaan 2.20)
            Perbarui posisi (Persamaan 2.21)
      End
      Hingga kondisi pemberhentian tercapai
```

Gambar 2.3 Algoritma PSO
Ilka (2012)

Berikut merupakan proses atau langkah di dalam PSO (Liu et al, 2013):

1. Menginisialisasi secara random *n* posisi vektor dengan masing-masing sebesar *m* partikel.
2. Menginisialisasi *n velocity* $\{V_k, k = 1,2,...,n\}$ secara random
3. Mengevaluasi nilai *fitness* di setiap partikel
4. menghitung kecepatan dari masing – masing partikel
5. menghitung pembaruan posisi dari masing-masing partikel
6. Melakukan perbandingan *fitness value* dengan partikel *global best* P_{gl} , untuk setiap partikel
7. Jika nilai *fitness ke i* lebih besar dari nilai *global best* , maka nilai diperbarui dan simpan nilai *fitness* ke I tersebut sebagai nilai P_{gl} yang baru
8. Update kecepatan dan posisi setiap partikel
9. Mengevaluasi fungsi nilai *fitness*
10. Melakukan pembaruan untuk P_{gl} dan P_{il}
11. Ulangi langkah ketiga sampai 9 yang nantinya terus melakukan perulangan hinga mencapai kondisi pemberhentian yang telah ditetapkan
12. Melakukan update partikel terbaik
13. Mencapai penemuan solusi dengan menemukan partikel paling terbaik

Keterangan :

i = merupakan notasi untuk partikel

k = merupakan notasi untuk waktu

X_k = pembaruan partikel posisi

V_k = pembaruan partikel kecepatan

P_{gl} = *global best*

P_{il} = *local best*

2.4 Algoritma *Support Vector Regression* (SVR)

Support Vector Regression (SVR) merupakan algoritma pengembangan dari SVM di dalam kasus regresi output yang berupa bilangan riil atau kontinue. SVM merupakan salah satu metode *machine learning* yang memiliki fitur yang baik serta akurasi yang tinggi dan optimal dalam menyelesaikan masalah (Wang et al., 2008). Algoritma SVM juga dijadikan sistem pembelajaran dengan menggunakan ruang hipotesis. Dimana ruang hipotesis tersebut berupa fungsi linier yang memiliki dimensi tinggi, yang kemudian dilatih dengan algoritma teori optimasi dengan mengimplementasikan *learning bias*. Algortima SVM digeneralisasi untuk melakukan pendekatan fungsi yang kemudian dikenal dengan algoritma SVR. Konsep SVR didasarkan pada *risk minimization*, yaitu untuk mengestimasi suatu fungsi dengan cara meminimalkan batas atas dari *generalization error*, sehingga SVR mampu mengatasi *overfitting* (Yasin, 2014). Tujuan dari SVR adalah untuk menemukan sebuah fungsi *f(x)* sebagai suatu *hyperplane* (garis pemisah) berupa fungsi regresi yang mana sesuai dengan semua input data dengan sebuah *error ε* dan membuat *ε* seminimal mungkin (Meesad, 2013).

Misal terdapat sebuah data latih yaitu $\{(X_1, y_1), (X_2, y_2), \ldots, (X_N, y_N)\}$ dengan $X_i \in R^m$ dan $y_i \in R$. Dimana X_i merupakan input vektor dan y_i adalah representasi nilai skalar target. Tujuan dari SVR adalah untuk mengembangkan fungsi *f(x)* yang akan digunakan untuk memprediksi nilai output berdasarkan variabel input. Di dalam SVR, SVR mencari fungsi yang disebut dengan fungsi $\varepsilon - SVR$ dimana fungsi ini memiliki nilai deviasi ε dari semua nilai keluaran aktual yang diperoleh dari pelatihan (Ling mei., 2009). Untuk kasus regresi nonlinear, misalkan *f(x)* berupa:

$$f(x) = w \bullet x + b \qquad (2.14)$$

Keterangan rumus:

$f(x)$ = Nilai prediksi
b = Nilai bias
w = Bobot
X = Data

$w \bullet x$ adalah hasil dot. Nilai bias merupakan nilai kalkulasi dari *support vector.* Dalam permasalahan optimasi konveks, dijabarkan dalam persamaan berikut:

Fungsi Minimalisasi $\frac{1}{2} \| \omega \|^2$

Dengan syarat kondisi sebagai berikut $\begin{cases} y_i - f(x_i) \le \varepsilon \\ f(x_i) - y_i \le \varepsilon \end{cases}$

Keterangan rumus:

$f(x)$ = Nilai prediksi

y_i = Nilai Aktual (2.15)

Kemudian apabila terdapat permasalahan dimana tidak ada ε yang presisi untuk semua data, maka variabel ξ_i dan ξ_i^* akan digunakan untuk meminimalisasi jumlah dari $\| \omega \|$ dalam rangka meningkatkan *flatness* dari sebuah fungsi. Berikut penjabaran optimasinya:

Minimalisasi $\frac{1}{2} \| \omega \|^2 + C \sum_{i=1}^{N} \left(\xi_{i+} \xi_i^* \right)$

Dengan syarat kondisi $\begin{cases} y_i - f(x_i) \le \varepsilon + \xi_i \\ f(x_i) - y_i \le \varepsilon + \xi_i^* \\ \xi_i^*, \xi_i \ge 0, C > 0 \end{cases}$ (2.16)

C merupakan sebuah konstanta dalam mendefinisikan pertukaran antara *flatness* fungsi dan *error*. Biasanya sering disebut dengan fungsi ε *-insensitive loss* dan dideskripsikan seperti fungsi berikut:

$$| \xi |_\varepsilon = \begin{cases} 0, | f(x) - y | \le \varepsilon \\ | f(x) - y | - \varepsilon \end{cases} \quad (2.17)$$

Permasalahan optimasi konveks juga dapat diselesaikan dengan bentuk dual seperti fungsi berikut:

Maks $-\frac{1}{2} \sum_{i,j=1}^{l} (\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*)(x \bullet x_j) + \sum_{i=1}^{l} y_i (\alpha_i - \alpha_i^*) - \varepsilon \sum_{i=1}^{l} (\alpha_i - \alpha_i^*)$

Dengan syarat kondisi $\sum_{i=1}^{l} (\alpha_i - \alpha_i^*) = 0, 0 \le \alpha_i, \alpha_i^* \le C$ (2.18)

α_i dan α_i^* digunakan untuk pelatihan ke-*i* yang akan memberikan nilai pada vektor bobot yang didefinisikan sebagai berikut:

$$w = \sum_{i=1}^{l} (\alpha_i - \alpha_i^*) x_i \quad (2.19)$$

Berikutnya akan diperoleh suatu persamaan regresi sebagai berikut:

$$f(x) = \sum_{i=1}^{l} (\alpha_i - \alpha_i^*)(x * x_i) + \lambda^2 \quad (2.20)$$

Keterangan rumus:

$f(x)$ = Nilai prediksi

α_i^*, α_i = Nilai Lagrange Multiplier

x_i, x = Data

λ^2 = variabel skalar

Karena seiring perkembangan dimensi yang semakin bertambah mengakibatkan fitur menjadi banyak dan proses pemetaanpun menjadi lama. Maka permasalahan tersebut dapat diselesaikan dengan menggunakan fungsi kernel yang sesuai untuk ruang fitur berdimensi tinggi.

$$K(x, x_i) = \phi(x) \bullet \phi(x_i) \quad (2.21)$$

Sehingga terbentuk fungsi regresi non linier sebagai berikut:

$$f(x) = \sum_{i=1}^{l} (\alpha_i - \alpha_i^*)(K(x, x_i) + \lambda^2) \quad (2.22)$$

Keterangan rumus:

$f(x)$ = Nilai prediksi

x_i, x = Data

λ^2 = variabel skalar

K = *Kernel* yang digunakan

Di dalam algoritma SVR, *Support Vector* merupakan data *training* yang terletak pada dan diluar batas *ƒ* dari fungsi keputusan, karena itu jumlah *Support Vector* menurun dengan naiknya *ε*.

2.4.1 Algoritma S*equential Learning*

Pada setiap perhitungan fungsi SVR, terdapat proses *sequential learning*. Algoritma *sequential* untuk regresi dapat dilihat di bawah ini (Vijayakumar, 1999):

Nomenklatur yang digunakan pada algoritma ini adalah:

$[R]_{ij}$ merupakan matriks *Hessian*.

E_i merupakan nilai *error ke i*.

y_i merupakan nilai kelas *Support Vector* (berupa nilai aktual data uji).

γ merupakan *learning rate* yang didapatkan dari

$$\frac{\text{konstanta } learning\ rate}{\max(\text{diagonal matriks Hessian})} \quad (2.23)$$

Dimana konstanta *learning rate* akan dibagi dengan nilai maksimum dari diagonal matriks *kernel*.

$\delta\alpha_i^*, \delta\alpha_i$ merupakan variable tunggal, bukan bentuk dari perkalian δ dan α_i atau perkalian δ dan α_i^*

α_i^*, α_i merupakan nilai *Lagrange Multiplier*

C merupakan nilai kompleksitas

ε merupakan nilai epsilon

λ merupakan variabel skalar

Berikut proses *sequential learning* dalam perhitungan fungsi SVR :

1. Inisialisasi $\alpha_i = 0, \alpha_i^* = 0$. Hitung

$$[R]_{ij} = K(x_i, x_j) + \lambda^2 \quad (2.24)$$

untuk $i, j = 1,...,n$

2. Untuk setiap data latih, hitung:

2.1 $$\mathrm{E}_i = y_i - \sum_{j=1}^{l} (\alpha_j^* - \alpha_j) R_{ij} \quad (2.25)$$

2.2 $$\delta\alpha_i^* = \min\{\max[\gamma(\mathrm{E}_i - \varepsilon), -\alpha_i^*], C - \alpha_i^*\} \quad (2.26)$$

$$\delta\alpha_i = \min\{\max[\gamma(-\mathrm{E}_i - \varepsilon), -\alpha_i], C - \alpha_i\} \quad (2.27)$$

2.3 $$\alpha_i^* = \alpha_i^* + \delta\alpha_i^* \quad (2.28)$$

$$\alpha_i = \alpha_i + \delta\alpha_i \quad (2.29)$$

3. Kembali ke langkah kedua, sampai kondisi iterasi maksimum atau sampai dengan kondisi : $\max(|\delta\alpha_i|) < \varepsilon$ dan $\max(|\delta\alpha_i^*|) < \varepsilon$
4. Dikatakan *support vector* apabila data memenuhi syarat $(\alpha_i^* - \alpha_i) != 0$
5. Menggunakan fungsi regresi seperti yang dijabarkan pada persamaan
6. Selesai

2.5 Improve Particle Swarm Optimization (IPSO)

Improve Particle Swarm Optimization merupakan pengembangan dari algoritma PSO dalam rangka mencegah konvergensi dini dengan menambahkan suatu mekanisme baru. Salah satu permasalahan pada PSO adalah setelah beberapa iterasi, hampir semua partikel mempunyai nilai yang sama sebelum tercapainya titik optimum yang diinginkan. Terdapat banyak cara atau metode untuk dapat mengatasi masalah ini. Mekanisme ini dilakukan dengan memasukkan beberapa partikel acak pada selang/interval iterasi tertentu. Penentuan berapa partikel acak yang harus dimasukkan dan berapa selang iterasi yang terbaik harus ditentukan melalui percobaan pendahuluan (Mahmudy, 2015).

Struktur dari PSO sebenarnya sederhana, kecepatan dari PSO juga cepat. Tetapi dalam proses optimasi jika partikel menemukan solusi optimum lokal maka

partikel lain akan ikut bergerak sehingga sulit untuk keluar dari solusi optimal lokal untuk menemukan solusi optimal global Dalam IPSO terdapat dua ide untuk menghindari fenomena premature. Yang pertama adalah menghindari partikel jatuh ke dalam solusi lokal yang sama. Partikel selalu memiliki keanekaragaman tertentu. Yang kedua adalah mengambil langkah untuk membuat partikel keluar dan meningkatkan keragaman ketika partikel berada dalam solusi lokal (keanekaragaman rendah) (Wang et al., 2011). Maka IPSO mengatasi kelemahan yang ada dalam PSO.

Adapun langkah – langkah dasar dari algoritma IPSO (Zhang et al., 2009):

1. Inisialisasi kelompok partikel secara acak.
2. Setiap partikel melakukan pembaruan kecepatan dan posisi.
3. Kondisi apakah partikel baru diterima atau tidak bergantung pada hasil seleksi simulasi.
4. Ulangi iterasi sampai pada kondisi *satisfied*.

Langkah-langkah di atas sama halnya dengan proses yang ada dalam metode PSO. Namun di dalam metode IPSO yang diajukan dalam paper ini, perbedaan terletak pada adanya penambahan koefisien λ yang menjadi faktor konvergensi yang ditambahkan di depan inersia weight (w) dimana kondisi $\lambda = \sin^3 \alpha$ dengan $\alpha = [0, \pi/8]$. Sehingga untuk melakukan pembaruan kecepatan dan posisi dengan improvisasi akan dijabarkan dalam persamaan sebagai berikut :

$$v_{il} = \lambda w v_{il} + s_1 \times rand \times (p_{il} - x_{il}) + s_2 \times rand \times (p_{gl} - x_{il}) \quad (2.28)$$

$$x_{il} = x_{il} + \lambda w_1 v_{il} \quad (2.29)$$

Keterangan rumus :

w = *inersia weight* (fungsi pemberat)

V_{id} = Kecepatan

c = faktor pemberat

P_{id} = partikel posisi

P_{gd} = *global best*

x = posisi terakhir *i* pada iterasi tersebut

Karena nilai velocity yang masalahnya adalah cenderung terjebak pada optimum lokal, maka diperlukan tambahan nilai bobot inertia dalam rangka meningkatkan performasi pada PSO. Berikut dijelaskan pada Persamaan:

$$w = w_{\min} + (w_{\max} - w_{\min}) \frac{(t_{\max} - t)}{t_{\max}} \quad (2.30)$$

Keterangan Rumus :

$t_{\max}$: Iterasi maksimum yang sebelum melakukan training IPSO telah diinisialisasi.

t : Merupakan iterasi yang sedang berjalan, yaitu iterasi 1 sampai *n.*

$w_{\min}$: Bobot minimum untuk membatasi w yang akan diolah .

$w_{\max}$: Bobot maksimum untuk membatasi w yang akan diolah.

2.6 IPSO-SVR

Parameter σ, ε, dan C merupakan parameter SVR pelatihan yang akan digunakan dalam proposal skripsi ini. Metode IPSO digunakan sebagai penentuan parameter SVR tersebut. Di dalam IPSO terdapat dua kondisi utama yaitu: pembaruan kecepatan dan pembaruan posisi. Kecepatan tiap partikel di masing-masing generasi akan berubah sesuai dengan posisinya, dimana setiap partikel akan merubah kecepatan terbang dengan mempertimbangkan posisi terbaik yang pernah dilewati.

Berikut langkah – langkah proses penggunaan IPSO yang di optimasi dengan SVR:

1. Inisialisasi parameter partikel dan parameter IPSO.
2. Menghasilkan parameter partikel asli yang terdiri dari σ, ε, dan C.
3. Menentukan parameter IPSO, w_1, c_1, c_2.
4. Mengevaluasi nilai *fitness*.

$$\frac{1}{MAPE+1} \tag{2.29}$$

5. Hitung setiap partikel nilai fungsi
6. Hitung kecepatan dari masing-masing partikel.
7. Update nilai posisi setiap partikel .
8. Melakukan perbandingan *fitness value* dengan partikel global yang terbaik.
9. Melakukan update posisi dan kecepatan tiap partikel.
10. Berhenti deteksi: dari awal proses peramalan sampai berhenti batas kriteria yang ditentukan. Jika tidak, ulangi kembali dari langkah kedua.

2.7 Metode *Kernel*

Proses dari *kernel* adalah untuk memetakan fitur ke dimensi yang lebih tinggi. Diharapkan bahwa apabila fitur berada dalam ruang yang lebih tinggi, dimensi data akan lebih terstruktur.

Karena terlalu banyaknya teknik-teknik untuk persoalan data mining yang sudah banyak dikembangkan dengan masalah teknik data linier, dan mengakibatkan terbatasnya pemecahan persoalan yang berbentuk linier. Dan dengan metode *Kernel* inilah sebuah data di dalam ruang input dipecah ke dalam ruang yang memiliki dimensi lebih tinggi (*feature space*) (Cheng, 2013; Rajkumar, 2013). Berikut beberapa metode *Kernel* yang biasa digunakan :

- Linier :

$$K(x, y) = x.y \tag{2.30}$$

- Polynomial :

$$K(x, y) = (x.y + c)^d \quad (2.31)$$

- Radial Basis Function (RBF) :

$$K(x, x_i) = \exp(-\frac{\| x - x_i \|^2}{2\sigma^2}) \quad (2.32)$$

- Tangent hyperbolic (sigmoid):

$$K(x, y) = \tanh(\sigma(x.y) + c) \quad (2.33)$$

x dan y merupakan data yang berpasangan yang menjadi bagian dari semua dara latih. Kemudian untuk parameter *c* dan *d* yang merupakan konstanta. Sebuah Teori Mercer mengatakan bahwa fungsi *kernel* harus memenuhi syarat kontinyu dan memiliki nilai yang *positive definite* (nilainya lebih dari nol) (Kuo, 2013; Xing, 2013).

2.8 Normalisasi Data

Di dalam proses perhitungan peramalan konsumsi air, diperlukan proses untuk normalisasi data. Tujuan dari normalisasi data adalah untuk proses penskalaan nilai atribut dari data sehingga data bias jatuh dalam range tertentu. Sehingga data yang digunakan akan berada pada jarak tertentu (Patro et al., 2015). Metode normalisasi yang akan digunakan dalam penelitian ini adalah metode *Min-Max*. Berikut adalah persamaan dari *Min-Max Normalization* :

$$x' = \frac{(x - x_{\min})}{x_{\max} - x_{\min}} \quad (2.34)$$

Keterangan rumus :

x_{min} = Nilai minimum dari dataset
x_{max} = Nilai maksimum dari dataset
x = Nilai data yang akan di normalisasi
x' = Nilai hasil normalisasi data yang berkisar antara 0 dan 1

2.9 Denormalisasi Data

Denormalisasi data merupakan kebalikan dari proses normalisasi data. Tujuan denormalisasi adalah untuk mengembalikan nilai data menjadi nilai riil kembali. Berikut penjabaran proses pencarian nilai denormalisasi :

$$x = (x'x(x_{\max} - x_{\min})) + x_{\min} \quad (2.35)$$

2.10 Nilai Evaluasi (MAPE)

Nilai evaluasi adalah nilai untuk mengetahui sampai batas mana akurasi peramalan yang akan dihasilkan. Nilai evaluasi merupakan selisih di antara nilai peramalan dengan nilai aktual yang biasanya disebut dengan istilah *error rate*. Kemudian untuk mendapakan nilai fitness, adalah nilai yang dihasilkan dari 1/*error*

rate. Dengan asumsi bahwa apabila semakin besar nilai fitness maka akan dikatakan semakin baik. Salah satu nilai evaluasi yang biasa digunakan adalah *Mean Absolute Percentage Error* (MAPE). MAPE merupakan jenis metode pendekatan nilai error yang pemahamannya mudah. Berikut persamaan rumus MAPE akan dijelaskan pada persamaan berikut (Liu et al., 2008):

$$MAPE = \sum_{i=1}^{n} \left| \frac{y_t - y't}{y_t} \right| \times \frac{100}{n} \qquad (2.34)$$

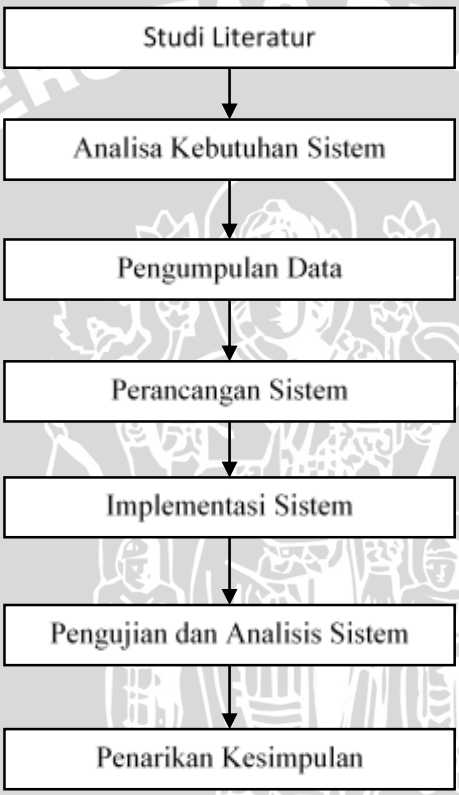
Keterangan:

y_t = Nilai aktual

y'_t = Nilai hasil peramalan

BAB 3 METODOLOGI

Pada metodologi penelitian ini berisi pembahasan tentang langkah-langkah dan rancangan yang digunakan dalam pembuatan sistem optimasi peramalan konsumsi air menggunakan metode *Support Vector Regression* dengan *Improve Particle Swarm Optimization*. Tahapan penelitian ini meliputi studi literatur, pengumpulan data, analisis data, analisis sistem, perancangan sistem, implementasi, pengujian sistem, dan kesimpulan yang ditunjukkan pada Gambar 3.1 berikut ini.



Gambar 3.1 Diagram Alir Tahapan Penelitian

3.1 Studi Literatur

Dalam penelitian ini, studi literatur dilakukan untuk mengumpulkan informasi yang bersumber dari buku, naskah penelitian, dan informasi dari internet. Studi literatur ini membahas mengenai teori-teori yang digunakan untuk menunjang penulisan skripsi. Teori-teori yang dibutuhkan diantaranya tentang:

1. Sistem *prediction* atau *forecasting* konsumsi air, yang didalamnya meliputi penjelasan umum sistem dan algoritma yang diimplentasikan.

2. *Support Vector Regression* (SVR) yang diimplentasikan ke dalam sistem *forecasting* atau *prediction.*
3. *Improve Particle Swarm Optimization* (IPSO) yang digunakan untuk mengoptimasi metode dalam *forecasting* atau *prediction.*
4. Teori mengenai konsumsi air.
5. Optimasi SVR-IPSO.
6. Pemrograman menggunakan bahasa java

3.2 Analisa Kebutuhan

3.2.1 Deskripsi Umum Sistem

Sistem yang akan dibuat bertujuan untuk melakukan peramalan terhadap konsumsi air di PDAM Kota Malang. Diharapkan dapat berguna bagi pihak PDAM dalam melakukan pembagian volume distribusi air sehingga terjadi keseimbangan antara air yang didistribusikan dengan tingkat konsumsi air oleh masyarakat. Objek yang digunakan atau data yang di inputkan untuk penelitian ini adalah data konsumsi air penduduk Kota Malang yang akan diramalkan menggunakan metode *Support Vector Regression* dan akan dioptimasi menggunakan metode *Improved Particle Swarm Optimization*.

3.2.2 Spesifikasi Kebutuhan Sistem

1. Spesifikasi Kebutuhan *Hardware*
   - Laptop HP core i3
   - RAM 2 GB
   - DDR3
   - VGA 1 GB

2. Spesifikasi Kebutuhan *Software*
   - Microsoft windows 10.
   - Microsoft office 2013 sebagai pengolah dokumentasi dan manual perhitungan
   - JAVA sebagai bahasa pemrograman yang akan digunakan untuk menjalankan sistem.
   - Menggunakan Netbeans 8.0 sebagai editor sistem.
   - Microsoft Visio 2010 sebagai pengolah dalam pembuatan diagram alir.

3. Spesifikasi Kebutuhan Data
   - Data konsumsi air di PDAM Kota Malang

3.3 Pengumpulan Data

Di dalam tahap ini merupakan tahap dimana penulis mulai mengumpulkan data yang akan digunakan sebagai pengembangan dan pengujian. Berikut data yang dibutuhkan :

1. Data yang digunakan dalam penelitian adalah data konsumsi air PDAM Kota Malang, 7 tahun terakhir mulai dari tahun 2008 sampai tahun 2014. Data tersebut didapatkan dari pihak kantor PDAM Malang.

3.4 Perancangan Sistem

Perancangan sistem dilakukan untuk lebih memudahkan implementasi, analisis dan pengujian. Berikut ini adalah langkah-langkah dalam perancangan :

1. Perancangan Antarmuka Pengguna
   Untuk memudahkan pengguna dalam menggunakan sistem yang akan dibangun
2. Perancangan Pengujian
   Pengujian yang dilakukan meliputi uji coba iterasi SVR, uji coba waktu eksekusi program, pengujian iterasi IPSO, uji coba nilai parameter, uji coba variasi jumlah data training dan variasi jumlah data latih.

3.5 Implementasi Sistem

Implementasi sistem dilakukan berdasarkan perancangan yang telah dibuat. Implementasi dilakukan dengan editor Netbeans IDE 8.0 serta menggunakan bahasa pemrograman java dengan, dan *tools* pendukung lainnya. Implementasi optimasi peramalan konsumsi air di PDAM Kota Malang meliputi:

1. Penerapan metode *Support Vector regression* dan *improve particle swarm optimization* dalam program dengan menggunakan bahasa pemrograman JAVA.
2. Pembuatan antarmuka program.
3. Memasukkan data yang diperlukan ke database MySQL.

3.6 Pengujian dan Analisis Sistem

Pengujian sistem disini bertujuan untuk menunjukkan program telah bekerja sesuai dengan sistem yang diharapkan. Beberapa uji coba yang akan dilakukan untuk mengevaluasi program ini antara lain:

1. Uji coba terhadap nilai parameter yang paling optimal dengan nilai *error rate* paling kecil
2. Uji coba iterasi SVR
3. Uji coba iterasi IPSO
4. Uji coba batas nilai parameter SVR yang dihasilkan pada IPSO
5. Uji coba terhadap variasi jumlah banyaknya data training dan data latih

3.7 Penarikan Kesimpulan

Penarikan kesimpulan diambil setelah penulis menyelesaikan tahapan-tahapan dalam penulisan skripsi yaitu, tahap rancangan implementasi dan pengujian. Kesimpulan ditarik ketika penulis telah melakukan evaluasi dan analisis dari hasil pengujian yang telah dilakukan. Tujuan dari penarikan kesimpulan ini adalah untuk memberikan jawaban terhadap rumusan masalah yang telah disusun

penulis. Saran merupakan tahap terakhir dalam penulisan skripsi dengan tujuan dapat melakukan perbaikan apabila terdapat kesalahan yang terjadi serta bertujuan dalam memberikan masukan apabila dilakukan pengembangan untuk penelitian berikutnya.

BAB 4 PERANCANGAN

Bab ini menjelaskan tahapan penelitian, formulasi permasalahan, siklus algortima *Support Vector Regression* dengan *Improve Particle Swarm Optimization*, siklus penyelesaian peramalan konsumsi air di PDAM menggunakan algortima *Support Vector Regression* dengan *Improve Particle Swarm Optimization*, perhitungan manual, dan perancangan antar muka

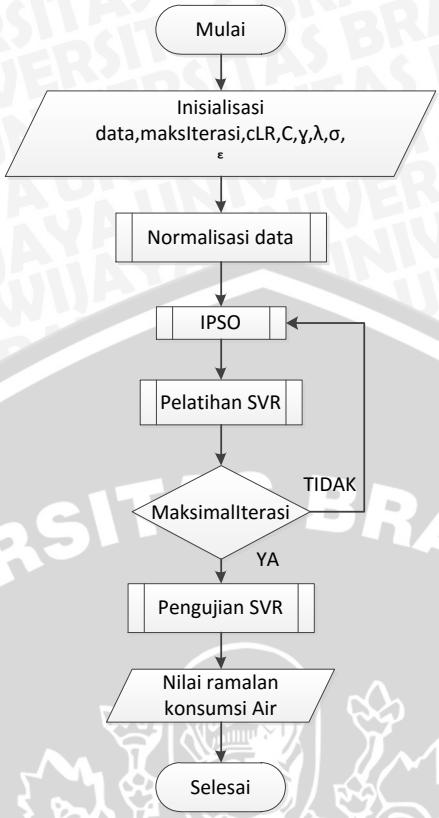
4.1 Formulasi Permasalahan

Peramalan konsumsi air dilakukan dengan membentuk model regresi, yang selanjutnya akan dilakukan pengujian untuk mendapatkan nilai *error rate*. Dengan *error rate* itu kita dapat mengetahui bagaimana hasil akurasi yang dibentuk atau biasa disebut dengan nilai evaluasi. Tujuan dari evaluasi adalah untuk meminimalkan tingkat error hasil peramalan yang dilakukan.
Pada penelitian ini, digunakan data konsumsi air sejak tahun 2008 hingga tahun 2014 dengan bentuk data bulanan yang ditunjukkan pada Tabel 4.1 berikut ini :

Tabel 4.1 Konsumsi Air PDAM Kota Malang

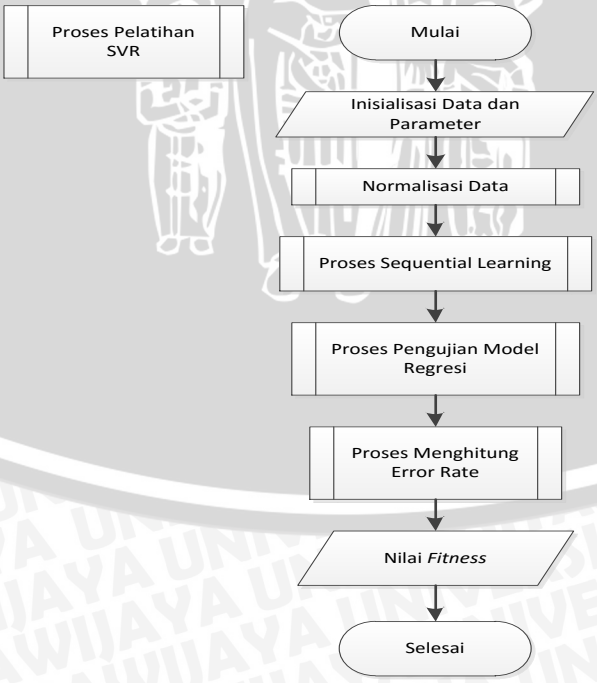
TAHUN/ BULAN	1	2	3	4	5	.....	12
2008	1684536	1627438	1592618	1702344	1611804	.....	1558064
2009	1773087	1658690	1680750	1765834	1799518	.....	1751939
2010	1807659	1752567	1773785	1901589	1858807	.....	1894268
2011	1894655	1902542	1867345	1893539	1905850	.....	1943646
2012	1986071	2014645	1908264	2192472	1999456	.....	2374281
2013	2127658	1987364	2068653	2118364	2092653	.....	2198034
2014	2235453	2153579					

Berdasarkan data konsumsi air di PDAM kota Malang diatas, dapat ditentukan data latih yang akan digunakan untuk pelatihan SVR sesuai dengan langkah-langkah yang terdapat di dalam arsitektur sistem SVR-IPSO pada Gambar 4.1 berikut ini :



Gambar 4.1 Arsitektur Sistem SVR-IPSO

4.2 Penyelesaian Masalah Menggunakan *Support Vector Regression*



Gambar 4.2 Diagram Alir Metode SVR

4.2.1 Perhitungan SVR

1. Memilih Fitur yang Digunakan

   Untuk menentukan jumlah konsumsi air yang terjadi pada bulan berikutnya maka akan diformulasikan dengan menggunakan data 4 bulan sebelumnya.

Tabel 4.2 Konsumsi Air Berdasarkan 4 Bulan Sebelumnya

No	Data	X_1	X_2	X_3	X_4	Y
1	Mei 2010	1807659	1752567	1773785	1901589	1858807
2	Juni 2010	1752567	1773785	1901589	1858807	1825106
3	Juli 2010	1773785	1901589	1858807	1825106	1869328
4	Agustus 2010	1901589	1858807	1825106	1869328	1820043
5	September 2010	1858807	1825106	1869328	1820043	1780592

2. Menentukan Data Latih dan Data Uji

   Dalam proses perhitungan SVR, harus ditentukan data yang akan digunakan sebagai data latih dan data uji. Untuk penelitian ini, digunakan data latih dan data uji yang sama sebanyak 5 data kemudian dinormalisasi sesuai dengan persamaan yang akan disajikan pada 4.4 berikut :

$$normalisasi = (dataTraining - MinDataSet) \div (MaxDataSet - MinDataSet)$$

$$= (1807659 - 1417037) \div (2374281 - 1417037)$$

$$= 0.408069416$$

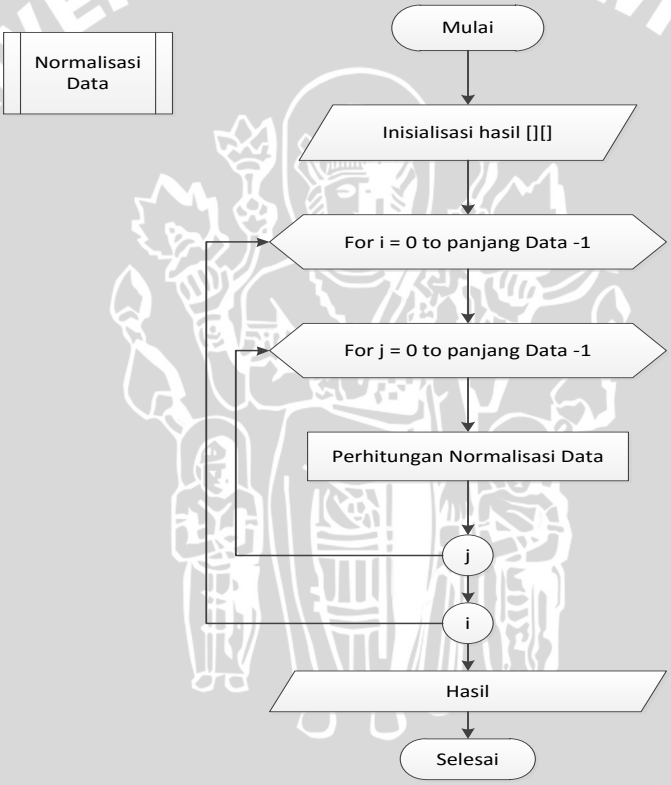
Tabel 4.3 Data Latih

No	Data	X_1	X_2	X_3	X_4	Y
1	Mei 2010	0.408069416	0.350516692	0.372682409	0.506194868	0.461501979
2	Juni 2010	0.350516692	0.372682409	0.506194868	0.461501979	0.426295699
3	Juli 2010	0.372682409	0.506194868	0.461501979	0.426295699	0.472492907
4	Agustus 2010	0.506194868	0.461501979	0.426295699	0.472492907	0.421006556
5	September 2010	0.461501979	0.426295699	0.472492907	0.421006556	0.379793449

Tabel 4.4 Data Uji

No	Data	X_1	X_2	X_3	X_4	Y
1	Mei 2010	0.408069416	0.350516692	0.372682409	0.506194868	0.461501979
2	Juni 2010	0.350516692	0.372682409	0.506194868	0.461501979	0.426295699
3	Juli 2010	0.372682409	0.506194868	0.461501979	0.426295699	0.472492907
4	Agustus 2010	0.506194868	0.461501979	0.426295699	0.472492907	0.421006556
5	September 2010	0.461501979	0.426295699	0.472492907	0.421006556	0.379793449

Adapun diagram alir dari proses normalisasi data dijabarkan seperti pada Gambar 4.3 berikut :



Gambar 4.3 Diagram Alir Normalisasi Data

3. Menentukan Nilai Parameter SVR

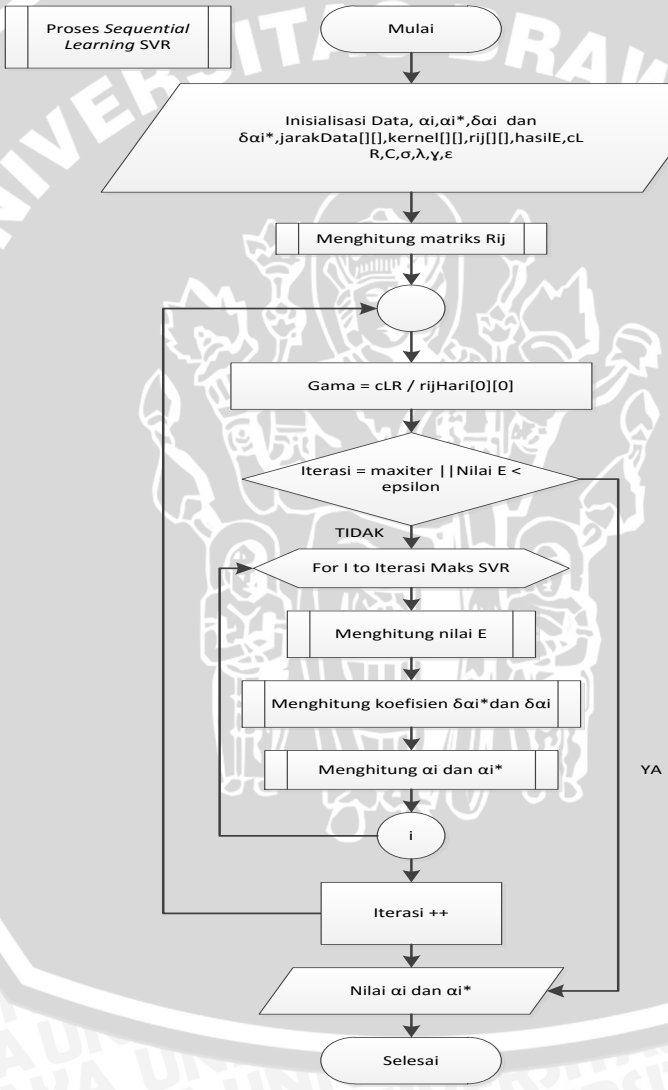
SVR memiliki beberapa nilai parameter perhitungan, dimana masing-masing parameter memilki batas nilai berbeda-beda. Berikut beberapa parameter yang digunakan dalam perhitungan metode SVR, yang akan disajikan pada Tabel 4.5 di bawah ini :

Tabel 4.5 Nilai Parameter SVR

cLR	C	ε	γ	λ	σ
0.003	150	0.0004	0.001829	0.8	0.08

4. Proses Perhitungan *Sequential Learning*

Proses *Sequential learning* merupakan proses untuk melakukan regresi dari konsumsi air. Berikut proses diagram dari perhitungan sequential learning yang dijabarkan pada Gambar 4.4 :



Gambar 4.4 Diagram Alir Proses *Sequential Learning*

- Langkah Pertama

Inisialisasi nilai α_i dan α_i^* dengan nilai 0 sebanyak jumlah data latih.

- Langkah Kedua

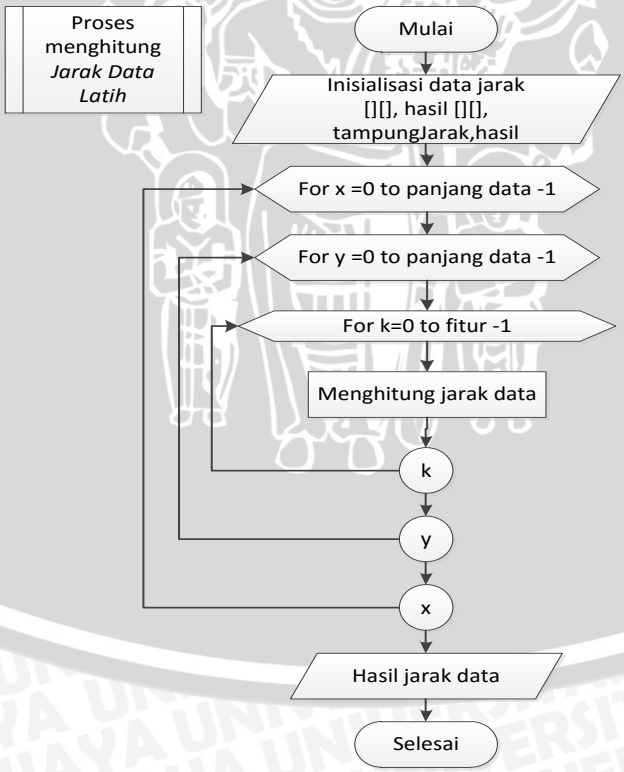
  Membangun model regresi, berikut ini contoh perhitungan data latih ke-3 dengan data latih ke-5. Pertama, menghitung jarak antar data. Tabel 4.6 merupakan hasil dari proses perhitungan jarak data latih :

$$\| x_3 - x_5 \|^2 = (0.372682 - 0.0.461502)^2 + (0.506194 - 0.426295)^2$$
$$+ (0.461502 - 0.472493)^2 + (0.426296 - 0.421007)^2$$
$$= 0.014421569$$

Tabel 4.6 Hasil Komputasi Jarak Data Latih

Data ke	1	2	3	4	5
1	0	0.023626666	0.039760728	0.025956545	0.025816681
2	0.023626666	0	0.021553832	0.038629288	0.01796782
3	0.039760728	0.021553832	0	0.023196695	0.014421569
4	0.025956545	0.038629288	0.023196695	0	0.008021963
5	0.025816681	0.01796782	0.014421569	0.008021963	0

Adapun diagram alir dari proses perhitungan jarak antar data dijabarkan seperti pada gambar 4.5 berikut :



Gambar 4.5 Diagram Alir Perhitungan Jarak Data

- Langkah Ketiga

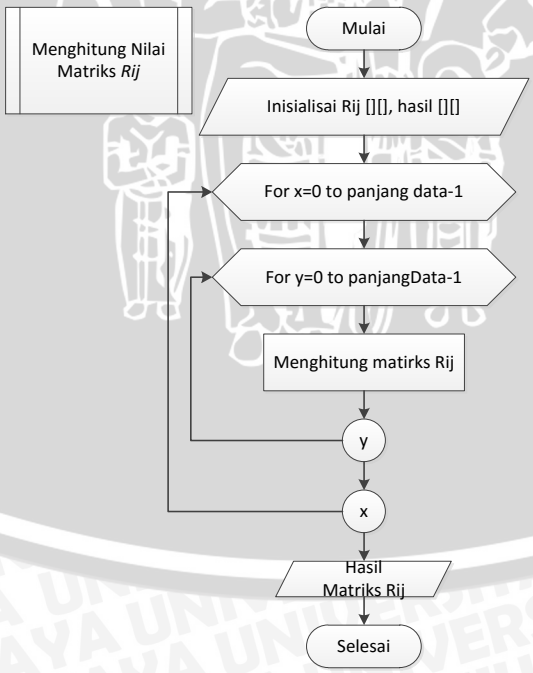
Dari hasil jarak data yang sudah didapatkan. Berikut ini merupakan contoh perhitungan menggunakan data latih ke-3 dan data latih ke-5. Tabel 4.7 merupakan hasil model regresi yang telah dibentuk. Sesuai dengan persamaan 2.22, kernel yang digunakan adalah fungsi kernel RBF.

$$[R]_{3,5} = K(x_3, x_5) + \lambda^2 = \exp\left(-\frac{0.014421569}{2\times(0.08)^2}\right) + 0.8^2 = 0.964105872$$

Tabel 4.7 Hasil Komputasi Matriks R_{ij}

Rij					
DATA	**1**	**2**	**3**	**4**	**5**
1	1.64	0.797893695	0.684765977	0.771617526	0.773063587
2	0.797893695	1.64	0.825649808	0.688903166	0.885677409
3	0.684765977	0.825649808	1.64	0.803287667	0.964105872
4	0.771617526	0.688903166	0.803287667	1.64	1.174343793
5	0.773063587	0.885677409	0.964105872	1.174343793	1.64

Adapun diagram alir dari proses perhitungan nilai matriks *Rij* dijabarkan seperti pada Gambar 4.6 berikut :



Gambar 4.6 Diagram Alir Proses Hitung Matriks *Rij*

- Langkah Keempat

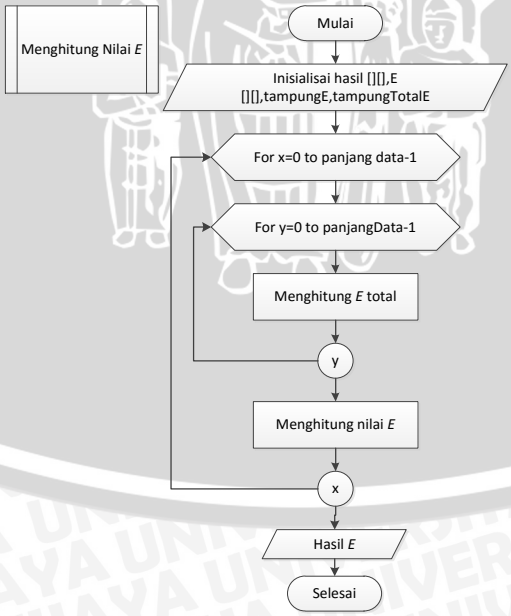
  Langkah keempat adalah melakukan perhitungan *E* sesuai dengan persamaan 2.4, sehingga nilai *E* pada iterasi 10 pelatihan SVR akan disajikan pada Tabel 4.8 sebagai berikut :

$$E_1 = 0.461501979 - ((0.007350944 - 0) * 1.6 + (0.006766145 - 0) * 0.797893695 + (0.007519207 - 0) * 0.684765977 + (0.006669537 - 0) * 0.771617526 + (0.005974436 - 0) * 0.773063587 = 0.42913392$$

Tabel 4.8 Nilai *Learning Rate* Iterasi 10

E
0.42913392
0.393239631
0.438423721
0.386679053
0.343238366

Adapun diagram alir dari proses perhitungan nilai *E* dijabarkan seperti pada Gambar 4.7 berikut :



Gambar 4.7 Diagram Alir Perhitungan Nilai *E*

- Langkah Kelima

Setelah mendapatkan nilai *E,* lakukan perhitungan $\delta\alpha_i$ dan $\delta\alpha_i^*$ pada iterasi tersebut sesuai dengan persamaan 2.26. Berikut merupakan contoh perhitungan nilai $\delta\alpha_i$ dan $\delta\alpha_i^*$ pada iterasi ke 10 dengan nilai α_i dan α_i* berasal dari iterasi ke 9.

$$\delta\alpha_i^* = (\min(\max(0.001829 * (0.42913392 - 0.0004), -0.007350944), 150 - 0.007350944))$$
$$= 0$$

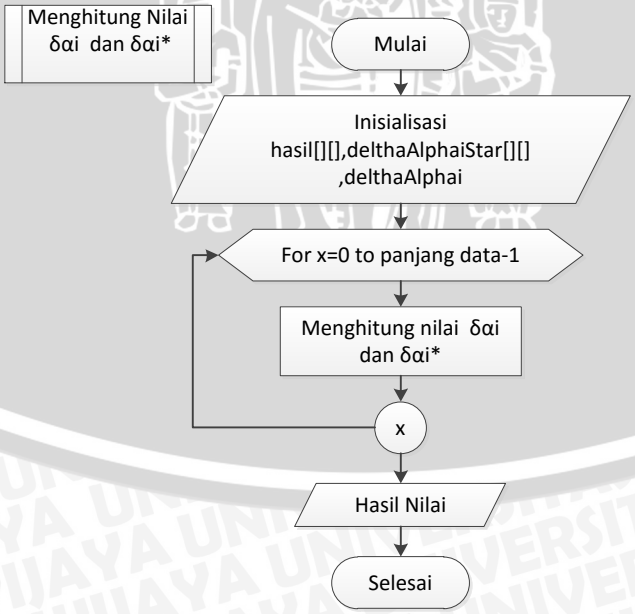
$$\delta\alpha_i = (\min(\max(0.001829 * (-0.4291339 2 - 0.0004), -0), 150 - 0)) = 0$$

Sehingga dihasilkan nilai seperti pada Tabel 4.9 berikut ini :

Tabel 4.9 Hasil $\delta\alpha_i$ dan $\delta\alpha_i^*$ Iterasi 10

$\delta\alpha_i^*$	$\delta\alpha_i$
0	0
0	0
0	0
0	0
0	0

Adapun diagram alir dari proses perhitungan nilai $\delta\alpha_i^*$ dan $\delta\alpha_i$ dijabarkan seperti pada Gambar 4.8 berikut :



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

- Langkah Keenam

  Setelah melakukan perhitungan $\delta\alpha_i^*$ dan $\delta\alpha_i$, langkah berikutnya adalah melakukan pembaruan atau update nilai α_i^* dan α_i yang seterusnya akan digunakan pada iterasi selanjutnya sesuai dengan persamaan 2.27. berikut contoh perhitungannya pada iterasi ke 10:

  $\alpha_i^* = 0.0073509444 + 0 = 0.008135213$

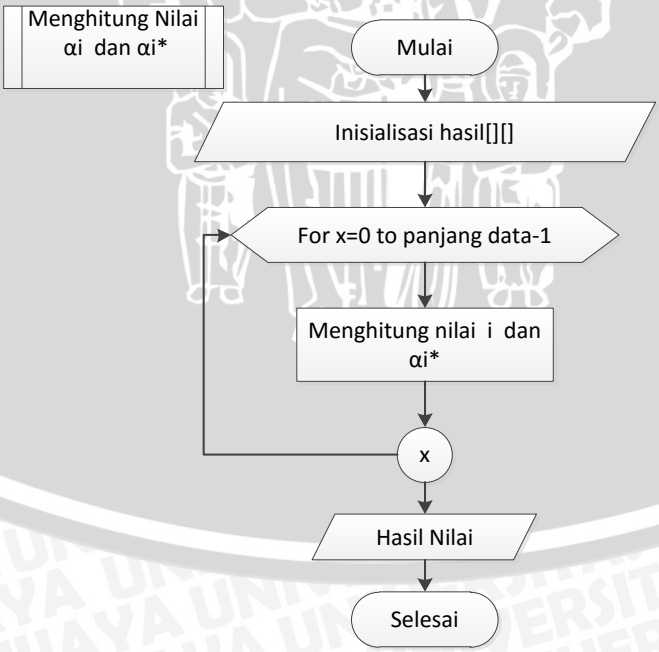
  $\alpha_i = 0 + 0 = 0$

  menghasilkan nilai pada Tabel 4.10 berikut :

Tabel 4.10 Nilai α_i dan α_i^*

α_i^*	α_i
0.008135213	0
0.007484754	0
0.00832047	0
0.007376145	0
0.006601579	0

Adapun diagram alir dari proses perhitungan nilai α_i^* dan α_i dijabarkan seperti pada Gmbar 4.9 berikut :



Gambar 4.9 Diagram Alir Perhitungan Nilai α_i^* dan α_i

5. Proses Pengujian Model Regresi

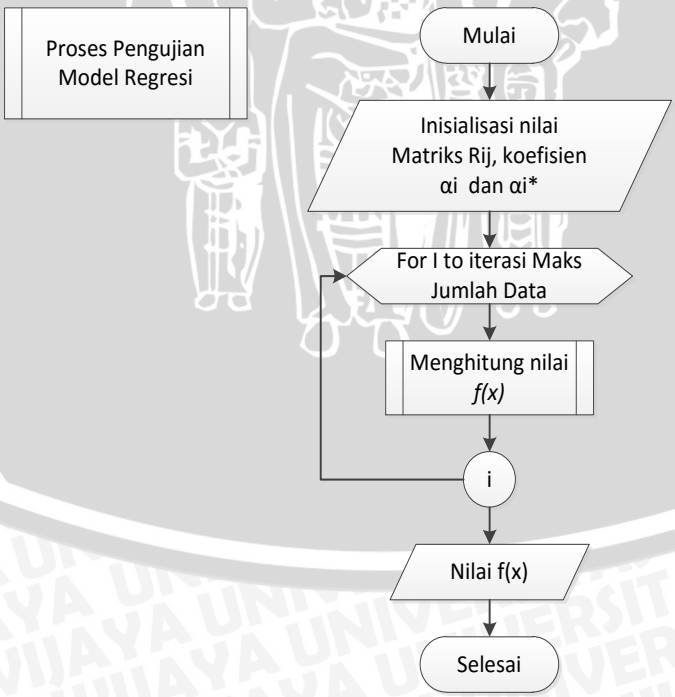
Pengujian model regresi dilakukan sesuai dengan fungsi perhitungan yang ada dalam persamaan 2.1, berikut contoh penjabaran fungsi persamaan untuk nilai peramalan data 1 yang telah dicantumkan pada Tabel 4.11 :

$$f(x) = ((0.008135213 - (0)) \times 1.64) + ((0.007484754 - (0)) \times 0.797893695) + ((0.00832047 - (0)) \times 0.684765977) + ((0.007376145 - (0)) \times 0.771617526) + ((0.006601579 - (0)) \times 0.773063587) = 0.03580634$$

Tabel 4.11 Nilai $f(x)$ Iterasi 10

KONSUMSI	$f(x)$
0.461501979	0.035806364
0.426295699	0.036564144
0.472492907	0.03768586
0.421006556	0.037966675
0.379793449	0.040428647

Proses pengujian model regresi merupakan proses perhitungan konsumsi air berdasarkan nilai-nilai yang telah didapat dari perhitungan sebelumnya. Diagram alir dari proses pengujian model regresi adalah seperti pada Gambar 4.10 berikut ini :



Gambar 4.10 Diagram Alir Pengujian Model Regresi

6. Proses Komputasi *Error Rate*

Proses terakhir adalah menentukan tingkat *error* atau biasa disebut dengan istilah *error rate*. Dalam skripsi ini penulis menggunakan nilai *Mean Absolute Percentage Error* (MAPE) sesuai dengan persamaan pada 2.24. sebelum melakukan perhitungan *error rate* maka data aktual dan data peramalan yang telah diperoleh akan di denormalisasi terlebih dahulu dengan fungsi sebagai berikut:

$$denormalisasi = (data \times (MaxDataSet - MinDtaSet)) + MinDataSet$$

$$= (0.035806364414052 \times (2374281 - 1417037)) + 1417037$$

$$= 1451312.427$$

Untuk keseluruhan data yang telah di denormalisasi, dicantumkan dalam Tabel 4.12 di bawah ini :

Tabel 4.12 Denormalisasi Data

KONSUMSI	PREDIKSI
1858807	1451312.427
1825106	1452037.808
1869328	1453111.563
1820043	1453380.371
1780592	1455737.08

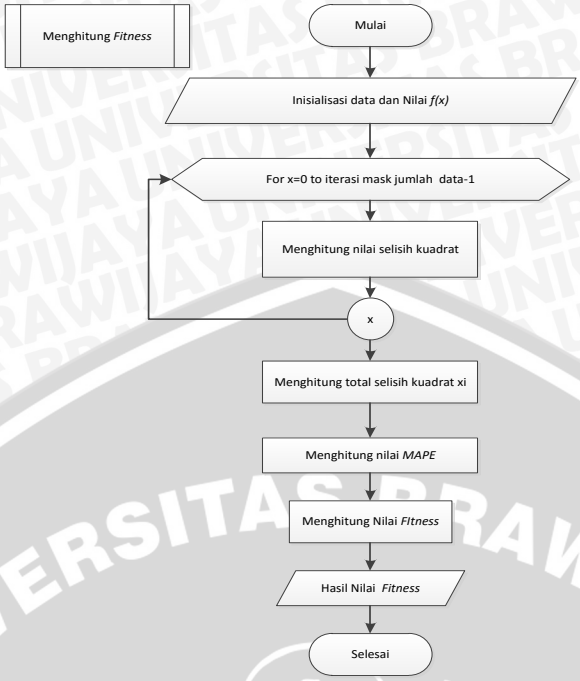
Dengan menggunakan MAPE berikut contoh untuk perhitungan *value error rate* SVR akan dijelaskan di bawah ini :

$$MAPE = \left(\left(\frac{1451312.427 - 1858807}{1451312.427}\right) + \left(\frac{1452037.808 - 1825106}{1452037.808}\right) + \left(\frac{1453111.563 - 1869328}{1453111.563}\right)\right)$$

$$= \left(\left(\frac{1453380.371 - 1820043}{1453380.371}\right) + \left(\frac{1455737.08 - 1780592}{1455737.08}\right)\right) \times \frac{100}{5}$$

$$= 25.991453334$$

Diagram alir dari proses menghitung *error rate,* akan dijabarkan Gambar 4.11 berikut ini :



Gambar 4.11 Diagram Alir Menghitung *Error Rate*

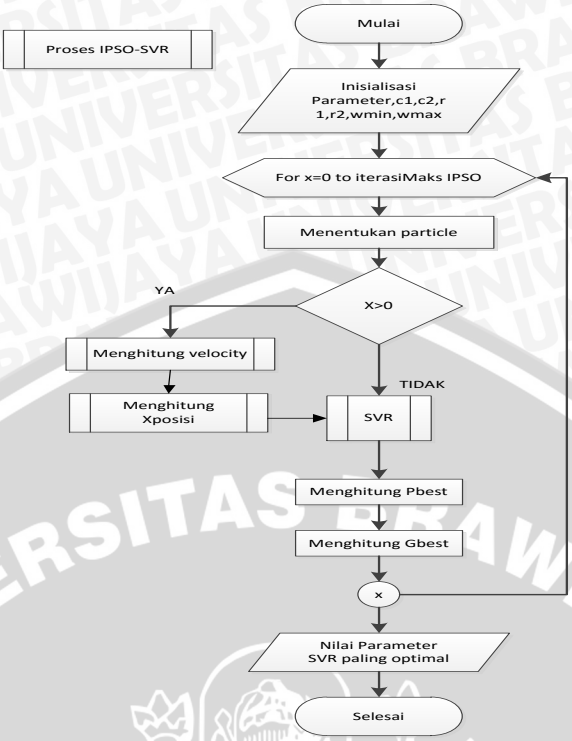
4.3 Proses Penyelesaian Menggunakan Optimasi IPSO

Pada bagian ini akan membahas proses dari perancangan sistem dalam mendapatkan solusi terbaik. Pada bagian ini penulis akan membahas proses penyelesaian masalah menggunakan optimasi IPSO. Langkah pertama dari proses ini adalah menentukan parameter awal dan memasukkan data yang digunakan. Langkah selanjutnya akan dilakukan tahap optimasi IPSO. Dimana pada tahap IPSO ini proses optimasi ini menggunakan parameter sesuai yang sudah ditetapkan.

Tujuan dari optimasi IPSO ini adalah untuk menghasilkan parameter SVR yang dapat memberikan nilai *error rate* yang lebih kecil dibanding metode regresi yang hanya menggunakan SVR saja. Setelah mendapatkan hasil nilai *error rate* langkah berikutnya adalah melakukan optimasi terhadap parameter-parameter tersebut menggunakan metode IPSO agar menghasilkan nilai error rate yang lebih baik dibandingkan dengan metode regresi SVR saja. Untuk lebih jelasnya akan dijabarkan pada Gambar 4.12.

Dalam proses perhitungan IPSO disini terdapat satu populasi. Dengan satu populasi tersebut memiliki 5 partikel yang masing masing partikel memiliki nilai parameter (C),(ε),(σ). Untuk lebih jelasnya akan dijabarkan pada Gambar 4.12.

Di dalam proses perhitungan IPSO juga terdapat koefisien baru yang tidak dimiliki oleh PSO yaitu penambahan nilai λ. Untuk nilai λ diperoleh berdasarkan nilai random yang telah dibuat dengan nilai $\lambda = \sin^3 \alpha$ yang memiliki batas syarat $\alpha = [0, \pi/8]$. Nilai λ akan berubah di setiap iterasinya.



Gambar 4.12 Diagram Alir Proses IPSO

1. Inisialisasi Populasi IPSO

   Dalam satu populasi terdapat 5 partikel yang memiliki nilai parameter Kompleksitas (C), Epsilon (ε), Sigma (σ). Nilai *fitness* dari tiap partikel diperoleh dari pelatihan metode SVR. Kemudian untuk inisialisasi nilai awal akan dihasilkan *Pbest* yang merupakan partikel itu sendiri dan *Gbest* yang akan terletak pada partikel 2 dengan nilai sesuai Tabel 4.13. Berikut merupakan hasil perhitungan *fitness* untuk iterasi 0, dengan nilai MAPE diperoleh melalui proses menghitung *error rate* pada data training SVR. Contoh perhitungan nilai *fitness*

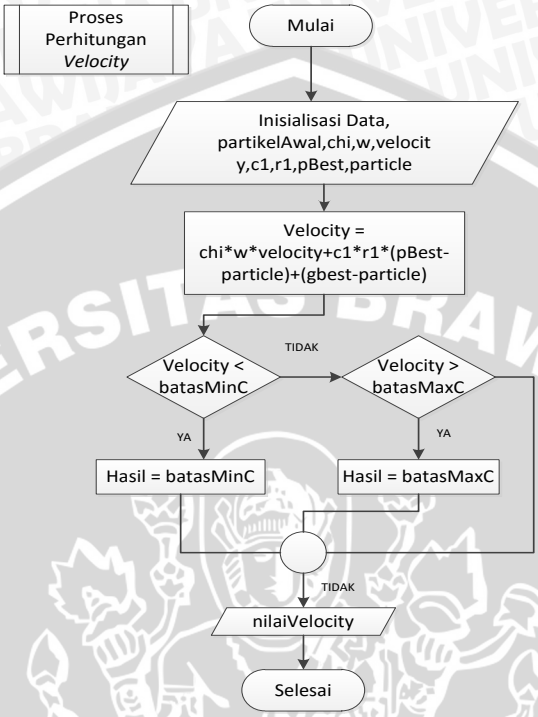
$$fitness_1 = \frac{1}{1 + 25.9914533\,4} = 0.03704876\,5$$

Tabel 4.13 Inisialisasi Populasi IPSO

ITERASI 0				
PARTICLE Ke	C	Ɛ	σ	FITNESS
PARTICLE 1	150.000000	0.000400	0.080000	0.037049
PARTICLE 2	100.000000	0.000500	0.090000	0.037279
PARTICLE 3	200.000000	0.006000	0.080000	0.036993
PARTICLE 4	300.000000	0.010000	0.700000	0.039556
PARTICLE 5	**100.000000**	**0.005000**	**0.600000**	**0.039614**

2. Perhitungan *Velocity*

Menghitung *velocity* dilakukan untuk memperbarui kecepatan masing-masing partikel. Berikut prosesnya akan dijelaskan pada Gambar 4.13 di bawah ini :



Gambar 4.13 Diagram Alir Proses Perhitungan Velocity

Sesuai dengan persamaan 2.12 maka dihasilkan nilai *Velocity* sesuai dengan Tabel 4.14. Berikut salah satu contoh perhitungan untuk merubah nilai *Velocity* dari parameter Kompleksitas (C) pada partikel 1 dengan melakukan perhitungan inertia weight sesuai dengan persamaan 2.28 :

$$v_{il} = \lambda w v_{il} + s_1 \times rand \times (p_{il} - x_{il}) + s_2 \times rand \times (p_{gl} - x_{il})$$

$$= 0.760911593 * 0.650000 * 0 + 0.3 * 0.2 * (150.000000 - 150.000000)$$

$$+ 0.7 * 0.8 * (100.000000 - 150.000000) = -6.000000$$

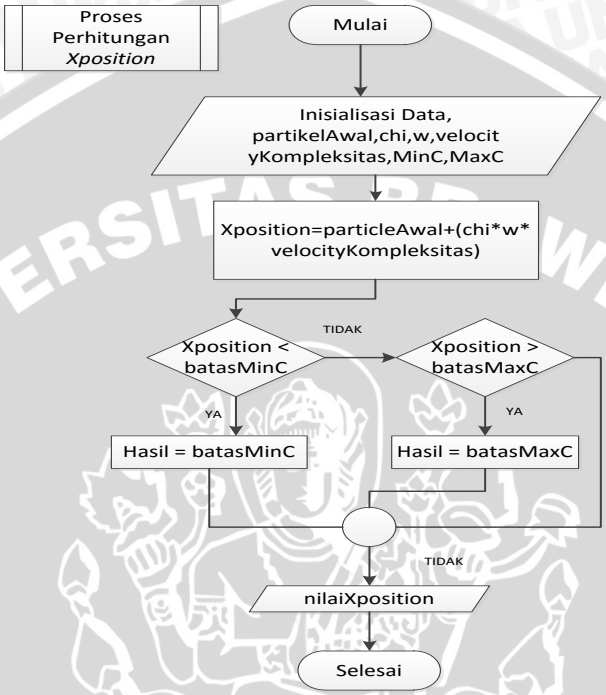
Tabel 4.14 Hasil Perhitungan Velocity

VELOCITY			
Vij	C	Ɛ	Σ
1	-6.000000	0.002576	0.291200
2	0.000000	0.002520	0.285600
3	-6.000000	-0.000060	0.291200
4	-6.000000	-0.000060	-0.000070

5	0.000000	0.000000	0.000000

3. Perhitungan *XPosition*

Perhitungan nilai posisi ini dilakukan untuk memperbarui posisi dari masing masing partikel. Berikut proses perhitungan yang telah dijabarkan pada Gambar 4.13 :



Gambar 4.14 Diagram Alir Proses Perhitungan Xposition

Sesuai dengan persamaan 2.19 berikut merupakan salah satu contoh perhitungan posisi dari Epsilon (**ε**) pada partikel 1 di iterasi 1 :

$$x_{il} = x_{il} + \lambda w_1 v_{il}$$

$$= 150.000000 + 0.760911593 \times 0.650000 \times -6.000000 = 147.032445$$

Tabel 4.15 Hasil Perhitungan X Posisi

X POSITION				
Xij	**C**	**Ɛ**	**σ**	**FITNESS**
1	147.032445	0.001674	0.224025	**0.039053**
2	100.000000	0.001746	0.231256	**0.039091**
3	197.032445	0.005970	0.224025	**0.038986**
4	297.032445	0.009970	0.699965	**0.039557**

5	100.000000	0.005000	0.600000	**0.039614**

4. Perhitungan pembaruan *Pbest* dan *Gbest*

   Untuk pembaruan nilai *Pbest* akan dilakukan di setiap iterasi. Nilai *Pbest* didapat dari perbandingan nilai *fitness* dari setiap partikel di setiap iterasinya. Dengan asumsi bahwa partikel yang memiliki nilai *fitness* terbesar itulah yang akan menjadi *Pbest*. Atau dengan kata lain adalah dengan cara membandingkan nilai *fitness* antara *x-position* dan *particle* sebelumnya. Bisa juga dengan membandingan *fitness value* antara partikel ke-*i* di iterasi ke-*j* dan partikel yang memiliki *fitness* terbesar. Dalam perbandingan tersebutlah nilai *Pbest* didapatkan.

   Berdasarkan perbandingan dari perhitungan diatas hasil *pbest* secara keseluruhan dapat dilihat pada Tabel berikut :

Tabel 4.16 Nilai PBest Iterasi 1 dan Iterasi 2

ITERASI 1		**ITERASI 2**	
PARTICLE	Nilai FITNESS	**PARTICLE**	Nilai FITNESS
PARTICLE 1	0.039053131	**PARTICLE 1**	0.039442165
PARTICLE 2	0.039091404	**PARTICLE 2**	0.039449629
PARTICLE 3	0.03898576	**PARTICLE 3**	0.039388838
PARTICLE 4	0.039556694	**PARTICLE 4**	0.039557096
PARTICLE 5	0.039614071	**PARTICLE 5**	0.039614071

   Dari tabel diatas menunjukkan beberapa partikel pada iterasi ke-2 menghasilkan *fitness value* yang lebih baik dibandingkan pada iterasi ke-1. Untuk partikel 1, partikel pada iterasi ke-2 yang menjadi *Pbest* begitu juga untuk partikel lainnya hingga partikel ke 5 posisi *Pbest* masih dimiliki oleh iterasi ke-2.

   Kemudian untuk pembaruan *Gbest (Global Best)* diperoleh dari *fitness value* terbaik yang ada dalam sebuah populasi. Dari tabel 4.16 di atas dapat dilihat bahwa dalam populasi tersebut yang menjadi Gbest untuk iterasi pertama dan iterasi kedua masih dimiliki oleh partikel 5.

5. Kondisi Berhenti

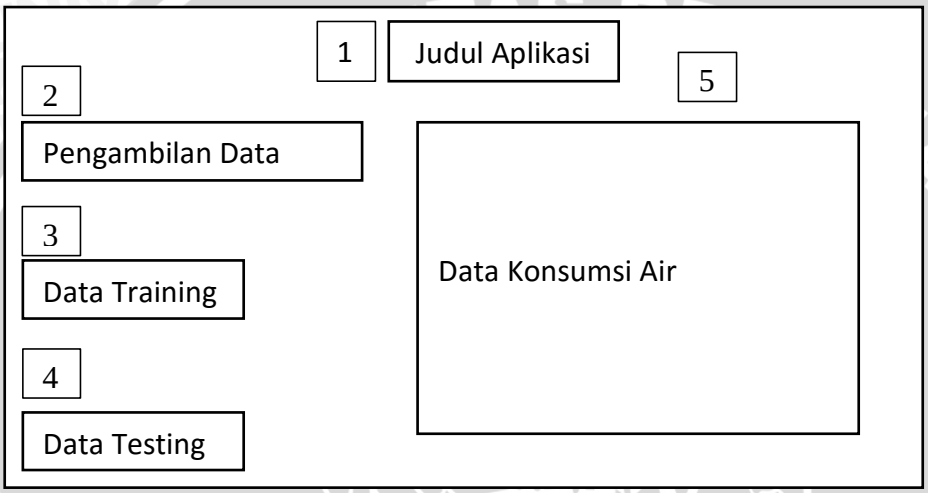
   Setelah melakukan update *Pbest* dan *Gbest*, maka langkah berikutnya adalah melakukan cek terhadap kondisi berhenti pada IPSO. Langkah ke-2 hingga

ke-4 akan terus melakukan perulangan sampai mencapai iterasi maksimum. Sehingga metode optimasi IPSO dapat menghasilkan *Gbest* sebagai solusi paling optimal dalam menentukan parameter SVR.

4.4 Perancangan User Interface

4.4.1 Rancangan Halaman Awal

Pada rancangan *user interface* untuk halaman awal ini adalah untuk mengisi data data yang akan digunakan untuk peramalan, judul aplikasi, button untuk memproses pengambilan data, dan button untuk melakukan proses peramalan data yang akan diolah. Perancangan halaman awal dapat dilihat pada Gambar 4.15 berikutini:

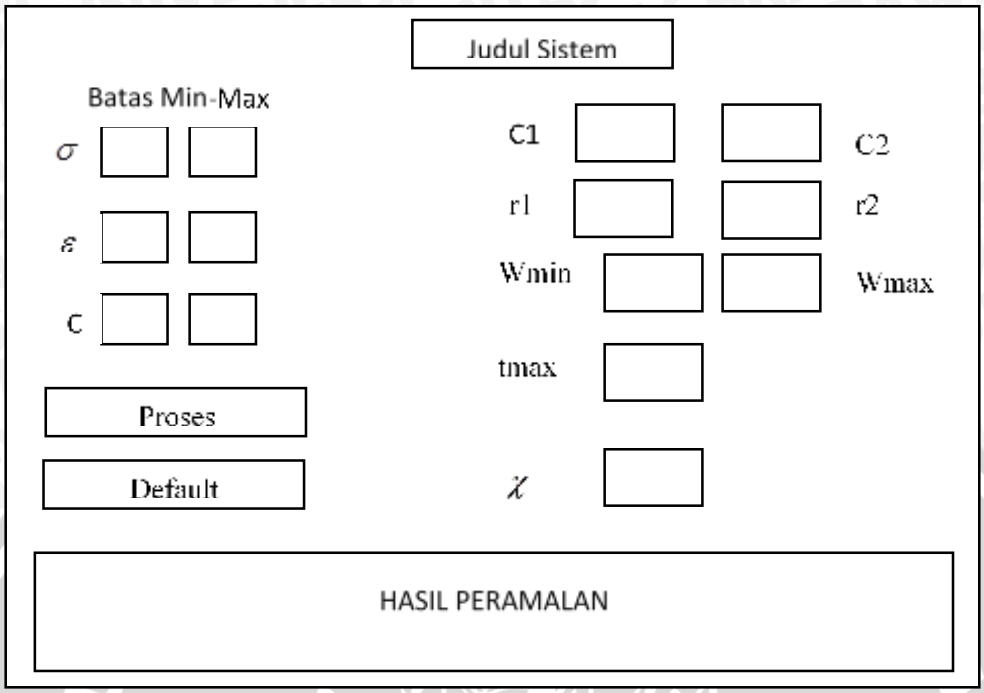


Gambar 4.15 Rancangan Halaman Data Awal

Keterangan :

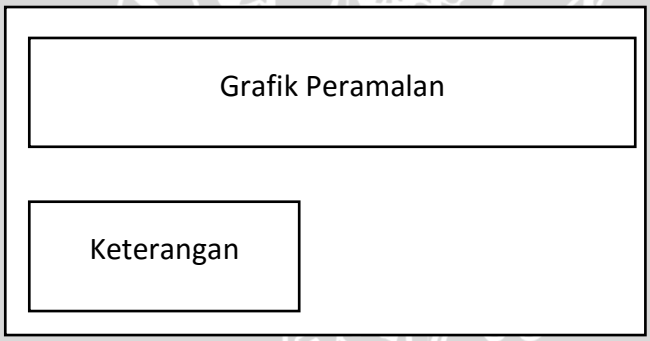
1. Judul Aplikasi yang telah dirancang.
2. Pengambilan data yang akan diolah dalam proses peramalan konsumsi air.
3. Button untuk pemrosesan data training.
4. Button untuk pemrosesan data Testing.
5. Data yang akan diolah dalam proses peramalan konsumsi air.

4.4.2 Perancangan Halaman Parameter Algoritma



Gambar 4.16 Rancangan Halaman Parameter dan Peramalan

4.4.3 Perancangan Halaman Grafik Peramalan



Gambar 4.17 Rancangan Halaman Grafik

4.5 Perancangan Pengujian dan Evaluasi

Perlunya dilakukan tahapan rancangan pengujian dan evaluasi ini adalah karena dalam melakukan peramalan konsumsi air ini sebenarnya tidak ada metode pasti yang bias digunakan. Serta untuk menentukan parameter paling optimal di dalam algoritma yang digunakan. Maka dari itu untuk melakukan evaluasi system, penulis akan melakukan beberapa uji coba diantaranya adalah :

1. Uji coba Jumlah iterasi IPSO
2. Uji coba Jumlah Iterasi SVR
3. Uji coba Jumlah Data Training

4. Uji coba Jumlah Data Testing
5. Uji coba Range Parameter *C*
6. Uji coba Range Parameter ε
7. Uji coba range parameter σ

4.5.1 Uji Coba Jumlah Iterasi IPSO

Uji coba jumlah iterasi IPSO ini digunakan untuk mendapatkan jumlah iterasi paling optimal dalam IPSO. Diharpkan agar dapat menghasilkan nilai peramalan yang baik. Rancangan uji coba iterasi IPSO dapat dilihat pada Tabel 4.17 berikut :

Tabel 4.17 Rancangan Uji Coba Jumlah Iterasi IPSO

Jumlah Iterasi IPSO	Nilai Fitness Percobaan ke-*i*										Rata-rata Fitness
	1	2	3	4	5	6	7	8	9	10	
10											
20											
30											
40											
50											
60											
70											
80											
90											
100											

4.5.2 Uji Coba Jumlah Iterasi SVR

Uji coba jumlah iterasi SVR ini digunakan untuk mendapatkan jumlah iterasi paling optimal dalam SVR, agar dapat menghasilkan nilai peramalan yang baik. Rancangan uji coba untuk mengetahui batas paling optimal dapat dilihat pada Tabel 4.18 berikut :

Tabel 4.18 Rancangan Uji Coba Jumlah Iterasi SVR

Jumlah Iterasi SVR	Nilai Fitness Percobaan ke-*i*										Rata-rata Fitness
	1	2	3	4	5	6	7	8	9	10	
50											
100											
200											
400											
500											
1000											
2000											

Jumlah Iterasi SVR	Nilai Fitness Percobaan ke-*i*										Rata-rata Fitness
	1	2	3	4	5	6	7	8	9	10	
5000											
10000											
100000											

4.5.3 Uji Coba Variasi Jumlah Data Training

Uji coba variasi banyaknya jumlah data *training* ini bertujuan untuk mengetahui jumlah data *training* yang paling optimal dalam peramalan. Rancangan uji coba variasi jumlah data training dapat dilihat pada Tabel 4.19 berikut :

Tabel 4.19 Rancangan Uji Coba Jumlah Data Training

Jumlah Data Training	Nilai Fitness Percobaan ke-*i*										Rerata Fitness
	1	2	3	4	5	6	7	8	9	10	
5											
10											
15											
20											
25											
30											
35											
40											
45											
50											

4.5.4 Uji Coba Variasi Jumlah Data Testing

Uji coba terhadap variasi jumlah data *testing* ini bertujuan untuk mengetahui pola data sequence yang dapat menghasilkan hasil peramalan terbaik. Rancangan uji coba variasi jumlah data *testing* dapat dilihat pada Tabel 4.20 berikut :

Tabel 4.20 Rancangan Uji Coba Jumlah Data Testing

Jumlah Data Training	Nilai Fitness Percobaan ke-*i*										Rerata Fitness
	1	2	3	4	5	6	7	8	9	10	
5											
10											
15											
20											
25											
30											

Jumlah Data Training	Nilai Fitness Percobaan ke-*i*										Rerata Fitness
	1	2	3	4	5	6	7	8	9	10	
35											
40											
45											
50											

4.5.5 Uji Coba Range Parameter *C*

Uji coba range parameter C ini bertujuan untuk mendapatkan range yang paling optimal sehingga dapat menghasilkan nilai peramalan yang baik. Rancangan uji coba untuk mengetahui range paling optimal dapat dilihat pada Tabel 4.21 berikut :

Tabel 4.21 Rancangan Uji Coba Range Parameter *C*

RANGE C	Nilai Fitness Percobaan ke-*i*										Rerata FITNESS
	1	2	3	4	5	6	7	8	9	10	
10-100											
101-200											
201-300											
301-400											
401-500											
501-600											
601-700											
701-800											
801-900											
901-1000											

4.5.6 Uji Coba Range Parameter ε

Uji coba range parameter ε ini bertujuan unbtuk memperoleh range batas paling optimal untuk nilai ε sehingga menghasilkan nilai peramalan yang baik. Uji coba range parameter ε ini dilakukan dengan range mulai dari 0.0001 – 0.09. Rancangan uji coba untuk mengetahui range paling optimal dapat dilihat pada Tabel 4.22 berikut :

Tabel 4.22 Rancangan Uji Coba Range Parameter ε

RANGE C	Nilai Fitness Percobaan ke-*i*										Rerata FITNESS
	1	2	3	4	5	6	7	8	9	10	
0.0001-0.01											
0.011-0.02											
0.021-0.03											
0.031-0.04											
0.041-0.05											
0.051-0.06											
0.061-0.07											
0.071-0.08											
0.081-0.085											
0.086-0.09											

4.5.7 Uji Coba Range Parameter σ

Uji coba range parameter σ ini dilakukan dengan tujuan mendapatkan range paling optimal σ sehingga dapat menghasilkan nilai peramalan yang baik. Rancangan uji coba untuk σ dapat dilihat pada Tabel 4.23 berikut :

Tabel 4.23 Rancangan Uji Coba Range Parameter σ

RANGE C	Nilai Fitness Percobaan ke - *i*										Rerata FITNESS
	1	2	3	4	5	6	7	8	9	10	
0.0001-0.01											
0.011-0.02											
0.021-0.03											
0.031-0.04											
0.041-0.05											
0.051-0.06											
0.061-0.07											
0.071-0.08											
0.081-0.085											
0.086-0.09											

BAB 5 IMPLEMENTASI SISTEM

Pada bagian bab ini akan membahas implementasi dari perancangan sistem peramalan konsumsi air menggunakan metode *Support Vector Regression* dan *Improved Particle Swarm Optimization*. Pembahasan ini meliputi beberapa bagian yaitu spesifikasi sistem, batasan sistem, implementasi algoritma dan implementasi user interface.

5.1 Spesifikasi Sistem

Spesifikasi sistem ini berdasarkan hasil perancangan yang telah diuraikan pada bab metodologi. Pada bagian implementasi sistem ini akan dibahas bagaimana spesifikasi dari sistem serta batasan dari sistem. Lingkungan implementasi sistem terdiri dari lingkungan perangkat keras dan lingkungan perangkat lunak.

5.1.1 Spesifikasi Perangkat Keras

Spesifikasi perangkat keras yang digunakan sesuai dengan Tabel 5.1 berikut ini:

Tabel 5.1 Spesifikasi Perangkat Keras

Nama Komponen	Spesifikasi
Prosesor	Intel(R) Core(TM) i3 CPU M 380 @ 2.53GHz (4 CPUs), ~2.5GHz
Memori (RAM)	2048 MB
Harddisk	500 GB

5.1.2 Spesifikasi Perangkat Lunak

Spesifikasi perangkat lunak yang digunakan pada implementasi sistem ini sesuai dengan Tabel 5.2 berikut ini:

Tabel 5.2 Spesifikasi Perangkat Lunak

Nama Komponen	Spesifikasi
Sistem Operasi	Windows 8.1 Enterprice
Bahasa Program	Java
Tools	Netbeans

5.2 Implementasi Algoritma

Implementasi algoritma dibagi menjadi 2 proses utama, yaitu implementasi algoritma *Improved Particle Swarm Optimization* dan *Support Vector Regression*.

5.2.1 Implementasi Algoritma *Improved Particle Swarm Optimization*

Implementasi algoritma *Improved Particle Swarm Optimization* meliputi 9 bagian yaitu :

a. Fungsi Perhitungan untuk Nilai Velocity Kompleksitas

```
1  public double[][] velocityKompleksitas(double chi, double
2  w, double C1, double C2, double r1, double r2, double
3  particle[][], double batasMinC,
4             double batasMaxC, int id, int jumlahParticle,
5  int tmax, double velocity[][], double pBest[][], double
6  gBest[][], int iterasi) {
7          double[][] hasil = new
8  double[tmax][jumlahParticle];
9          double nilaiVelocity[] = new
10 double[jumlahParticle];
11         for (int i = 0; i < jumlahParticle; i++) {
12             nilaiVelocity[i] = chi * w * velocity[id -
13 1][i] + (C1 * r1 * (pBest[i][0] - particle[i][0]) + C2 *
14 r2 * (gBest[iterasi][0] - particle[i][0]));
15             if (nilaiVelocity[i] < batasMinC) {
16                 nilaiVelocity[i] = batasMinC;
17                 hasil[id][i] = nilaiVelocity[i];
18             } else {
19                 if (nilaiVelocity[i] > batasMaxC) {
20                     nilaiVelocity[i] = batasMaxC;
21                     hasil[id][i] = nilaiVelocity[i];
22                 } else {
23                     nilaiVelocity[i] = nilaiVelocity[i];
24                     hasil[id][i] = nilaiVelocity[i];
25                 }
26             }
27         }
28         return hasil;
29     }
30
```

Kode Program 5.1 Perhitungan Nilai Velocity Kompleksitas

Penjelasan Kode Program 5.1 di atas adalah sebagai berikut:

1. Baris 1-6 merupakan inisialisasi nama method untuk fungsi perhitungan *velocity kompleksitas* dengan 16 variabel parameter di dalamnya
2. Baris 7-10 merupakan inisialisasi variabel yang akan digunakan dalam method
3. Baris 11-27 merupakan fungsi perulangan untuk menghitung nilai *velocity kompleksitas*

b. Fungsi Perhitungan untuk Nilai Velocity Epsilon

```
1  public double[][] velocityEpsilon(double chi, double w,
2  double C1, double C2, double r1, double r2, double
3  particle[][], double batasMinEpsilon,
```

```
    double batasMaxEpsilon, int id, int jumlahParticle, int
4   tmax, double velocity[][], double pBest[][], double
5   gBest[][], int iterasi) {
6           double[][] hasil = new
7   double[tmax][jumlahParticle];
8           double nilaiVelocity[] = new
9   double[jumlahParticle];
10          for (int i = 0; i < jumlahParticle; i++) {
11              nilaiVelocity[i] = chi * w * velocity[id -
12  1][i] + (C1 * r1 * (pBest[i][1] - particle[i][1]) + C2 *
13  r2 * (gBest[iterasi][1] - particle[i][1]));
14              if (nilaiVelocity[i] < batasMinEpsilon) {
15                  nilaiVelocity[i] = batasMinEpsilon;
16                  hasil[id][i] = nilaiVelocity[i];
17              } else {
18                  if (nilaiVelocity[i] > batasMaxEpsilon) {
19                      nilaiVelocity[i] = batasMaxEpsilon;
20                      hasil[id][i] = nilaiVelocity[i];
21                  } else {
22                      nilaiVelocity[i] = nilaiVelocity[i];
23                      hasil[id][i] = nilaiVelocity[i];
24                  }}}
          return hasil;}
```

Kode Program 5.2 Fungsi Perhitungan Velocity Epsilon

Penjelasan kode program 5.2 di atas adalah sebagai berikut:

1. Baris 1-4 merupakan inisialisasi nama method untuk fungsi perhitungan *velocity epsilon*, dimana pada method ini terdapat 16 variabel parameter.
2. Baris 4-8 merupakan inisialisai variabel yang digunakan dalam method.
3. Baris 9-23 merupakan fungsi perulangan untuk menghitung *nilai velocity epsilon*.
4. Baris 24 merupakan fungsi pengembalian nilai method.

c. Fungsi Perhitungan untuk Nilai Velocity Sigma

```
1   public double[][] velocitySigma(double chi, double w,
2   double C1, double C2, double r1, double r2, double
3   particle[][], double batasMinSigma,double batasMaxSigma,
4   int id, int jumlahParticle, int tmax, double
5   velocity[][], double pBest[][], double gBest[][], int
6   iterasi) {
7           double[][] hasil = new
8   double[tmax][jumlahParticle];
9           double nilaiVelocity[] = new
10  double[jumlahParticle];
11          for (int i = 0; i < jumlahParticle; i++) {
12              nilaiVelocity[i] = chi * w * velocity[id -
13  1][i] + (C1 * r1 * (pBest[i][2] - particle[i][2]) + C2 *
14  r2 * (gBest[iterasi][2] - particle[i][2]));
15              if (nilaiVelocity[i] < batasMinSigma) {
```

```
16                  nilaiVelocity[i] = batasMinSigma;
17                  hasil[id][i] = nilaiVelocity[i];
18              } else {
19                  if (nilaiVelocity[i] > batasMaxSigma) {
20                      nilaiVelocity[i] = batasMaxSigma;
21                      hasil[id][i] = nilaiVelocity[i];
22                  } else {
23                      nilaiVelocity[i] = nilaiVelocity[i];
24                      hasil[id][i] = nilaiVelocity[i];
25                  } }}
26  return hasil;}
```

Kode Program 5.3 Fungsi Perhitungan Velocity Sigma

Penjelasan kode program 5.3 di atas adalah sebagai berikut :

1. Baris 1-6 merupakan inisialisasi nama method untuk fungsi perhitungan velocity sigma dengan 16 variabel parameter di dalamnya.
2. Baris 7-10 merupakan inisialisasi variabel yang digunakan dalam method.
3. Baris 11-25 merupakan fungsi perulangan untuk menghitung nilai *velocity sigma*.
4. Baris 26 merupakan fungsi pengembalian nilai.

d. Fungsi Perhitungan untuk Nilai xPosition Kompleksitas

```
 1  public double[][] xPositionKompleksitas(double
 2  particleAwal[][], double chi, double w, double
 3  velocityKompleksitas[][], double MinC, double MaxC, int
 4  id, int jumlahParticle, int tmax) {
 5          double[][] hasil = new
 6  double[tmax][jumlahParticle];
 7          double[] nilaiXPosition = new
 8  double[jumlahParticle];
 9          for (int j = 0; j < jumlahParticle; j++) {
10              nilaiXPosition[j] = particleAwal[j][0] +
11  (chi * w * velocityKompleksitas[id][j]);
12              if (nilaiXPosition[j] < MinC) {
13                  nilaiXPosition[j] = MinC;
14                  hasil[id][j] = nilaiXPosition[j];
15              } else {
16                  if (nilaiXPosition[j] > MaxC) {
17                      nilaiXPosition[j] = MaxC;
18
19                      hasil[id][j] = nilaiXPosition[j];
20                  } else {
21                      nilaiXPosition[j] =
22  nilaiXPosition[j];
23                      hasil[id][j] = nilaiXPosition[j];
24                  } } }
25          return hasil;   }
```

Kode Program 5.4 Fungsi Perhitungan xPosition Kompleksitas

Penjelasan kode program 5.4 di atas adalah sebagai berikut :

1. Baris 1-4 merupakan inisialisasi nama method untuk perhitungan xPosition Kompleksitas, dengan 9 variabel parameter.
2. Baris 5-8 merupakan inisialisasi variabel yang digunakan dalam method.
3. Baris 9-24 nerupakan fungsi perulangan untuk menghitung nilai xPosition Kompleksitas
4. Baris 25 merupakan fungsi pengembalian nilai method.

e. Fungsi Perhitungan untuk Nilai xPosition Epsilon

```
1  public double[][] xPositionEpsilon(double
2  particleAwal[][], double chi, double w, double
3  velocityEpsilon[][], double minEpsilon, double
4  maxEpsilon, int id, int jumlahParticle, int tmax) {
5          double[][] hasil = new
6  double[tmax][jumlahParticle];
7          double[] nilaiXPosition = new
8  double[jumlahParticle];
9          for (int j = 0; j < jumlahParticle; j++) {
10             nilaiXPosition[j] = particleAwal[j][1] +
11 (chi * w * velocityEpsilon[id][j]);
12             if (nilaiXPosition[j] < minEpsilon) {
13                 nilaiXPosition[j] = minEpsilon;
14                 hasil[id][j] = nilaiXPosition[j];
15             } else {
16                 if (nilaiXPosition[j] > maxEpsilon) {
17                     nilaiXPosition[j] = maxEpsilon;
18
19                     hasil[id][j] = nilaiXPosition[j];
20                 } else {
21                     nilaiXPosition[j] =
22 nilaiXPosition[j];
23                     hasil[id][j] = nilaiXPosition[j];
24                 }}}
25         return hasil;}
```

Kode Program 5.5 Fungsi Perhitungan Xposition Epsilon

Penjelasan kode program 5.5 di atas adalah sebagai berikut :

1. Baris 1-4 merupakan inisialisasi nama method,dengan 9 variabel parameter di dalamnya
2. Baris 5-8 merupakan inisialisasi variabel yang digunakan dalam method
3. Baris 9-24 merupakan fungsi perulangan untuk menghitung nilai xPosition Epsilon

f. Fungsi Perhitungan untuk Nilai xPosition Sigma

```
1  public double[][] xPositionSigma(double
2  particleAwal[][], double chi, double w, double
3
```

```
4   velocitySigma[][], double minSigma, double maxSigma, int
5   id, int jumlahParticle, int tmax) {
6           double[][] hasil = new
7   double[tmax][jumlahParticle];
8           double[] nilaiXPosition = new
9   double[jumlahParticle];
10          for (int j = 0; j < jumlahParticle; j++) {
11              nilaiXPosition[j] = particleAwal[j][2] +
12  (chi * w * velocitySigma[id][j]);
13              if (nilaiXPosition[j] < minSigma) {
14                  nilaiXPosition[j] = minSigma;
15                  hasil[id][j] = nilaiXPosition[j];
16              } else {
17                  if (nilaiXPosition[j] > maxSigma) {
18                      nilaiXPosition[j] = maxSigma;
19
20                      hasil[id][j] = nilaiXPosition[j];
21                  } else {
22                      nilaiXPosition[j] =
23  nilaiXPosition[j];
24                      hasil[id][j] = nilaiXPosition[j];
25                  }} }
            return hasil; }
```

Kode Program 5.6 Fungsi Perhitungan xPosition Sigma

Penjelasan kode program 5.6 di atas adalah sebagai berikut :

1. Baris 1-4 merupakan inisialisasi nama method dengan 9 variabel parameter di dalamnya
2. Baris 5-8 merupakan inisialisasi variabel yang digunakan dalam method
3. Baris 9-24 merupakan fungsi perulangan untuk menghitung nilai xPosition Sigma

g. Fungsi Perhitungan Nilai W

```
1   double w[] = new double[tmax];
2           for (int j = 0; j < tmax; j++) {
3               w[j] = wmin + (wmax - wmin) * (tmax - (j + 1))
4   / tmax;
```

Kode Program 5.7 Fungsi Perhitungan Nilai W

Penjelasan kode program 5.7 di atas adalah sebagai berikut :

1. Baris 1 merupakan inisialisasi nama variabel untuk perhitungan w
2. Baris 2-4 merupakan fungsi perulangan untuk menghitung nilai w

h. Fungsi Perhitungan Nilai Gbest

```
1    maxFitness = fitness[x][0];
2                   double tampung = 0.0;
3                   idMax = 0;
4                   for (int k = 0; k < jumlahParticle;
5   k++) {
```

```
6                           if (maxFitness < fitness[x][k])
7   {
8                               tampung = fitness[x][k];
9                               maxFitness = tampung;
10                              idMax = k;
11                          }
12                      }
13  for (int j = 0; j < jumlahParameter; j++) {
14                          gBest[x][j] = pBest[idMax][j];
15                      }
16
17                      FitnessFinal[x] = maxFitness;
18                      idFitnessfinal[x] = idMax;
19                  }
20              }
21              double maxFitnessFinal = FitnessFinal[0];
22              int idGenerasi = 0;
23              double tempNilai = 0.0;
24              for (int i = 0; i < iterasiIPSO; i++) {
25                  if (maxFitnessFinal < FitnessFinal[i]) {
26                      tempNilai = FitnessFinal[i];
27                      maxFitnessFinal = tempNilai;
28                      idGenerasi = i;
29                  } else {
30                      maxFitnessFinal = maxFitnessFinal;
31                      idGenerasi = idGenerasi;
32                  }
33              }
```

Kode Program 5.8 Fungsi Perhitungan Gbest

Penjelasan kode program 5.7 di atas adalah sebagai berikut :

1. Baris 1-3 merupakan inisialisasi nama variabel yang digunakan dalam method
2. Baris 4-12 merupakan perulangan untuk mencari nilai gbestnya dengan ditampung terlebih dulu
3. Baris 13-33 merupakan perulangan untuk menentukan yang mana nilai gbestnya

i. Fungsi Perhitungan Nilai Pbest

```
1   for (int j = 0; j < jumlahParticle; j++) {
2                           if (fitness[x - 1][j] >
3   fitness[x][j]) {
4                               for (int k = 0; k <
5   jumlahParameter; k++) {
6                                   pBestBaru[j][k] =
7   particle[j][k];
8                               }
9                               pBestBaru[j][3] = fitness[x
10  - 1][j];
11                          } else {
12
```

```
13                                  pBestBaru[j][0] =
14  xPositionKompleksitas[x][j];
15                                  pBestBaru[j][1] =
16  xPositionEpsilon[x][j];
17                                  pBestBaru[j][2] =
18  xPositionSigma[x][j];
19                                  pBestBaru[j][3] =
20  fitness[x][j];
21                              }
                            }
```

Kode Program 5.9 Fungsi Perhitungan Pbest

Penjelasan kode program 5.7 di atas adalah sebagai berikut :

1. Baris 1-10 merupakan fungsi perulangan untuk menghitung nilai pbest baru
2. Baris 11-21 merupakan fungsi perulangan selain kondisi awal dengan kondisi bahwa nilai pbest baru yang dihasilkan akan menjadi nilai untuk xPosition dari kompleksitas,epsilon,sigma serta akan menjadi nilai fitness

5.2.2 Implementasi Algoritma Support Vector Regression

Implementasi algoritma Support Vector Regression meliputi di bawah ini, yaitu

a. Fungsi perhitungan Jarak Data Training

```
1   public double[][] jarakDataTraining(double[][] data, int
2   pjgDt, int fitur) {
3           double[][] dataJarak = new double[pjgDt][pjgDt];
4           double[][] result = new double[pjgDt][pjgDt];
5           double tempJarak, hasil;
6           for (int x = 0; x < pjgDt; x++) {
7               for (int y = 0; y < pjgDt; y++) {
8                   tempJarak = 0.0;
9                   hasil = 0.0;
10                  for (int k = 0; k < fitur - 1; k++) {
11                      tempJarak = Math.pow((data[y][k] -
12  data[x][k]), 2);
13                      hasil += tempJarak;
14                      dataJarak[x][y] = hasil;
15                      result[x][y] = dataJarak[x][y];
16                  }}}
17          return result; }
```

Kode Program 5.10 Fungsi Perhitungan Jarak Data Training

Penjelasan kode program 5.10 di atas adalah sebagai berikut :

1. Baris 1-2 merupakan inisialisasi nama method dengan 3 variabel parameter di dalamnya
2. Baris 3-5 merupakan inisialisasi nama variabel yang digunakan dalam method

3. Baris 6-16 merupakan fungsi perulangan untuk menghitung seluruh jarak data training
4. Baris 17 merupakan fungsi pengembalian nilai

b. Fungsi Perhitungan Jarak Data Testing

```
1  public double[][] jarakDataTesting(double[][]
2  dataTraining, double[][] dataTesting, int pjgDt, int
3  fitur, int pjgTRain) {
4          double[][] dataJarak = new
5  double[pjgDt][pjgTRain];
6          double[][] result = new double[pjgDt][pjgTRain];
7          double tempJarak, hasil;
8          for (int x = 0; x < pjgDt; x++) {
9              for (int y = 0; y < pjgTRain; y++) {
10                 tempJarak = 0.0;
11                 hasil = 0.0;
12                 for (int k = 0; k < fitur - 1; k++) {
13                     tempJarak =
14 Math.pow((dataTesting[x][k] - dataTraining[y][k]), 2);
15                     hasil += tempJarak;
16                     dataJarak[x][y] = hasil;
17                     result[x][y] = dataJarak[x][y];
18                 } } }
19         return result;}
```

Kode Program 5.11 Fungsi Perhitungan Jarak Data Testing

Penjelasan kode program 5.11 di atas adalah sebagai berikut :

1. Baris 1-3 merupakan inisialisasi nama method dengan 5 variabel parameter di dalamnya
2. Baris 4-7 merupakan inisialisasi variabel yang digunakan dalam method
3. Baris 8-18 merupakan fungsi perulangan untuk menghitung jarak data testing dengan data training

c. Fungsi Perhitungan Kernel Data Training

```
1  public double[][] kernelTraining(double[][] dataJarak,
2  int pjgDt, double sigma) {
3          double[][] result = new double[pjgDt][pjgDt];
4          double[][] kernel = new double[pjgDt][pjgDt];
5          for (int a = 0; a < pjgDt; a++) {
6              for (int b = 0; b < pjgDt; b++) {
7                  kernel[a][b] = Math.exp(-
8  ((dataJarak[a][b]) / (2 * (Math.pow(sigma, 2)))));
9                  result[a][b] = kernel[a][b];
10             }}  return result;
11     }
```

Kode Program 5.12 Fungsi perhitungan Kernel Data Training

Penjelasan kode program 5.12 di atas adalah sebagai berikut :

1. Baris 1-2 merupakan inisialisasi nama method dengam 3 variabel parameter di dalamnya
2. Baris 3-4 merupakan inisialisasi variabel yang digunakan dalam method
3. Baris 5-9 adalah fungsi perulangan untuk menh=ghtiung nilai kernel data training
4. Baris 10-11 merupakan fungsi pengembalian nilai

d. Fungsi Perhitungan Kernel Data Testing

```
1  public double[][] kernelTesting(double[][] dataJarak,
2  int pjgDt, double sigma, int pjgTrain) {
3          double[][] result = new
4  double[pjgDt][pjgTrain];
5          double[][] kernel = new
6  double[pjgDt][pjgTrain];
7          for (int a = 0; a < pjgDt; a++) {
8              for (int b = 0; b < pjgTrain; b++) {
9                  kernel[a][b] = Math.exp(-
10 ((dataJarak[a][b]) / (2 * (Math.pow(sigma, 2)))));
11                 result[a][b] = kernel[a][b];
12             }}return result;
13     }
```

Kode Program 5.13 Fungsi Perhitungan Kernel Data Testing

Penjelasan kode program 5.13 di atas adalah sebagai berikut :

1. Baris 1-3 merupakan inisialisasi nama method dengan 3 variabel parameter di dalamnya
2. Baris 4-7 merupakan inisialisasi variabel yang digunakan dalam method
3. Baris 8-12 merupakan fungsi perulangan untuk menghitung nilai kernel data testing
4. Baris 13 merupakan fungsi pengembalian nilai

e. Fungsi Perhitungan Rij Data Training

```
1  public double[][] rij(double[][] kernel, int pjgDt,
2  double lamdha) {
3          double[][] result = new double[pjgDt][pjgDt];
4          double[][] Rij = new double[pjgDt][pjgDt];
5          for (int c = 0; c < pjgDt; c++) {
6              for (int d = 0; d < pjgDt; d++) {
7                  Rij[c][d] = kernel[c][d] +
8  (Math.pow(lamdha, 2));
9                  result[c][d] = Rij[c][d];
10             }} return result;
11     }
```

Kode Program 5.14 Fungsi Perhitungan *Rij* Data Training

Penjelasan kode program 5.14 di atas adalah sebagai berikut :

1. Baris 1-2 merupakan inisialisasi nama method dengan 3 variabel parameter di dalamnya
2. Baris 3-4 merupakan inisialisasi variabel yang akan digunakan dalam method
3. Baris 5-9 merupakan fungsi perulangan untuk menghitung nilai *matriks Rij*
4. Baris 10-11 merupakan fungsi pengembalian nilai result method

f. Fungsi Perhitungan Rij Data Testing

```
1  public double[][] rijTesting(double[][] kernel, int
2  pjgDt, double lamdha, int pjgTrain) {
3          double[][] result = new
4  double[pjgDt][pjgTrain];
5          double[][] Rij = new double[pjgDt][pjgTrain];
6          for (int c = 0; c < pjgDt; c++) {
7              for (int d = 0; d < pjgTrain; d++) {
8                  Rij[c][d] = kernel[c][d] +
9  (Math.pow(lamdha, 2));
10                 result[c][d] = Rij[c][d];
11             }}
12 return result; }
```

Kode Program 5.15 Fungsi Perhitungan *Rij* Data Testing

Penjelasan kode program 5.15 di atas adalah sebagai berikut :

1. Baris 1-2 merupakan inisialisasi nama method dengan 4 variabel parameter di dalamnya
2. Baris 3-5 merupakan inisialisasi nama variabel yang digunakan dalam method
3. Baris 6-11 merupakan fungsi perulangan untuk menghitung nilai *Matriks Rij* Data testing
4. Baris 12 merupakan fungsi kembalian nilai result method

g. Fungsi Perhitungan Nilai Gamma (γ)

```
1  public double gamma(double[][] rij, double cLR) {
2          double result = 0.0;
3          double gamma = cLR / rij[0][0];
4          result = gamma;
5          return result; }
```

Kode Program 5.16 Fungsi Perhitungan Nilai Gamma

Penjelasan kode program 5.16 di atas adalah sebagai berikut :

1. Baris 1 merupakan inisialisasi nama method dengan 2 variabel parameter di dalamnya
2. Baris 2 merupakan inisialisasi variabel yang digunakan dalam method
3. Baris 3-4 merupakan fungsi perhitungan nilai gamma

4. Baris 5 merupakan fungsi pengembalian nilai method

h. Fungsi Perhitungan Nilai Error

```
1  public double[] error(double[][] data, double[][] Rij,
2  double[] alphai, double[] alphaiStar, int pjgDt, int
3  fitur) {
4          double[] result = new double[pjgDt];
5          double[] E = new double[pjgDt];
6          double tempE, totalTempE;
7          for (int i = 0; i < pjgDt; i++) {
8              tempE = 0.0;
9              totalTempE = 0.0;
10             for (int j = 0; j < pjgDt; j++) {
11                 tempE = (alphaiStar[j] - alphai[j]) *
12 Rij[j][i];
13                 totalTempE += tempE;
14             }
15             E[i] = (data[i][fitur - 1]) - totalTempE;
16             result[i] = E[i];
17         }
18         return result; }
```

Kode Program 5.17 Fungsi Perhitungan Nilai Error

Penjelasan kode program 5.17 di atas adalah sebagai berikut :

1. Baris 1-3 merupakan inisialisasi nama method dengan memiliki 6 variabel parameter di dalamnya
2. Baris 4-6 merupakan inisialisasi variabel yang digunakan dalam method
3. Baris 7-17 merupakan fungsi perulangan untuk menghitung nilai *Error*
4. Baris 18 merupakan fungsi pengembalian nilai method

i. Fungsi Perhitungan Deltha Alphai Star ($\delta\alpha_i^*$)

```
1  public double[] deltaAlphaiStar(double gama, int pjgDt,
2  double eps, double C, double[] E, double[] alphai,
3  double[] alphaiStar) {
4          double[] result = new double[pjgDt];
5          double[] deltaAlphaiStar = new double[pjgDt];
6          for (int i = 0; i < pjgDt; i++) {
7              deltaAlphaiStar[i] =
8  Math.min(Math.max((gama * (E[i] - eps)), (-
9  alphaiStar[i])), (C - (alphaiStar[i])));
10             result[i] = deltaAlphaiStar[i];}
11         return result;}
```

Kode Program 5.18 Fungsi Perhitungan *Deltha Alpha i Star*

Penjelasan kode program 5.18 di atas adalah sebagai berikut :

1. Baris 1-3 merupakan inisialisasi nama method dengan memiliki 7 variabel parameter di dalamnya
2. Baris 4-5 merupakan inisialisasi variabel yang digunakan dalam method

3. Baris 6-10 merupakan fungsi perulangan untuk perhitungan nilai Deltha Alphai Star
4. Baris 11 merupakan fungsi pengembalian nilai method

j. Fungsi Perhitungan Deltha Alphai ($\delta\alpha_i$)

```
1   public double[] deltaAlphai(double gama, int pjgDt,
2   double eps, double C, double[] E, double[] alphai,
3   double[] alphaiStar) {
4           double[] result = new double[pjgDt];
5           double[] deltaAlphai = new double[pjgDt];
6           for (int i = 0; i < pjgDt; i++) {
7               deltaAlphai[i] = Math.min(Math.max((gama *
8   (-E[i] - eps)), -alphai[i]), (C - (alphai[i])));
9               result[i] = deltaAlphai[i];
10          }
11          return result;
12      }
13
```

Kode Program 5.19 Fungsi Perhitungan *Deltha Alpha i*

Penjelasan kode program 5.19 di atas adalah sebagai berikut :

1. Baris 1-3 merupakan inisialisasi nama method dengan 7 variabel parameter di dalamnya
2. Baris 4-5 merupakan inisialisasi variabel yang digunakan dalam method
3. Baris 6-10 merupakan fungsi perulangan untuk menghitung nilai deltha alpha i
4. Baris 12-13 merupakan fungsi pengembalian nilai method

k. Fungsi Perhitungan Nilai Alphai

```
1   public double[] alphaI(int pjgDt, double[] alphai,
2   double[] deltaAlphai) {
3           double[] result = new double[pjgDt];
4           for (int i = 0; i < pjgDt; i++) {
5               alphai[i] = deltaAlphai[i] + alphai[i];
6               result[i] = alphai[i];
7           }
8           return result;
9       }
```

Kode Program 5.20 Fungsi Perhitungan Alpha i

Penjelasan kode program 5.20 di atas adalah sebagai berikut :

1. Baris 1-2 merupakan inisialisasi nama method dengan 3 variabel parameter di dalamnya
2. Baris 3 merupakan inisialisasi variabel yang akan digunakan dalam method
3. Baris 4-7 merupakan fungsi perulangan untuk menghitung nilai alpha i
4. Baris 8-9 merupakan kembalian nilai result method

l. Fungsi Perhitungan Nilai Alphai Star

```
1  public double[] alphaIStar(int pjgDt, double[]
2  alphaiStar, double[] deltaAlphaiStar) {
3          double[] result = new double[pjgDt];
4          for (int i = 0; i < pjgDt; i++) {
5              alphaiStar[i] = deltaAlphaiStar[i] +
6  alphaiStar[i];
7              result[i] = alphaiStar[i];
8          }
9          return result;}
```

Kode Program 5.21 Fungsi Perhitungan Alpha i Star

Penjelasan kode program 5.21 di atas adalah sebagai berikut :

1. Baris 1-2 merupakan inisialisasi nama method dengan 3 variabel parameter di dalamnya
2. Baris 3 merupakan inisialisasi variabel yang digunakan dalam method
3. Baris 4-8 merupakan fungsi perulangan untuk menghitung nilai aplhai star
4. Baris 9 merupakan kembalian nilai result method

m. Fungsi Perhitungan Fx Data Training

```
1  public double[] fx(int pjgDt, double[] alphaiStar,
2  double[] alphai, double[][] rij) {
3          double[] result = new double[pjgDt];
4          double[] Fx = new double[pjgDt];
5          double tempFx, totalTempFx;
6          for (int i = 0; i < pjgDt; i++) {
7              tempFx = 0.0;
8              totalTempFx = 0.0;
9              for (int j = 0; j < pjgDt; j++) {
10                 tempFx = (alphaiStar[j] - alphai[j]) *
11 (rij[i][j]);
12                 totalTempFx += tempFx;
13                 Fx[i] = totalTempFx;
14                 result[i] = Fx[i];
15             }}return result;
16     }
```

Kode Program 5.22 Fungsi Perhitungan Fx Data Training

Penjelasan kode program 5.22 di atas adalah sebagai berikut :

1. Baris 1-2 merupakan inisialisasi nama method, dengan 4 variabel parameter di dalamnya
2. Baris 3-5 merupakan inisialisasi variabel yang digunakan dalam method
3. Baris 6-14 merupakan fungsi perulangan untuk menghitung nilai *Fx* pada data training
4. Baris 15-16 merupakan kembalian nilai result method

n. Fungsi Perhitungan Fx Data Testing

```
1   public double[] fxTesting(int pjgDt, double[]
2   alphaiStar, double[] alphai, double[][] rij, int
3   pjgDtTraining) {
4           double[] result = new double[pjgDt];
5           double[] Fx = new double[pjgDt];
6           double tempFx, totalTempFx;
7           for (int i = 0; i < pjgDt; i++) {
8               tempFx = 0.0;
9               totalTempFx = 0.0;
10              for (int j = 0; j < pjgDtTraining; j++) {
11                  tempFx = (alphaiStar[j] - alphai[j]) *
12  (rij[i][j]);
13                  totalTempFx += tempFx;
14                  Fx[i] = totalTempFx;
15                  result[i] = Fx[i];
16              }
17          }
18          return result;
19      }
```

Kode Program 5.23 Fungsi Perhitungan Fx Data Testing

Penjelasan kode program 5.23 di atas adalah sebagai berikut :

1. Baris 1-3 merupakan inisialisasi nama method dengan 5 variabel parameter di dalamnya
2. Baris 4-6 merupakan inisialisasi nama variabel yang akan digunakan dalam method
3. Baris 7-15 merupakan fungsi perulangan untuk menghitung nilai *Fx* data testing

o. Fungsi Perhitungan Fitness Training

```
1   public double fitness(double[][] data, double[] fx, int
2   panjangData, int fiturData) {
3           double[][] dataAktual = new
4   double[panjangData][fiturData];
5           double[] dataPeramalan = new
6   double[panjangData];
7           double[] tampungHasil = new
8   double[panjangData];
9           double totalTampungHasil = 0.0;
10          double banyakData = panjangData;
11          double MAPE, fitness, hasil;
12
13          dataPeramalan = fx;
14          dataAktual = data;
15          for (int i = 0; i < panjangData; i++) {
16              tampungHasil[i] =
17  Math.abs(((dataPeramalan[i] - dataAktual[i][fiturData -
18  1]) / dataPeramalan[i]) * 100);
19  //              System.out.println("
20  "+tampungHasil[i]);
```

```
21              totalTampungHasil += tampungHasil[i];
22          }
23          MAPE = totalTampungHasil / banyakData;
24  System.out.println("==================================
25  ========");
26          hasil = fitness;
27          return hasil;}
```

Kode Program 5.24 Perhitungan Fitness Training

Penjelasan kode program 5.24 di atas adalah sebagai berikut :

1. Baris 1-2 merupakan inisialisasi nama method dengan memiliki 3 variabel parameter di dalamnya
2. Baris 3-14 merupakan inisialisasi variabel yang digunakan dalam method
3. Baris 15-26 merupakan fungsi perulangan untuk perhitungan nilai fitness
4. Baris 27 merupakan fungsi kembalian nilai result method

p. Fungsi Perhitungan Fitness Testing

```
1   public double fitnessTesting(double[][] data, double[]
2   fx, int panjangData, int fiturData) {
3           double[][] dataAktual = new
4   double[panjangData][fiturData];
5           double[] dataPeramalan = new
6   double[panjangData];
7           double[] tampungHasil = new
8   double[panjangData];
9           double totalTampungHasil = 0.0;
10          double banyakData = panjangData;
11          double MAPE, fitness, hasil;
12
13          dataPeramalan = fx;
14          dataAktual = data;
15          for (int i = 0; i < panjangData; i++) {
16              tampungHasil[i] =
17  Math.abs(((dataPeramalan[i] - dataAktual[i][fiturData -
18  1]) / dataPeramalan[i]) * 100);
19  //              System.out.println("
20  "+tampungHasil[i]);
21              totalTampungHasil += tampungHasil[i];
22          }
23          MAPE = totalTampungHasil / banyakData;
24          System.out.println("MAPE TESTING    : " +
25  MAPE);
26          fitness = 1 / (1 + MAPE);
27          hasil = fitness;
28          return hasil;}
```

Kode Program 5.25 Perhitungan Fitness Testing

Penjelasan kode program 5.25 di atas adalah sebagai berikut :

1. Baris 1-2 merupakan inisialisasi nama method dengan memiliki 4 variabel parameter di dalamnya
2. Baris 3-14 merupakan inisialisasi nama variabel yang digunakan dalam method
3. Baris 15-27 merupakan fungsi perulangan perhitungan nilai *fitness*

q. Fungsi Perhitungan Normalisasi Data

```
1  public double[][] normalisasiData(double[][] data,
2  double maxData, double minData, int tabulasi, int
3  fitur) {
4          double[][] result = new
5  double[tabulasi][fitur];
6          double range = maxData - minData;
7          for (int i = 0; i < tabulasi; i++) {
8              for (int j = 0; j < fitur; j++) {
9                  result[i][j] = (data[i][j] - minData) /
10 range;
11             }}return result;
12   }
```

Kode Program 5.26 Perhitungan Normalisasi Data

Penjelasan kode program 5.26 di atas adalah sebagai berikut :

1. Baris 1-3 merupakan inisialisasi nama method dengan 4 variabel parameter di dalamnya
2. Baris 4-6 merupakan inisialisasi nama variabel yang digunakan dalam method
3. Baris 7-10 adalah fungsi perulangan untuk perhitungan nilai normalisasi data

r. Fungsi Perhitungan Denormalisasi Data

```
1  public double[][] deNormalisasiData(double[][]
2  dataNormalisasi, double maxData, double minData, int
3  tabulasi, int fitur) {
4          double[][] result = new
5  double[tabulasi][fitur];
6          double range = maxData - minData;
7          for (int i = 0; i < tabulasi; i++) {
8              for (int j = 0; j < fitur; j++) {
9                  result[i][j] = (dataNormalisasi[i][j] *
10 range) + minData;
11             }
12         }
13         return result;
14     }
```

Kode Program 5.27 Perhitungan Denormalisasi Data

Penjelasan kode program 5.27 di atas adalah sebagai berikut :

1. Baris 1-3 ialah inisialisasi nama method dengan 5 variabel parameter di dalamnya
2. Baris 4-6 ialah inisialisasi variabel yang akan digunakan dalam method
3. Baris 7-12 merupakan fungsi perulangan untuk perhitungan denormalisasi data

s. Fungsi Perhitungan Denormalisasi Fx

```
1  public double[] deNormalisasiFx(double[] Fx, double
2  maxData, double minData) {
3          double[] result = new double[Fx.length];
4          double range = maxData - minData;
5          for (int i = 0; i < Fx.length; i++) {
6              result[i] = (Fx[i] * range) + minData;
7          }
8          return result;}
```

Kode Program 5.28 Perhitungan Denormalisasi Fx

Penjelasan kode program 5.28 di atas adalah sebagai berikut :

1. Baris 1-2 : inisialiasi nama method dengan 3 parameter di dalamnya
2. Baris 3-4 : inisialisasi variabel yang digunakan dalam method
3. Baris 5-7 : fungsi perulangan untuk menghitung denormalisasi nilai Fx (denormalisasi nilai regresi)
4. Baris 8 merupakan fungsi kembalian nilai method

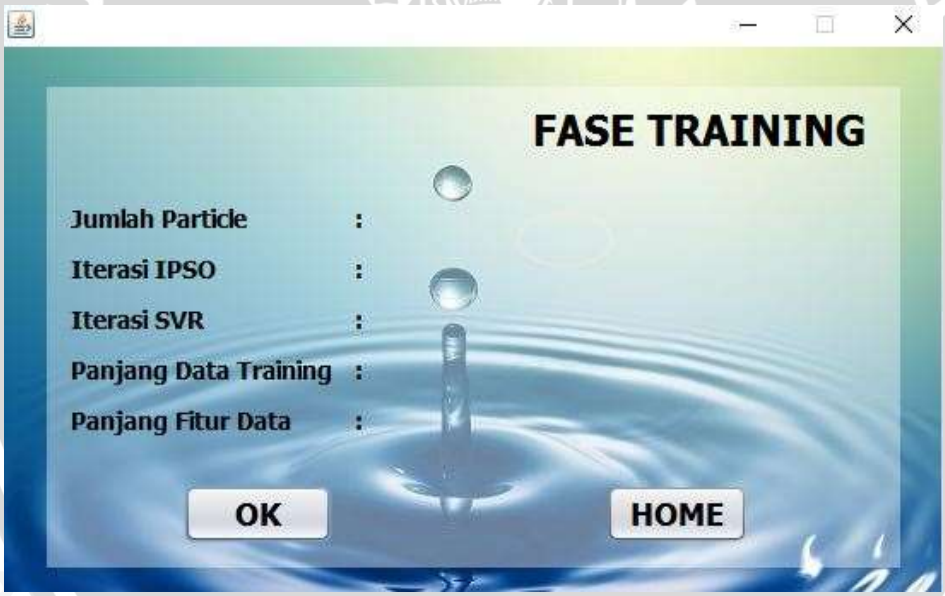
5.3 Implementasi User Interface

Pada implementasi *user interface,* aplikasi ini dibuat 5 halaman yaitu halaman utama sistem, halaman fase training, halaman fase testing, halaman result training, halaman result testing.

5.3.1 Implementasi Home

Gambar 5.1 Halaman Utama Sistem

5.3.2 Implementasi Fase Training

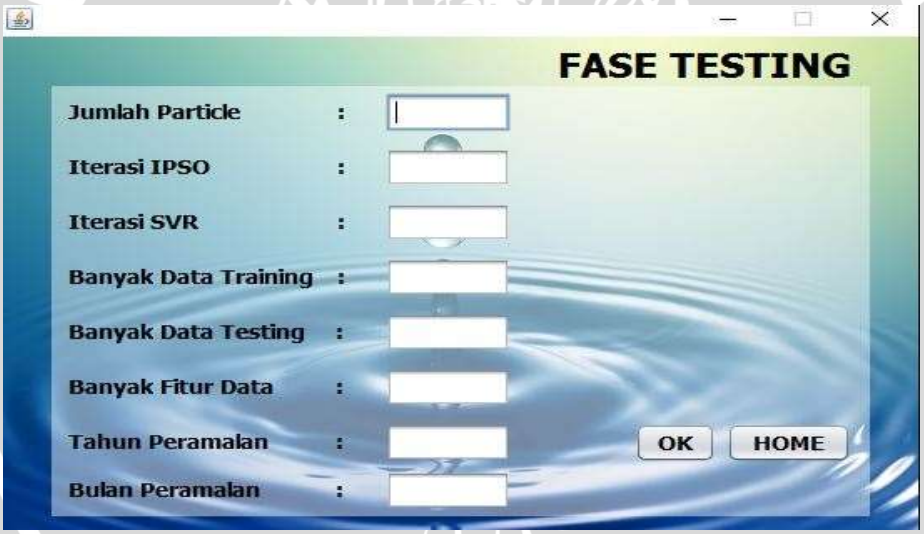


Gambar 5.2 Halaman Fase Training

5.3.3 Implementasi Result Training

Gambar 5.3 Halaman Result Training

5.3.4 Implementasi Fase Testing



Gambar 5.4 Halaman Fase Testing

5.3.5 Implementasi Result Testing

Gambar 5.5 Halaman Result Testing

BAB 6 PENGUJIAN DAN ANALISIS

Pada bab ini membahas mengenai tahapan pengujian dan analisis dari implementasi sistem algoritma *Support Vector Regression* dan *Improved Particle Swarm Optimization*.

6.1 Hasil dan Analisa Uji Coba Jumlah Iterasi IPSO

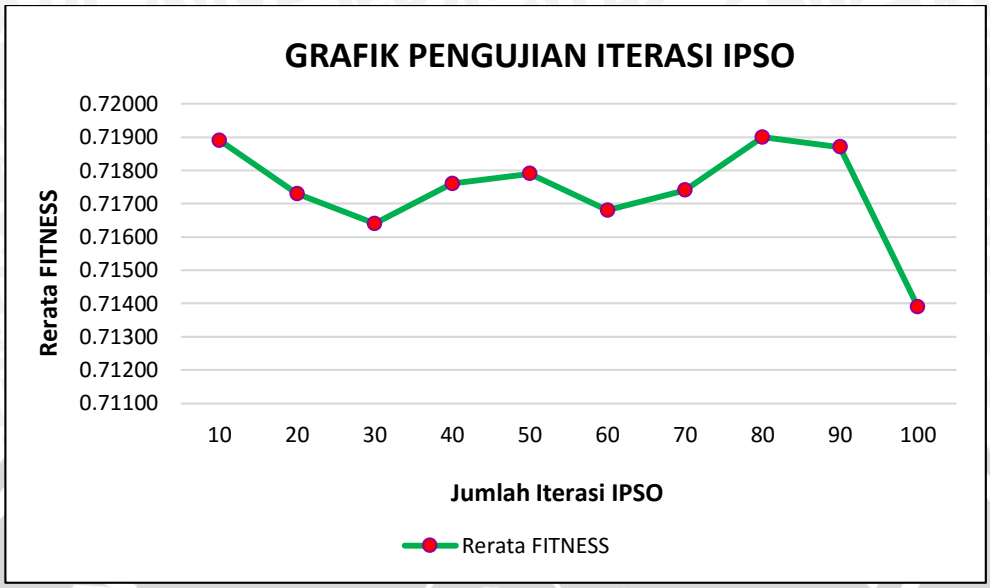
Pada uji coba yang pertama yaitu uji coba Jumlah Iterasi IPSO. Data yang digunakan sebagai uji coba ini adalah data konsumsi air di PDAM Malang pada tahun 2008 – 2012 yang dibuat dalam bentuk data bulanan untuk meramalkan 31 bulan kedepannya. Dalam uji coba ini jumlah iterasi IPSO maksimal yang digunakan yaitu sebesar 100 dengan iterasi maksimal SVR sebesar 10000 dengan jumlah partikel sebanyak 10. Dalam uji coba pertama ini nilai fitness terbaik yang diperoleh adalah sebesar 0.72 dengan jumlah iterasi sebanyak 80 kali. Berikut Hasil uji coba Jumlah Iterasi IPSO dapat diamati pada Tabel 6.1 di bawah ini :

Tabel 6.1 Hasil Uji Coba Jumlah Iterasi IPSO

Jumlah Iterasi IPSO	Nilai Fitness Percobaan ke –*i*										Rerata FITNESS
	1	2	3	4	5	6	7	8	9	10	
10	0.71800	0.71700	0.72000	0.71900	0.71800	0.71900	0.71900	0.72000	0.71900	0.72000	0.71890
20	0.71900	0.71900	0.71800	0.72000	0.71000	0.72000	0.71900	0.71700	0.71200	0.71900	0.71730
30	0.72000	0.71500	0.71700	0.71500	0.71600	0.71900	0.71400	0.71900	0.71500	0.71400	0.71640
40	0.71800	0.71900	0.71400	0.72000	0.71800	0.71400	0.71700	0.71900	0.71900	0.71800	0.71760
50	0.71900	0.71900	0.72000	0.71900	0.71900	0.71900	0.71100	0.72000	0.71400	0.71900	0.71790
60	0.71900	0.72000	0.71600	0.71900	0.71900	0.71900	0.70800	0.71500	0.71400	0.71900	0.71680
70	0.71800	0.71900	0.72000	0.71900	0.71900	0.71900	0.71900	0.70400	0.72000	0.71700	0.71740
80	0.72000	0.71900	0.71900	0.72000	0.71900	0.71800	0.71700	0.72000	0.71900	0.71900	0.71900
90	0.71600	0.71900	0.72000	0.72000	0.71800	0.71700	0.71900	0.71900	0.71900	0.72000	0.71870
100	0.72000	0.71600	0.71700	0.71900	0.71300	0.72000	0.71700	0.69300	0.70400	0.72000	0.71390

Berdasarkan hasil grafik uji coba jumlah iterasi IPSO pada Gambar 6.1 yaitu Grafik Pengujian Iterasi IPSO. Rata-rata nilai *fitness* terbesar diperoleh dari jumlah iterasi ke 80. Pada iterasi ke 80 nilai *fitness* yang dihasilkan sebesar 0.72, hasil nilai *fitness* dalam uji coba iterasi IPSO ini sangat bervariasi. Oleh karena itu, berdasarkan analisa hasil uji coba pada Gambar 6.1 Grafik Pengujian Iterasi IPSO ini dapat disimpulkan bahwa semakin banyak jumlah iterasi IPSO belum tentu semakin baik nilai *fitness* yang didapatkan. Ini disebabkan karena adanya faktor pembangkitan nilai random χ. Berikut hasil grafik rata-rata nilai *fitness* yang

dihasilkan pada uji coba jumlah iterasi IPSO dapat dilihat pada Gambar 6.1 Grafik Pengujian Iterasi IPSO :



Gambar 6.1 Grafik Pengujian Iterasi IPSO

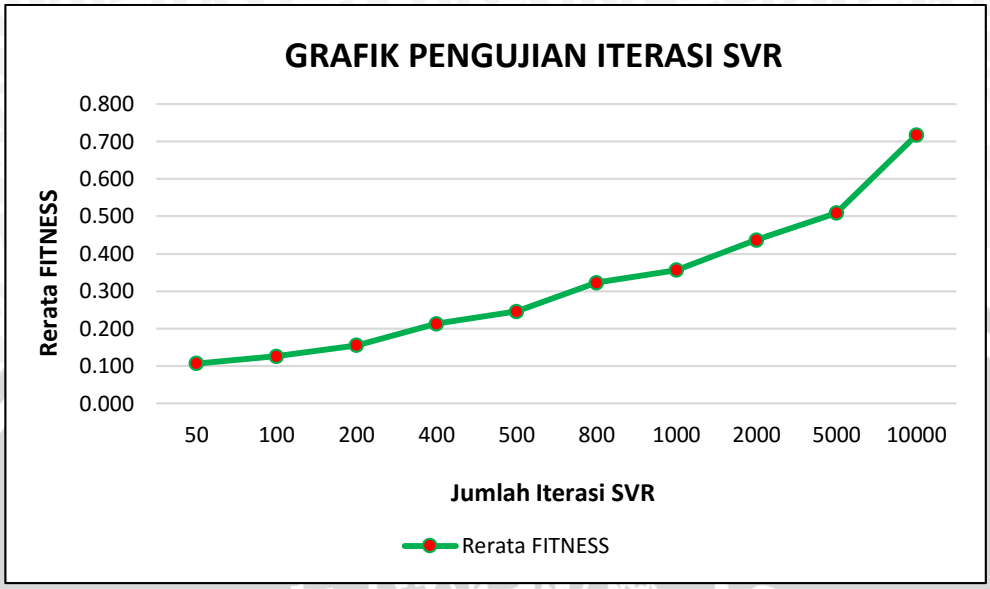
6.2 Hasil dan Analisa Uji Coba Jumlah Iterasi SVR

Pada uji coba kedua ini merupakan uji coba Jumlah Iterasi SVR. Data yang digunakan dalam uji coba ini data konsumsi air pada tahun 2008-2014. Dalam uji coba jumlah iterasi SVR ini semakin banyak jumlah iterasi semakin baik hasil peramalannya. Pada uji coba jumlah iterasi SVR ini hasil nilai *fitness* yang dihasilkan pada iterasi 10000 merupakan jumlah iterasi paling optimal. Berikut hasil Uji Coba Jumlah Iterasi SVR dapat dilihat pada Tabel 6.2 Hasil Uji Coba Jumlah Iterasi SVR :

Tabel 6.2 Hasil Uji Coba Jumlah Iterasi SVR

Jumlah Iterasi SVR	Nilai Fitness Percobaan ke-*i*										Rerata FITNESS
	1	2	3	4	5	6	7	8	9	10	
50	0.107	0.107	0.107	0.107	0.107	0.107	0.107	0.107	0.107	0.107	0.107
100	0.125	0.126	0.126	0.126	0.126	0.126	0.126	0.126	0.126	0.126	0.126
200	0.156	0.153	0.155	0.156	0.155	0.156	0.156	0.156	0.155	0.154	0.155
400	0.211	0.215	0.209	0.213	0.214	0.213	0.214	0.214	0.214	0.213	0.213
500	0.247	0.246	0.244	0.246	0.245	0.249	0.245	0.243	0.246	0.248	0.246
800	0.321	0.324	0.323	0.324	0.323	0.323	0.322	0.323	0.321	0.323	0.323
1000	0.356	0.356	0.357	0.357	0.357	0.357	0.358	0.357	0.355	0.356	0.357
2000	0.432	0.439	0.434	0.435	0.441	0.440	0.435	0.441	0.434	0.437	0.437
5000	0.530	0.527	0.465	0.457	0.532	0.522	0.467	0.528	0.530	0.532	0.509
10000	**0.716**	**0.714**	**0.718**	**0.719**	**0.720**	**0.719**	**0.719**	**0.713**	**0.715**	**0.719**	**0.717**

Berdasarkan grafik hasil uji coba jumlah iterasi SVR pada Gambar 6.2, rata-rata nilai *fitness* terbesar didapatkan pada iterasi ke 10000. Pada iterasi 10000 nilai *fitness* yang dihasilkan yaitu sebesar 0.717, hasil ini dapat menunjukkan bahwa iterasi paling optimal untuk metode *Support Vector Regression* yaitu jumlah iterasi ke 10000.



Gambar 6.2 Grafik Pengujian Iterasi SVR

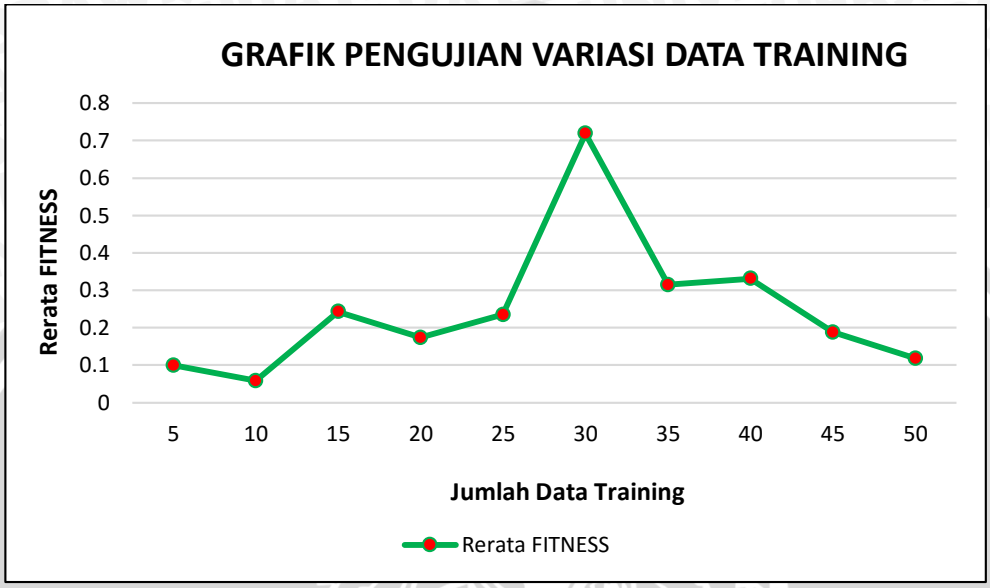
6.3 Hasil dan Analisa Uji Coba Jumlah Data Training

Pada uji coba kali ini yaitu Uji Coba Jumlah Data Training. Data yang digunakan pada uji coba ini adalah data konsumsi air pada tahun 2008-2014. Pada uji coba banyak jumlah data training, jumlah data training yang ideal adalah data training dengan jumlah 30 data training. Berikut hasil uji coba dijelaskan pada Tabel 6.3 Hasil Uji Coba Jumlah Data Training.

Tabel 6.3 Hasil Uji Coba Jumlah Data Training

Data *Training*	Nilai Fitness Percobaan ke *-i*										Rerata FITNESS
	1	2	3	4	5	6	7	8	9	10	
5	0.099	0.099	0.1	0.099	0.099	0.099	0.1	0.099	0.1	0.099	0.0993
10	0.058	0.058	0.058	0.058	0.057	0.058	0.061	0.057	0.057	0.059	0.0581
15	0.243	0.241	0.242	0.243	0.244	0.244	0.243	0.245	0.243	0.237	0.2425
20	0.174	0.173	0.174	0.173	0.171	0.174	0.173	0.174	0.174	0.174	0.1734
25	0.236	0.233	0.236	0.236	0.236	0.236	0.234	0.233	0.233	0.232	0.2345
30	0.719	0.72	0.719	0.72	0.72	0.714	0.719	0.719	0.719	0.72	0.7189
35	0.313	0.314	0.315	0.315	0.314	0.315	0.315	0.315	0.313	0.315	0.3144
40	0.331	0.331	0.332	0.331	0.331	0.331	0.331	0.331	0.331	0.331	0.3311
45	0.188	0.188	0.188	0.189	0.188	0.188	0.188	0.188	0.188	0.188	0.1881
50	0.117	0.117	0.118	0.119	0.118	0.118	0.119	0.117	0.118	0.119	0.118

Berdasarkan Gambar 6.3 Grafik Pengujian Jumlah Variasi Data Training. Hasil grafik pengujian jumlah variasi data training dapat disimpulkan bahwa jumlah data training yang ideal yaitu sebanyak 30. Jumlah data training 30 menghasilkan nilai *fitness* sebesar 0,72. Hal ini dipengaruhi dengan karakteristik data. Dapat dikatakan bahwa semakin data tersebut bervariasi maka semakin baik nilai *fitness* yang akan dihasilkan.



Gambar 6.3 Grafik Pengujian Variasi Data Training

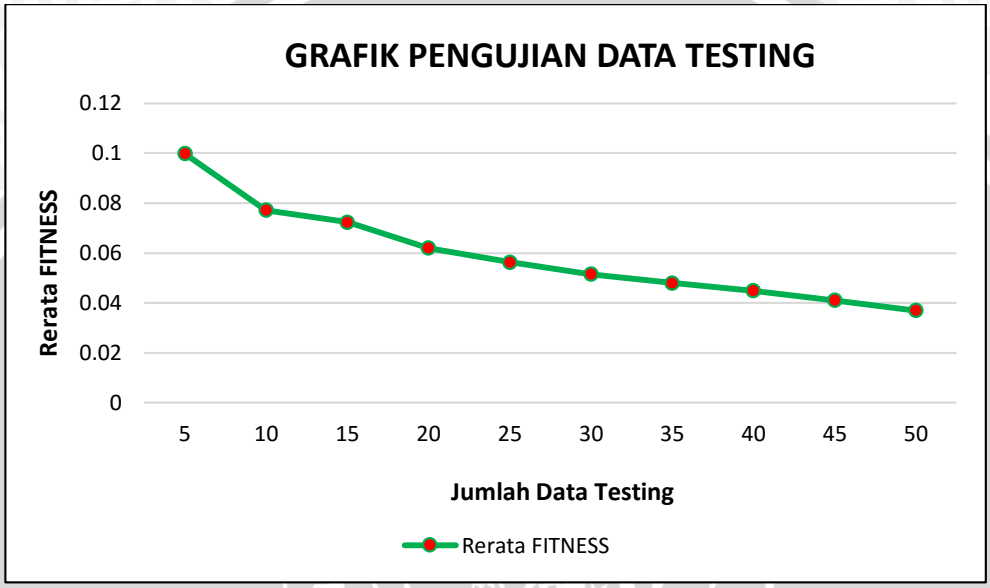
6.4 Hasil dan Analisa Uji Coba Jumlah Data Testing

Pada uji coba kali ini yaitu pengujian variasi Jumlah Data Testing. Data yang digunakan dalam uji coba ini merupakan data konsumsi air pada tahun 2008-2014. Jumlah data testing yang ideal dalam uji coba kali ini yaitu sebanyak 5 data testing. Berikut Tabel 6.4 Hasil Uji Coba Jumlah Data Testing.

Tabel 6.4 Hasil Uji Coba Jumlah Data Testing

Data *Testing*	Nilai Fitness Percobaan ke-*i*										Rerata FITNESS
	1	2	3	4	5	6	7	8	9	10	
5	0.1	0.099	0.1	0.099	0.101	0.099	0.1	0.099	0.1	0.101	0.0998
10	0.077	0.078	0.078	0.077	0.077	0.077	0.077	0.077	0.077	0.077	0.0772
15	0.072	0.073	0.072	0.072	0.073	0.073	0.073	0.072	0.072	0.072	0.0724
20	0.062	0.062	0.062	0.062	0.062	0.062	0.062	0.062	0.062	0.062	0.062
25	0.057	0.056	0.056	0.056	0.057	0.057	0.056	0.056	0.056	0.056	0.0563
30	0.052	0.051	0.051	0.052	0.051	0.051	0.052	0.052	0.052	0.051	0.0515
35	0.048	0.048	0.048	0.048	0.048	0.048	0.048	0.048	0.048	0.048	0.048
40	0.045	0.045	0.045	0.045	0.045	0.045	0.045	0.045	0.045	0.044	0.0449
45	0.041	0.041	0.041	0.041	0.041	0.041	0.041	0.041	0.041	0.041	0.041
50	0.037	0.037	0.037	0.037	0.037	0.037	0.037	0.037	0.037	0.037	0.037

Berdasarkan hasil pada Gambar 6.4 Grafik Pengujian Data Testing. Dapat disimpulkan bahwa jumlah data testing yang ideal adalah sejumlah 5. Nilai *fitness* yang dihasilkan adalah sebesar 0,1. Hal ini sama dengan halnya data training, karena data set yang tersedia pada penelitian kali ini terbatas dan variasi nilai pada tiap bulannya tidak terlalu jauh selisihnya. Hal ini mengakibatkan adanya range jumlah tertentu yang dapat menghasilkan peramalan terbaik. Karena hasil data testing juga dipengaruhi oleh karakteristik data training. Oleh karena itu kombinasi yang tepat untuk perbandingan penelitian ini adalah 30 data training dan 5 data testing.



Gambar 6.4 Grafik Pengujian Data Testing

6.5 Hasil dan Analisa Uji Coba Range Parameter *C*

Pada uji coba kelima ini merupakan uji coba batas range parameter *C*. pengujian ini dilakukan bertujuan untuk menentukan batas pencarian parameter *C* yang optimal dan terbaik dalam peramalan. Untuk pengujian pada parameter *C* ini yang digunakan adalah 10-100,101-200,201-300,301-400,401-500,501-600,601-700,701-800,801-900,901-1000. Pengujian dilakukan sebanyak 10 kali. Berikut detail parameter yang digunakan dalam uji coba batas parameter *C* :

a. Jumlah partikel : 10
b. Iterasi IPSO : 10
c. Iterasi SVR : 10000
d. Nilai cLR : 0.003
e. Nilai *Lambda* (λ) : 0.8
f. Nilai C_1 dan C_2 : 0.3 dan 0.7
g. Nilai r_1 dan r_2 : 0.2 dan 0.8

h. Nilai $w_{\min}$: 0.4

i. Nilai $w_{\max}$: 0.9

j. Range parameter ε : 0.0001 – 0.09

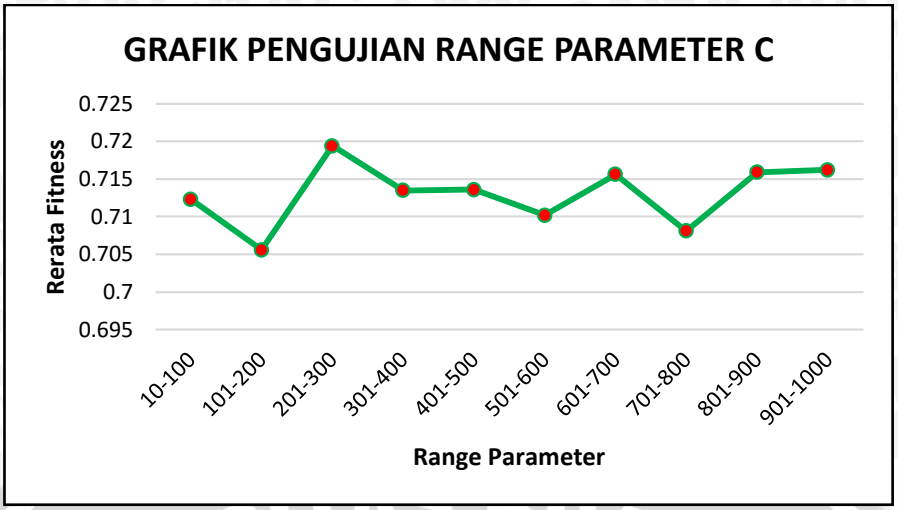
k. Range parameter σ : 0.0001 - 0.9999

Hasil pengujian dapat dilihat pada Tabel 6.5 di bawah ini :

Tabel 6.5 Hasil Uji Coba Range Parameter C

RANGE C	Nilai Fitness Percobaan ke-*i*										Rerata FITNESS
	1	2	3	4	5	6	7	8	9	10	
10-100	0.7190	0.7190	0.6810	0.7140	0.7190	0.7150	0.7190	0.7180	0.7200	0.6990	0.7123
101-200	0.6940	0.7190	0.6650	0.7170	0.7000	0.6900	0.7140	0.7190	0.7190	0.7190	0.7056
201-300	**0.7190**	**0.7200**	**0.7190**	**0.7190**	**0.7200**	**0.7190**	**0.7190**	**0.7190**	**0.7200**	**0.7200**	**0.7194**
301-400	0.7190	0.7120	0.7060	0.7130	0.7200	0.7130	0.7170	0.7190	0.7190	0.6970	0.7135
401-500	0.7190	0.6810	0.7200	0.7190	0.7190	0.7190	0.7150	0.7200	0.7200	0.7040	0.7136
501-600	0.7190	0.6970	0.7090	0.7190	0.7200	0.6940	0.7110	0.7180	0.7190	0.6960	0.7102
601-700	0.7190	0.7190	0.7190	0.7170	0.7200	0.7200	0.7190	0.7190	0.7190	0.6850	0.7156
701-800	0.7190	0.7170	0.6980	0.7190	0.6810	0.7190	0.7190	0.7190	0.7190	0.6710	0.7081
801-900	0.7140	0.7060	0.7180	0.7190	0.7190	0.7190	0.7150	0.7170	0.7190	0.7130	0.7159
901-1000	0.7200	0.7190	0.7180	0.7200	0.7190	0.7190	0.7130	0.7190	0.6960	0.7190	0.7162

Sesuai dengan grafik hasil pengujian dalam Gambar 6.5 di bawah. Nilai fitness terbesar dihasilkan pada range 201 – 300. Nilai *C* pada range 201-300 tersebut terbukti menghasilkan nilai fitness paling optimal dibandingkan dengan batas lainnya. *C* merupakan parameter dalam menentukan seberapa besar deviasi dari parameter epsilon yang masih dapat ditoleransi. Namun hasil rata-rata fitness yang dihasilkan masih di dalam range yang sama antara 0.70-0.72.



Gambar 6.5 Grafik Pengujian Range C

6.6 Hasil dan Analisa Uji Coba Range Parameter ε

Uji coba keenam ini merupakan uji coba range parameter ε. Pengujian ini diakukan untuk menemukan batas pencarian parameter ε paling optimal untuk menghasilkan solusi terbaik di dalam peramalan. Pengujian dilakukan sebanyak 10 kali. Berikut detail parameter lainnya yang digunakan :

a. Jumlah partikel : 10
b. Iterasi IPSO : 10
c. Iterasi SVR : 10000
d. Nilai cLR : 0.003
e. Nilai *Lambda* (λ) : 0.8
f. Nilai C_1 dan C_2 : 0.3 dan 0.7
g. Nilai r_1 dan r_2 : 0.2 dan 0.8
h. Nilai w_{min} dan w_{max} : 0.4 dan 0.9
i. Batas parameter pencarian σ : 0.0001 - 0.9999
j. Batas parameter pencarian *C* : 10 - 1000

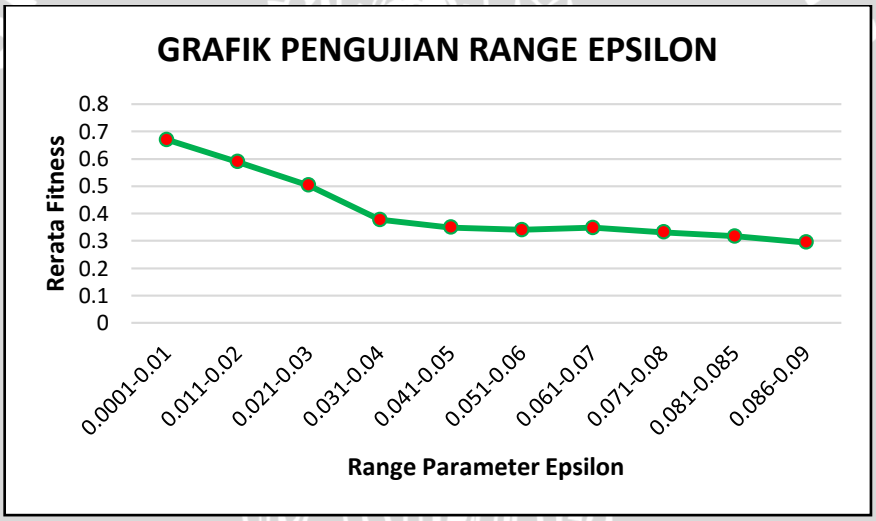
Berikut hasil uji coba range parameter epsilon. Dijelaskan pada Tabel 6.6 :

Tabel 6.6 Hasil Uji Coba Range Parameter ε

RANGE EPSILON	Nilai Fitness Percobaan ke-*i*										Rerata FITNESS
	1	2	3	4	5	6	7	8	9	10	
0.0001-0.01	**0.686**	**0.641**	**0.718**	**0.688**	**0.616**	**0.625**	**0.718**	**0.711**	**0.648**	**0.651**	**0.6702**
0.011-0.02	0.571	0.601	0.606	0.56	0.6	0.619	0.588	0.567	0.599	0.583	0.5894
0.021-0.03	0.529	0.427	0.522	0.534	0.426	0.545	0.556	0.43	0.54	0.522	0.5031
0.031-0.04	0.371	0.37	0.364	0.366	0.39	0.373	0.376	0.406	0.374	0.38	0.377
0.041-0.05	0.345	0.354	0.358	0.352	0.341	0.349	0.343	0.35	0.352	0.347	0.3491

0.051-0.06	0.34	0.341	0.352	0.338	0.339	0.344	0.336	0.338	0.335	0.339	0.3402
0.061-0.07	0.344	0.335	0.355	0.35	0.355	0.348	0.346	0.349	0.352	0.352	0.3486
0.071-0.08	0.338	0.321	0.329	0.343	0.331	0.318	0.334	0.33	0.335	0.338	0.3317
0.081-0.085	0.32	0.322	0.317	0.306	0.328	0.319	0.307	0.316	0.32	0.312	0.3167
0.086-0.09	0.293	0.299	0.292	0.295	0.307	0.282	0.282	0.285	0.305	0.307	0.2947

Berdasarkan hasil dari Gambar 6.6 Grafik Pengujian Range Epsilon. Dihasilkan bahwa nilai *fitness* terbesar dihasilkan dengan range antara 0.0001-0.01. Pada batas tersebut nilai *fitness* yang dihasilkan sebesar 0.6702, hasil ini menunjukkan batas nilai parameter paling optimal. Bahwa semakin kecil batas range epsilon akan menghasilkan nilai *fitness* terbaik untuk peramalan. Parameter epsilon mengontrol lebar dari zona regresi yang digunakan dalam mempelajari data dimana apabila semakin besar nilai epsilon maka estimasi regresi justru akan semakin datar (mendekati garis linier) namun ketika batas range epsilon semakin kecil maka estimasi regresi justru semakin baik dan terbukti dengan nilai *fitness* yang dihasilkan justru semakin meningkat ketika batas epsilon semakin kecil.



Gambar 6.6 Grafik Pengujian Range Epsilon

6.7 Hasil dan Analisa Uji Coba Range Parameter σ

Pada uji coba ketujuh ini merupakan uji coba range parameter σ. Pengujian ini dilakukan untuk menemukan batas pencarian parameter σ paling optimal dalam menghasilkan solusi terbaik di peramalan. Berikut detail parameter yang digunakan di dalam pengujian ini :

a. Jumlah partikel : 10
b. Iterasi IPSO : 10
c. Iterasi SVR : 10000
d. Nilai cLR : 0.003
e. Nilai *Lambda* (λ) : 0.8

f. Nilai C_1 dan C_2 : 0.3 dan 0.7

g. Nilai r_1 dan r_2 : 0.2 dan 0.8

h. Nilai w_{min} dan w_{max} : 0.4 dan 0.9

i. Batas parameter pencarian ε : 0.0001 – 0.09

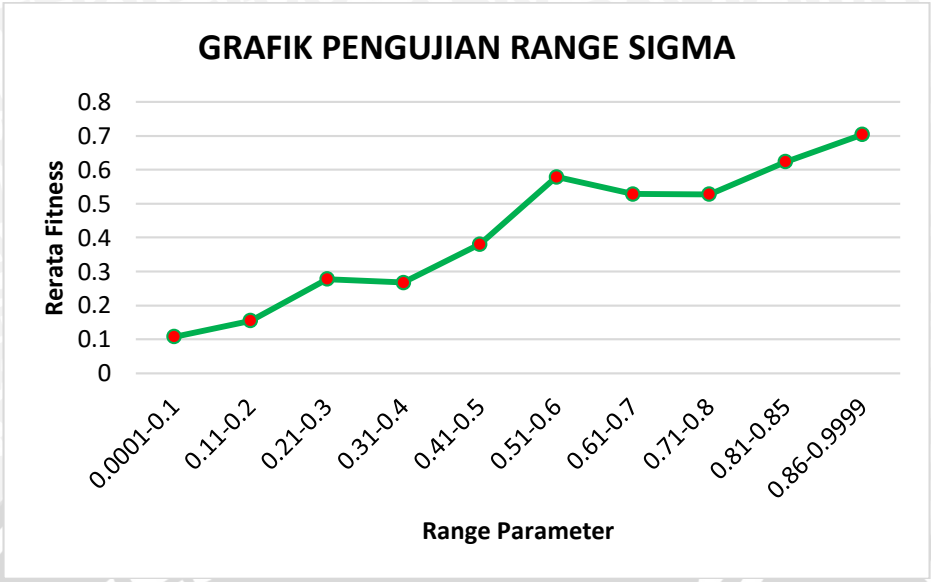
j. Batas parameter pencarian *C* : 10 – 1000

Berikut Tabel 6.7 hasil uji coba range parameter σ :

Tabel 6.7 Hasil Uji Coba Range Parameter σ

RANGE SIGMA	Nilai Fitness Percobaan ke-*i*										Rerata FITNESS
	1	2	3	4	5	6	7	8	9	10	
0.0001-0.1	0.116	0.096	0.096	0.107	0.125	0.098	0.096	0.125	0.12	0.097	0.1076
0.11-0.2	0.152	0.174	0.157	0.156	0.157	0.15	0.149	0.149	0.152	0.157	0.1553
0.21-0.3	0.276	0.275	0.276	0.274	0.28	0.274	0.281	0.279	0.279	0.284	0.2778
0.31-0.4	0.266	0.267	0.27	0.267	0.266	0.268	0.266	0.267	0.27	0.266	0.2673
0.41-0.5	0.398	0.366	0.363	0.37	0.378	0.374	0.378	0.366	0.451	0.36	0.3804
0.51-0.6	0.575	0.586	0.584	0.567	0.589	0.58	0.567	0.587	0.576	0.575	0.5786
0.61-0.7	0.533	0.523	0.532	0.532	0.516	0.533	0.521	0.533	0.534	0.532	0.5289
0.71-0.8	0.534	0.524	0.524	0.528	0.523	0.526	0.531	0.527	0.528	0.53	0.5275
0.81-0.85	0.625	0.625	0.625	0.627	0.62	0.622	0.623	0.622	0.623	0.622	0.6234
0.86-0.9999	**0.705**	**0.683**	**0.705**	**0.713**	**0.71**	**0.719**	**0.701**	**0.719**	**0.698**	**0.686**	**0.7039**

Berdasarkan grafik hasil uji coba batas nilai parameter sigma yang dijelaskan pada Gambar 6.7, rata-rata nilai fitness terbesar didapatkan pada batas 0.86 – 0.9999. hasil ini membuktikan bahwa batas paling optimal untuk nilai parameter sigma yang terbaik untuk peramalan ini adalah batas 0.86 sampai 0.9999. Grafik ini menunjukkan bahwa semakin besar range yang digunakan maka hasil *fitness* yang dihasilkan akan semakin besar, maka semakin besar batas range nilai sigma yang didaptkan menunjukkan bahwa nilai yang digunakan pada regresi juga semakin besar sehingga menghasilkan peramalan yang lebih akurat.



Gambar 6.7 Grafik Pengujian Range σ

BAB 7 KESIMPULAN DAN SARAN

Di dalam bab ini penulis akan membahas kesimpulan dan saran terkait penelitian yang telah diselesaikan yaitu peramalan konsumsi air di PDAM Kota Malang menggunakan metode *Support Vector Regression* dengan *Improve Particle Swarm Optimization*.

7.1 Kesimpulan

Berikut hasil penelitian mengenai peramalan konsumsi air di PDAM Kota Malang menggunakan metode *Support Vector Regression* dengan *Improve Particle Swarm Optimization* :

1. Algoritma *Improve Particle Swarm Optimization* dapat diimplementasikan untuk optimasi peramalan konsumsi air di PDAM Kota Malang menggunakan metode *Support Vector Regression* (SVR) dengan menemukan koefisien-koefisien terbaik untuk setiap parameter SVR yang dioptimasi. Adapun koefisien terbaik yang diperoleh dalam penelitian ini yaitu :
   - nilai Kompleksitas (*C*) yang terletak pada range [201-300]
   - nilai Epsilon (ε) yang terletak pada range [0.0001-0.1]
   - nilai Sigma (σ) yang terletak pada range [0.86-0.9999]
2. Tabulasi data yang digunakan dalam SVR adalah sebanyak 30 data yang terdiri dari data konsumsi air di PDAM Kota Malang dalam rentang waktu perbulan, dengan panjang fitur data sebanyak 6 fitur. Pada sistem ini algoritma *Improve Particle Swarm Optimization* dalam mengoptimasi peramalan konsumsi air di PDAM Kota Malang menggunakan metode *Support Vector Regression* menghasilkan tingkat akurasi nilai *fitness* sebesar 0.719.

7.2 Saran

Berikut merupakan saran yang dapat digunakan untuk penelitian selanjutnya:

1. Pada penelitian selanjutnya dapat menambahkan penggunaan peramalan data konsumsi air dengan menggunakan analisis fundamental. Sebagai contoh dalam penelitian ini hanya menggunakan analisis teknikal. Analisis fundamental memiliki faktor – faktor yang mempengaruhi data konsumsi air sebagai fitur peramalan.
2. Pada penelitian selanjutnya dapat menggunakan penambahan *Time Varying Acceleration Coefficients* (TVAC). Dalam penelitian ini, penulis menggunakan nilai konstan untuk koefesien akselerasi c_1 dan c_2 sehingga menyebabkan tidak adanya keseimbangan antara eksplorasi global dan eksploitasi local.

DAFTAR PUSTAKA

Lu S., Cai Z., Zhang X., 2009. *Forecasting Agriculture Water Consumption Based on PSO and SVM.* School of Computer Science and Information Engineering Chongqing Technology and Business University Chongqing, China.

Zou, J., Li, C., Yang, Q., 2015. *Fault Prediction Method based on SVR of Improved PSO*. School of Information Science and Engineering, Shenyang Ligong University, Shenyang 110159, China.

Supijatno., Chozin Ah., Sopandie Di., Tri., Junaedi A., Lubis I., 2012. *Water Consumption Evaluation among Rice Genotypes Showing Possibility to Explore Benefit of Water Use Efficiency*. Departemen Agronomi dan Hortikultura, Fakultas Pertanian, Institut Pertanian Bogor. 5 Januari 2012.

Wachjar A., Angga R., 2013. *Productivity Increasement and Water Consumption Efficiency of Amaranth (Amaranthus tricolor L.) in Hydroponic Technique by Plant Population Arrangement.* Departemen Agronomi dan Hortikultura, Fakultas Pertanian, Institut Pertanian Bogor.

Briawan D., Rachma P., Annisa K., 2011. *Kebiasaan Konsumsi Minuman Dan Asupan Cairan Pada Anak Usia Sekolah Di Perkotaan*. Departemen Gizi Masyarakat, Fakulas Ekologi Manusia, Institut Pertanian Bogor 16680.

Sutarno., 2010. *Analisis Perbandingan Transformasi Wavelet pada Pengenalan Citra Wajah.* Fakultas Ilmu Komputer, Universitas Sriwijaya, Vol.5 No.2 Juli 2010.

Ratnawera, A., Saman, K., Halgamuge., 2004. *Self-Organizing Hierarchical Particle Swarm Optimizer With Time-Varying Acceleration Coefficients*. IEEE Transactions On Evolutionary Computation, Vol. 8, No. 3, June 2004.

Kinoshita, K., Watanabe, K., Isshiki., M., 2014. *Estimation of Inverse Model Based on ANN and PSO with Adaptively Varying Acceleration Coefficients*. SICE Annual Conference 2014 September 9-12, 2014, Hokkaido University, Sapporo, Japan.

Liu, B.X., Wu, Y., Xu, X., dkk., 2014. *Chaos Adaptive Improved Particle Swarm Algorithm for Solving Multi-Objective Optimization*. TELKOMNIKA Indonesian Journal of Electrical Engineering Vol.12, No.1, January 2014, pp. 703 – 710.

Wang , Li., Zhao., Qi, Liu L., *The Application of BP Neural Network Based on Improved PSO in BF Temperature Forecast.*

Ilka, R., Gholamian S.A., 2012. *Optimum Design of a Five-phase PMSM for Underwater Vehicles by use of PSO*. TELKOMNIKA Indonesian Journal of Electrical Engineering Vol.10, No.5, September 2012, pp. 925 – 932.

Wang, F., Tan, G., Deng, Z., dkk., 2008. *Real-time Traffic Flow Forecasting Model and Parameter Selection based on Ɛ-SVR*. Proceedings of the 7th World Congress on Intelligent Control and Automation June 25 - 27, 2008, Chongqing, China.

Ling W., Mei Fu., 2009. *A Novel Approach Using SVR Ensembles for Minor Prototypes Prediction of Seawater Corrosion Rate.* Automation department, Information Engineering School University of Science and Technology Beijing, China.

Iza R, Aswin M, Mustofa A., 2013 *Steganografi Pada Citra Digital Menggunakan Metode Discrete Wavelet Transform*, Jurusan Teknik elektro Fakultas Teknik Universitas Brawijaya. Malang,Indonesia.

Sutarno., 2010. *Analisis Perbandingan Transformasi Wavelet pada Pengenalan Citra Wajah*. Fakultas Ilmu Komputer, Universitas Sriwijaya. Vol.5, No.2, Juli 2010

Vijayakumar, S., Wu S., 1999. Sequential Support Vector Classifiers and Regression. RIKEN Brain Science Institute 1999, pp.610 – 619.

Rajkumar, N., Jaganathan., P., 2013. *A New RBF Kernel Based Learning Method Applied to Multiclass Dermatology Diseases Classification*. Proceedings of 2013 IEEE Conference on Information and Communication Technologies (ICT 2013).

Putri R, Dewi C, Indriati.,2013, *Implementasi Algoritma Particle Swarm Optimization Untuk Optimasi Fungsi Keanggotaan Pada Kondisi Penderita Penyakit Hepatitis*. Program Studi Ilmu Komputer, Jurusan Ilmu Komputer Fakultas Program Teknologi Informasi dan Ilmu Komputer Jl. Veteran No 8, Malang 65145, Indonesia.

Qing Y, Jian-qiao, Chang L., Nian L., 2013 Real-Time fault diagnosis approach based on lifting wavelet and recursive incremental clustering. System Engineering and Electronics, China. Vol.35, No.1 January 2013.

LAMPIRAN A DATA KONSUMSI AIR PDAM KOTA MALANG

A.1 Data Konsumsi Air Tahun 2008-2014 Per Bulan

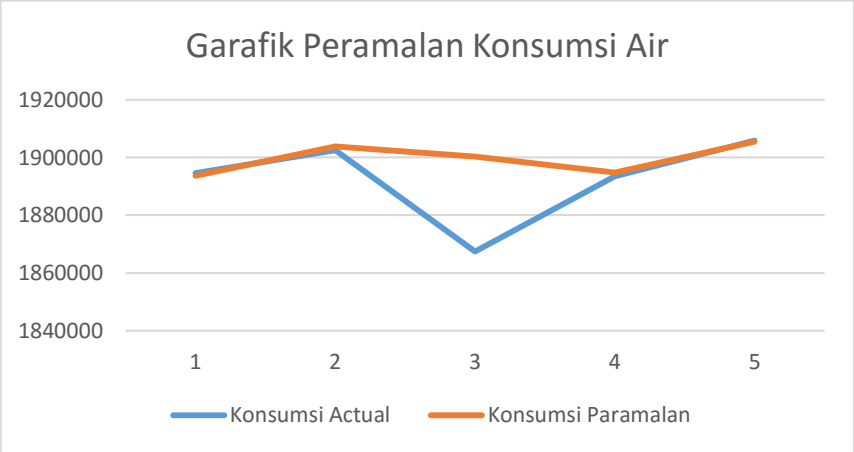
TAHUN/BULAN	JANUARI	FEBRUARI	MARET	APRIL	MEI	JUNI
2008	1684536	1627438	1592618	1702344	1611804	1658421
2009	1773087	1658690	1680750	1765834	1799518	1763284
2010	1807659	1752567	1773785	1901589	1858807	1825106
2011	1894655	1902542	1867345	1893539	1905850	1824748
2012	1986071	2014645	1908264	2192472	1999456	2013574
2013	2127658	1987364	2068653	2118364	2092653	1936538
2014	2235453	2153579				

TAHUN/BULAN	JULI	AGUSTUS	SEPTEMBER	OKTOBER	NOVEMBER	DESEMBER
2008	1568368	1729248	1546298	1417037	1606972	1558064
2009	1837567	1698905	1772154	1732172	1694870	1751939
2010	1869328	1820043	1780592	1861409	1913115	1894268
2011	1920348	1932854	1888674	1953525	1924508	1943646
2012	2102384	1934682	2024783	2014573	2240356	2374281
2013	2287385	2178435	2278648	2317424	2144479	2198034
2014						

LAMPIRAN B VISUALISASI HASIL PENGUJIAN

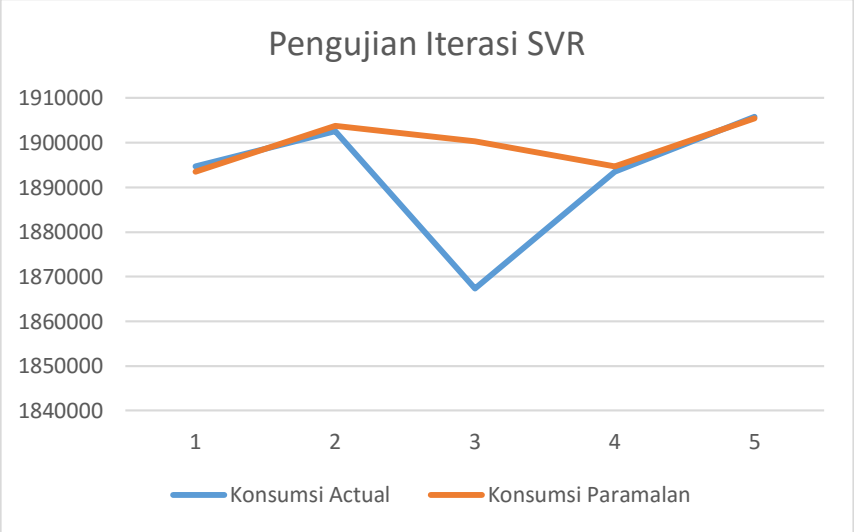
B.1 GRAFIK HASIL PENGUJIAN JUMLAH ITERASI IPSO

Grafik di bawah ini merupakan hasil pengujian terhadap banyaknya jumlah iterasi metode *Improve Particle Swarm Optimization* yaitu sebanyak 100 iterasi.



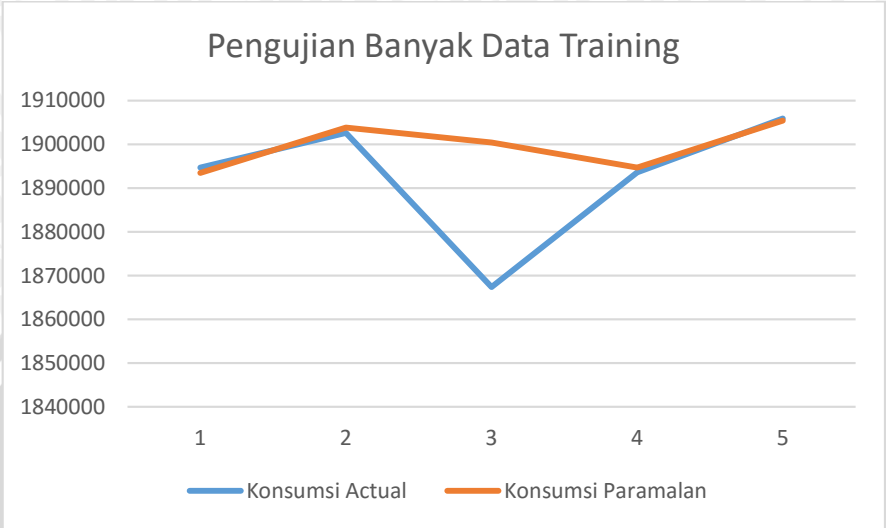
B.2 GRAFIK HASIL PENGUJIAN JUMLAH ITERASI SVR

Grafik di bawah ini merupakan hasil pengujian terhadap banyaknya jumlah iterasi metode *Support Vector Regression* dengan jumlah iterasi sebanyak 10000 kali. Dengan iterasi IPSO sebanyak 10 kali.



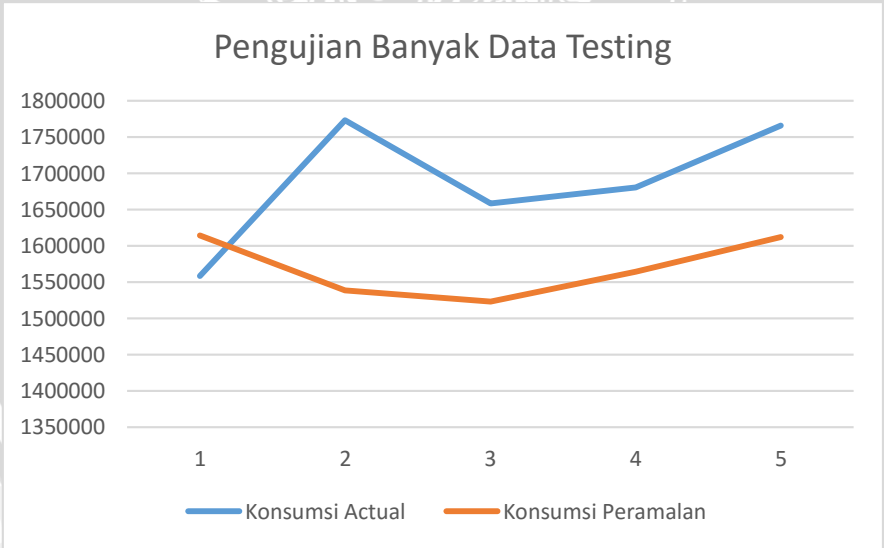
B.3 GRAFIK HASIL PENGUJIAN JUMLAH DATA TRAINING

Grafik di bawah ini merupakan hasil pengujian terhadap variasi jumlah data *training* yaitu dengan menggunakan 30 data *training*.



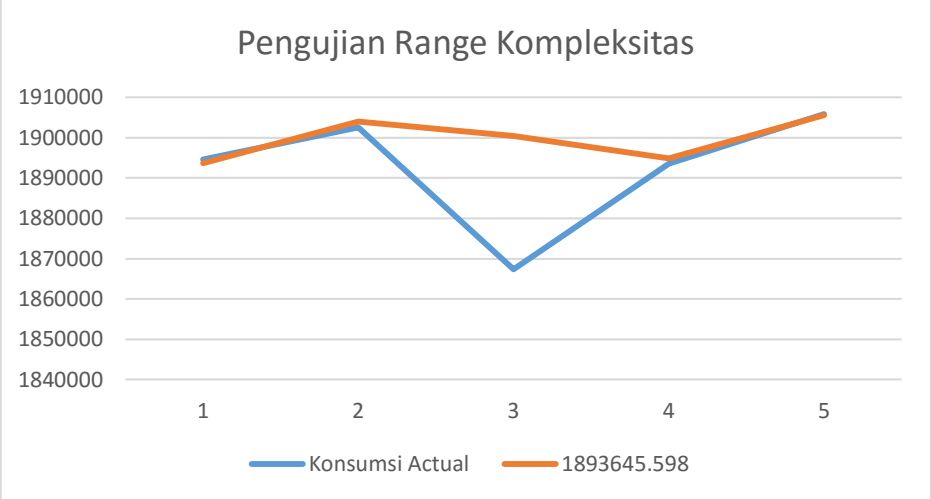
B.4 GRAFIK HASIL PENGUJIAN JUMLAH DATA TESTING

Grafik di bawah ini merupakan hasil pengujian terhadap variasi jumlah data *testing* yaitu dengan menggunakan 5 data *testing*.



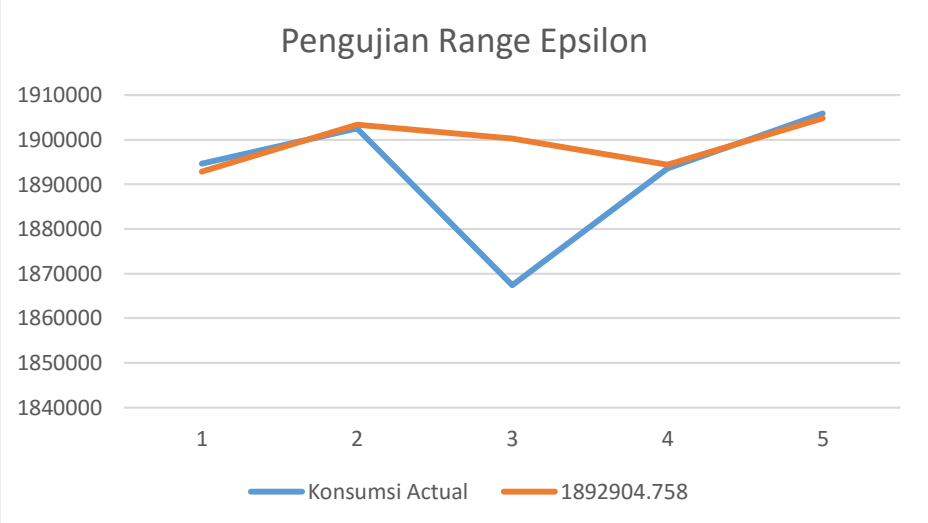
B.5 GRAFIK HASIL PENGUJAN RANGE PARAMETER *C*

Grafik di bawah ini merupakan hasil pengujian terhadap batas nilai kompleksitas (C) metode SVR dengan batas antara 10 sampai 1000



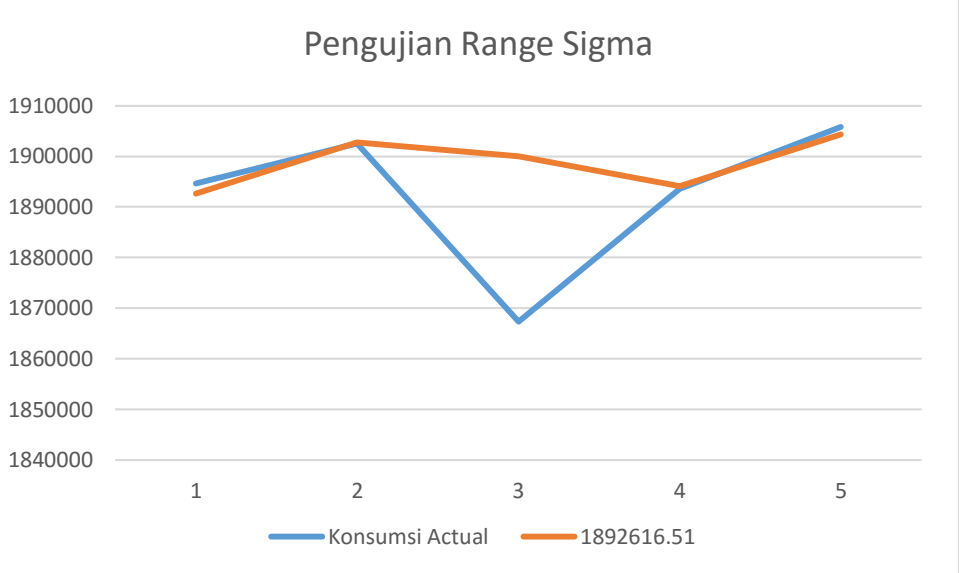
B.6 GRAFIK HASIL PENGUJIAN RANGE PARAMETER ε

Grafik di bawah ini merupakan hasil pengujian terhadap batas nilai epsilon metode SVR dengan batas nilai antara 0.0001 sampai 0.09.



B.7 GRAFIK HASIL PENGUJIAN RANGE PARAMETER σ

Grafik di bawah ini merupakan hasil pengujian terhadap batas nilai sigma metode SVR dengan batas nilai antara 0.0001-0.9999.



B.8 Nilai Parameter Optimal Untuk Peramalan Data Konsumsi Air Di PDAM Kota MALANG Menggunakan Metode SVR-IPSO

Di bawah ini merupakan hasil nilai-nilai yang optimal untuk setiap variabel yang ada dalam metode SVR dan metode IPSO setelah pengujian yang telah dilakukan :

1) Range Parameter *C* : 201 - 300
2) Range Parameter ε : 0.0001-0.1
3) Range Parameter σ : 0.86-0.9999
4) Jumlah Partikel : 5
5) Jumlah Iterasi IPSO : 80
6) Jumlah Iterasi SVR : 10000
7) Banyak Data Training : 30
8) Banyak Data Testing : 5
9) Banyak Fitur Data : 7
10) Nilai *cLR* : 0.003
11) Nilai λ : 0.8
12) Nilai C_1 : 0.3
13) Nilai C_2 : 0.7
14) Nilai r_1 : 0.2

15) Nilai r_2 : 0.8

16) Nilai w_{min} : 0.4

17) Nilai w_{max} : 0.9

18) Nilai *minChi* : 0.0

19) Nilai *maxChi* : 22.5

20) Batas *minChi* : 0.760911593

Hasil peramalan data konsumsi air di PDAM MALANG dengan menggunakan parameter di atas menghasilkan nilai MAPE sebesar 0.3889571 dan nilai *Fitness* sebesar 0.719.

B.9 Nilai *αi** dan *αi* optimal untuk Fungsi Regresi pada SVR

Tabel di bawah ini merupakan nilai *α** dan *α* yang optimal untuk peramalan SVR yang didapatkan pada iterasi terakhir yaitu iterasi 10000 :

NO	αi*	αi
1	0.0	1.9130537688789608
2	1.5999915259215427	0.0
3	0.0	2.6502473000086146
4	0.0	4.284146201504206
5	0.37292641399049503	0.0
6	0.0	0.6120665885419346
7	2.8838610700452003	0.0
8	0.12953344149079338	0.0
9	0.11578616491380989	0.016920870883111155
10	1.1597303045113596	0.0
11	1.4404907787785717	0.0
12	0.0	0.08419143336715545
13	1.2807216942244175	0.0
14	0.0	1.9332901256316521
15	0.02349592344798465	0.02011798056373906
16	0.0	0.9031005611625879
17	0.0	1.2891845022880002
18	0.12464802230351459	0.0

NO	αi*	αi
19	1.1509906404089338	0.0
20	0.0	0.4156096353796484
21	0.014463802803824822	0.0
22	2.318486767075055	0.0
23	0.6857886268228734	0.0
24	0.0	0.3131895974386355
25	0.4930397160885984	0.0
26	0.0	0.6166560006708565
27	0.0	1.1532530206151606
28	0.6542434372179862	0.0
29	1.4475571506634188	0.0
30	0.6779943812557613	0.0

LAMPIRAN C DETAIL DATA PENGUJIAN

C.1 PENGGUNAAN SEQUENCE DATA *TRAINING* PERAMALAN

Tahun	X1	X2	X3	X4	X5	X6	X7	Y
Agustus 2008	1684536	1627438	1592618	1702344	1611804	1658421	1568368	1729248
September 2008	1627438	1592618	1702344	1611804	1658421	1568368	1729248	1546298
Oktober 2008	1592618	1702344	1611804	1658421	1568368	1729248	1546298	1417037
November 2008	1702344	1611804	1658421	1568368	1729248	1546298	1417037	1606972
Desember 2008	1611804	1658421	1568368	1729248	1546298	1417037	1606972	1558064
Januari 2009	1658421	1568368	1729248	1546298	1417037	1606972	1558064	1773087
Februari 2009	1568368	1729248	1546298	1417037	1606972	1558064	1773087	1658690
Maret 2009	1729248	1546298	1417037	1606972	1558064	1773087	1658690	1680750
April 2009	1546298	1417037	1606972	1558064	1773087	1658690	1680750	1765834
Mei 2009	1417037	1606972	1558064	1773087	1658690	1680750	1765834	1799518
Juni 2009	1606972	1558064	1773087	1658690	1680750	1765834	1799518	1763284
Juli 2009	1558064	1773087	1658690	1680750	1765834	1799518	1763284	1837567
Agustus 2009	1773087	1658690	1680750	1765834	1799518	1763284	1837567	1698905
September 2009	1658690	1680750	1765834	1799518	1763284	1837567	1698905	1772154
Oktober 2009	1680750	1765834	1799518	1763284	1837567	1698905	1772154	1732172
November 2009	1765834	1799518	1763284	1837567	1698905	1772154	1732172	1694870
Desember 2009	1799518	1763284	1837567	1698905	1772154	1732172	1694870	1751939
Januari 2010	1763284	1837567	1698905	1772154	1732172	1694870	1751939	1773785
Februari 2010	1837567	1698905	1772154	1732172	1694870	1751939	1807659	1752567
Maret 2010	1698905	1772154	1732172	1694870	1751939	1807659	1752567	1773785
April 2010	1772154	1732172	1694870	1751939	1807659	1752567	1773785	1901589
Mei 2010	1732172	1694870	1751939	1807659	1752567	1773785	1901589	1858807
Juni 2010	1694870	1751939	1807659	1752567	1773785	1901589	1858807	1825106
Juli 2010	1751939	1807659	1752567	1773785	1901589	1858807	1825106	1869328
Agustus 2010	1807659	1752567	1773785	1901589	1858807	1825106	1869328	1820043

September 2010	1752567	1773785	1901589	1858807	1825106	1869328	1820043	1780592

C.2 PENGGUNAAN SEQUENCE DATA *TESTING* PERAMALAN

Tahun	X1	X2	X3	X4	X5	X6	X7	Y
Januari 2011	1825106	1869328	1820043	1780592	1861409	1913115	1894268	1894655
Februari 2011	1869328	1820043	1780592	1861409	1913115	1894268	1894655	1902542
Maret 2011	1820043	1780592	1861409	1913115	1894268	1894655	1902542	1867345
April 2011	1780592	1861409	1913115	1894268	1894655	1902542	1867345	1893539
Mei 2011	1861409	1913115	1894268	1894655	1902542	1867345	1893539	1905850