PRAKIRAAN KEBUTUHAN ENERGI LISTRIK MENGGUNAKAN *SUPPORT VECTOR REGRESSION* (SVR) DENGAN OPTIMASI ALGORITMA GENETIKA (STUDI KASUS PLN KOTA MALANG)

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:
Tanti Meta Sari
NIM: 125150201111041

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

PENGESAHAN

PRAKIRAAN KEBUTUHAN ENERGI LISTRIK MENGGUNAKAN *SUPPORT VECTOR REGRESSION* (SVR) DENGAN OPTIMASI ALGORITMA GENETIKA (STUDI KASUS PLN KOTA MALANG)

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh :
Tanti Meta Sari
NIM: 125150201111041

Skripsi ini telah diuji dan dinyatakan lulus pada
21 Juli 2016
Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I	Dosen Pembimbing II
Imam Cholissodin, S.Si, M.Kom	Budi Darma Setiawan, S.Kom, M.Cs
NIK: 201201 850719 1 001	NIP: 198410152014041002

Mengetahui
Ketua Jurusan Teknik Informatika

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

PERNYATAAN ORISINALITAS

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

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

Malang, 8 Agustus 2016

Tanti Meta Sari

NIM: 125150201111041

KATA PENGANTAR

Puji dan syukur penulis panjatkan kepada Tuhan Yang Maha Esa atas berkat, rahmat, ridho dan karunia-Nya sehingga penulis dapat menyeleseikan tugas akhir yang berjudul "Prakiraan Kebutuhan Energi Listrik Menggunakan *Support Vector Regression* (SVR) dengan Optimasi Algoritma Genetika (Studi Kasus PLN Kota Malang)" sebagai salah satu persyaratan untuk menyeleseikan studi di Jurusan Informatika, Fakultas Ilmu Komputer Universitas Brawijaya.

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

1. Bapak Imam Cholissodin, S.Si, M.Kom, selaku dosen pembimbing I dan Bapak Budi Darma Setiawan, S.Kom, M.Cs yang telah banyak sekali memberikan ilmu, membantu dan membimbing serta saran untuk skripsi ini.
2. Ibu Rekyan Regasari Mardi Putri, S.T, M.T dan Bapak Barlian Henryranu Prasetio, S.T, M.Tdosen penguji yang telah meluluskan saya serta banyak memberikan kritik dan juga saran untuk skripsi agar menjadi lebih baik lagi.
3. Seluruh dosen Fakultas Ilmu Komputer Universitas Brawijaya Malang atas segala bimbingan serta ilmu yang telah diajarkan kepada penulis
4. Orang tua penulis dan saudara penulis, Bapak Anwar dan Ibu Eva Siswiyanti, dan Tania Agustia Sari yang memberikan dukunngan moral dan material.
5. Sahabat penulis Diana Maulida Putri Wijayanti dan Diana Oktavianti yang selalu memberi semangat penulis dalam berbagai situasi. Terimakasih atas perhatian dan bantuannya.
6. Teman penulis Tenika, Christine, Nungky, Mila, Mega yang selalu menemani penulis dalam menuntut ilmu dari awal kuliah dan membantu penulis saat membutuhkan bantuan.
7. Teman-teman Cemara Diana, Juju, Zilfikri, Fahmi, There, Yuyon, Irsyad, Ibnu, Dito, Afif dan Wawan, terima kasih atas segala bantuan yang diberikan.
8. Seluruh pengurus dan staff Badan Eksekutif Mahasiswa Kabinet Bersatu II atas dukungan dan motivasinya.
9. Teman-teman IF-F terimakasih atas kebersamaan serta bantuan yang telah diberikan dan kita lakukan bersama dari semester 1 hingga semester 8 ini
10. Sahabat SMA penulis Firsthiyana Kharismatika Adie Jelita terimakasih atas support yang selalu diberikan semoga kelak kita sukses bersama.
11. Teman-teman Akselerasi SMP Angga, Ata, Atika, Trimaryanto, Hida, Eno, Raynaldi, Rizandre, Savitri, Suci, Mita, Tia, Triyoga, Uqiek, Zakaria dan Rohbi terimakasih atas motivasi dan bantuannya dalam penyusunan skripsi ini.

12. Teman-teman seperjuangan Fakultas Komputer angkatan 2012 atas rasa kebersamaan yang begitu erat.
13. 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 bersifat membangun sangat diharapkan untuk menyempurnakan skripsi ini. Penulis berharap skripsi ini dapat bermanfaat khususnya bagi diri sendiri dan bagi semua pihak.

Malang, 8 Agustus 2016

Tanti Meta Sari

tantimetasari@gmail.com

ABSTRAK

Listrik merupakan salah satu kebutuhan manusia yang paling penting. Hampir semua kegiatan manusia membutuhkan energi listrik. Tingginya kebutuhan konsumsi manusia akan listrik menyebabkan semakin tinginya konsumsi listrik yang dibutuhkan. Kota Malang merupakan salah satu kota yang memiliki penduduk yang besar yaitu 973.716 jiwa per tanggal 1 Juli 2015. Dengan besarnya jumlah penduduk Kota Malang maka berbanding lurus dengan kebutuhan akan konsumsi listrik. Mengingat konsumsi listrik Kota Malang yang cenderung meningkat terus menerus setiap bulannya, maka dibutuhkan perancangan kedepannya untuk membangun pembangkit listrik baru jika sewaktu-waktu jumlah konsumsi listrik melebihi batas maksimum yang mampu disediakan. Namun untuk membangun pembangkit baru memerlukan biaya yang sangat besar dan perancangan yang sangat matang. Oleh karena prakiraan konsumsi energi listrik sangat diperlukan untuk membantu membuat keputusan dalan membangun pembangkit listrik. Dengan demikian prakiraan kebutuhan konsumsi listrik merupakan langkah tepat untuk mengantisipasi kebutuhan energi listrik yang diduga akan bertambah setiap bulannya. Banyak teknik prakiraan atau peramalan yang dapat digunakan untuk memprakiraan jumlah konsumsi listrik, salah satunya adalah Support Vector Regression (SVR). SVR adalah pengembangan dari Support Vector Machine untuk masalah prakiraan dan peramalan untuk kasus regresi. Support Vector Regression meminimalisir pembatasan dikarenakan metode ini memungkinkan penggunaan fungsi non-linier. Berdasarkan permasalahan beserta solusi dari beberapa penelitian sebelumnya, dirasa belum optimal. Saat ini, banyak metode baru untuk melakukan optimasi yang dirasa mampu menyelesaikan permasalahan di atas. Salah satu algoritma yang dapat digunakan untuk optimasi adalah algoritma genetika. Algoritma genetika adalah metode ilmiah heuristik yang didasarkan pada teori evolusi biologis Darwin. Sesuai dengan pengujian yang dilakukan menggunakan data yang didapat dari PLN Kota Malang berupa konsumsi kebutuhan listrik (kWh) dari tahun 2010 hinggan 2014, metode SVR yang dioptimasi dengan algoritma genetika dapat menghasilkan MAPE dengan nilai 0.161741.

Kata kunci: Konsumsi Listrik, kWh, Kota Malang, SVR, Algoritma Genetika, MAPE

ABSTRACT

Electricity is one of the most important human needs. Almost all human activities requiring electricity. The high demand for human consumption for electricity contributed to greater consumption of electricity needed. Malang is a city that has a large population is 973.716 inhabitants as of July 1, 2015. With a large population of Malang then directly proportional to the electricity consumption requirements. Considering Malang electricity consumption which tend to increase continuously every month, it takes planning to build new power plants if at any time the amount of electricity consumption exceeds the maximum limit able to be provided. However, to build new power plants require huge funds and careful design. Thus, forecasts of electricity consumption is the right step to anticipate the needs of the electrical energy which is expected to increase each month. Many forecasting techniques that can be used to estimate the amount of electricity consumption such as Support Vector Regression (SVR). SVR is the development of Support Vector Machine to issue forecasting for the case of regression. Support Vector Regression minimize restriction because this method allows the use of non-linear functions. Based on the problems and solutions of some previous studies, deemed not optimal. Today, many new methods to perform optimization felt able to solve the problems above. One algorithm that can be used for optimization is a genetic algorithm. Genetic algorithm is heuristic scientific method that is based on Darwin's theory of biological evolution. According to tests performed using data obtained from PLN Malang in the form of consumption demand for electricity (kWh) from 2010 to 2014, SVR method optimized with genetic algorithm can produce MAPE with a value of 0.161741.

Key words: *Electricity Consumption, kWh, Malang City, SVR, Genethic Alghorithm, MAPE*

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB 1 PENDAHULUAN

1.1 Latar belakang

Kota Malang merupakan salah satu kota berkembang yang ada di Jawa Timur. Kota Malang memiliki jumlah penduduk yang besar dan meningkat setiap tahunnya. Menurut Dinas Kependudukan dan Pencatatan Sipil Kota Malang per tanggal 1 Juli 2015, jumlah penduduk Kota Malang mencapai 873.716 jiwa. Sebagai kota yang padat dan sibuk, Kota Malang membutuhkan pasokan listrik yang besar. Hal ini dikarenakan Kota Malang memiliki 31 Perguruan Tinggi dan 83 Industri Besar dan Sedang (BPS Kota Malang, 2013).

Kebutuhan akan listrik memegang peranan penting dalam melakukan kegiatan sehari-hari sehingga diperlukan perhatian khusus dalam menanganinya. Seiring dengan peningkatan kegiatan sektor industri dan pendidikan, penggunaan listrik pun mengalami peningkatan. Untuk memenuhi kebutuhan warga akan listrik, maka pihak penyuplai energi listrik dalam hal ini adalah PLN perlu melakukan pengembangan penyediaan tenaga listrik seperti pembangunan pembangkit baru yang memerlukan waktu yang lama dan biaya yang besar dan diusahakan agar daya yang dibangkitkan cukup untuk memenuhi kebutuhan konsumen. Peningkatan konsumsi listrik tersebut akan berakibat fatal jika tidak ditangani dengan baik. Oleh karena prakiraan konsumsi energi listrik sangat diperlukan untuk membantu membuat keputusan dalan membangun pembangkit listrik. Dengan demikian prakiraan kebutuhan konsumsi listrik merupakan langkah tepat untuk mengantisipasi kebutuhan energi listrik yang diduga akan bertambah setiap bulannya.

Peramalan kebutuhan konsumsi listrik yang tepat dapat memberikan keuntungan pada segi ekonomi dan efisien dalam melakukan pendistribusian listrik. Pembangunan pembangkit listrik yang salah dapat menyebabkan kerugian besar. Jika pembangunan pembangkit listrik lebih sedikit dari kebutuhan konsumsi listrik, maka dapat menyebabkan krisis listrik. Sebaliknya, jika melebihi kebutuhan konsumsi listrik maka menyebabkan beban listrik yang besar sehingga membuang-buang sumber daya (Wang et al., 2013).

Untuk saat ini, pihak PLN Kota Malang belum mempunyai sistem untuk memprakirakan kebutuhan energi listrik. Mereka hanya menggunakan rumus rata-rata sederhana untuk memperkirakan penggunaan hari berikutnya. Sistem mereka hanya memprakirakan kebutuhan energi istrik dalam jumlah hari, tidak bulan atau tahun. Hal itu menunjukkan bahwa pihak PLN Kota Malang hanya bisa memprakirakan kebutuhan energi listrik dalam jangka pendek. Untuk itu diperlukan sebuah sistem yang mampu memprakirakan kebutuhan energi listrik untuk jangka panjang.

Banyak teknik prakiraan atau peramalan yang dapat digunakan untuk memprakiraan jumlah konsumsi listrik (Yi-feng et al., 2010; Bagnasco et al., 2015; Escrivá et al., 2011; Kaytez et al., 2015). Prakiraan jumlah konsumsi listrik ini termasuk pada kasus regresi. Regresi tradisional atau regresi linier sering

dinyatakan memiliki penyimpangan paling sedikit. *Support Vector Regression* (SVR) adalah pengembangan dari *Support Vector Machine* untuk masalah prakiraan dan peramalan untuk kasus regresi. *Support Vector Regression* meminimalisir pembatasan dikarenakan metode ini memungkinkan penggunaan fungsi non-linier (Ogcu et al., 2012).

Salah saru paper membahas analisis implementasi jaringan syaraf adaptif untuk peramalan kebutuhan energi listrik di wilayah Malang (Pasman et al., 2010). Sedangkan Ogcu et al.(2012) membahas tentang peramalan konsumsi listrik menggunakan jaringan syaraf dan *support vector regression*. Selain itu terdapat juga yang membuat algoritma penjadwalan untuk mengurangi ledakan konsumsi listrik oleh Lu et al.(2014). Tiga penelitian diatas menjadi referensi untuk penelitian skripsi ini.

Berdasarkan permasalahan beserta solusi dari beberapa penelitian sebelumnya, dirasa belum optimal. Saat ini, banyak metode baru untuk melakukan optimasi yang dirasa mampu menyelesaikan permasalahan di atas. Salah satu algoritma yang dapat digunakan untuk optimasi adalah algoritma genetika. Oleh karena itu algoritma genetika dirasa dapat mengoptimasi masalah regresi di atas. Pada penelitian sebelumnya yang dilaksanakan di Universitas Muhammadiyah Malang, dilakukan optimasi titik pusat *cluster* pada *Fuzzy C-Means* menggunakan algoritma genetika untuk menentukan nilai akhir mahasiswa (Mas'udia, 2012). Penelitian tersebut menghasilkan bahwa *Genetic Fuzzy System* menghasilkan nilai fungsi objektif yang lebih kecil daripada *Fuzzy C-Means*. Dari beberapa proses tersebut diyakini Algoritma Genetika memiliki beberapa kelebihan dibandingkan algoritma yang lain dalam menghasilkan *output* yang optimal dan dapat dimanfaatkan untuk menyelesaikan permasalahan di atas.

1.2 Rumusan masalah

Berdasarkan latar belakang permasalahan di atas maka rumusan masalah yang dibuat adalah:

1. Bagaimana mengimplementasikan metode *Support Vector Regression* (SVR) untuk prakiraan kebutuhan energi listrik dengan optimasi algoritma genetika?
2. Berapa nilai *error rate* prakiraan yang dapat dihasilkan oleh *Support vector Regression* (SVR) yang dioptimasi menggunakan algoritma genetika?

1.3 Tujuan

Tujuan pembuatan skripsi ini adalah :

1. Mengimplementasikan metode *Support Vector Regression* (SVR) untuk prakiraan kebutuhan energi listrik dengan optimasi algoritma genetik.
2. Mengetahui nilai *error rate* prakiraan yang dapat dihasilkan oleh *Support vector Regression* (SVR) yang dioptimasi menggunakan algoritma genetika.

1.4 Manfaat

Manfaat dari pembuatan skripsi ini adalah:

1. Untuk membantu PLN Kota Malang dalam memperkirakan konsumsi listrik untuk bulan berikutnya.
2. Untuk membantu PLN Kota Malang dalam memberikan rekomendasi atau masukan sebelum melakukan pendistribusian listrik.
3. Untuk membantu pemerintah Kota Malang dalam menganalisis penggunaan daya listrik di Kota Malang agar lebih optimal.

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 dipakai adalah *Support Vector Regression* (SVR) dengan menggunakan kernel RBF
2. Optimasi yang dipakai adalah Algoritma Genetika yang dimulai dari proses *crossover*, mutasi, evaluasi dan seleksi.
3. Objek yang digunakan adalah daya listrik Kota Malang tahun 2010-2014.
4. Prakiraan yang dibuat nantinya dalam rentang waktu satu tahun kedepan.

1.6 Sistematika pembahasan

Sistematika penulisan tugas akhir ini dibagi menjadi enam bab dengan masing-masing bab diuraikan sebagai berikut :

1.6.1. BAB I PENDAHULUAN

Meliputi latar belakang masalah, rumusan masalah, batasan masalah, tujuan penelitian, manfaat penelitian, sistematika penulisan laporan serta alokasi penjadwalan penelitian.

1.6.2. BAB II LANDASAN KEPUSTAKAAN

Bab ini menjelaskan teori-teori, temuan dan bahan penelitian sebelumnya yang diperoleh dari berbagai referensi yang menunjang penelitian dalam penulisan skripsi. Adapun teori yang tercangkup dalam bab ini yaitu mengenai *support vector regression*, definisi dan konsep algoritma genetika dan mengenai konsumsi daya listrik.

1.6.3. BAB III METODOLOGI

Bab ini berisi metode atau langkah-langkah yang akan digunakan dalam penelitian skripsi yang terdiri dari studi literatur, penyusunan dasar teori, metode pengambilan data, analisa dan perancangan sistem, pengujian rancangan aplikasi perangkat lunak yang akan dibuat, pengambilan kesimpulan, hingga penulisan laporan.

1.6.4. BAB IV PERANCANGAN

Bab ini berisi analisis kebutuhan dan perancangan aplikasi yang dibuat, meliputi deskripsi aplikasi, spesifikasi kebutuhan, dan perancangan atau desain sistem *support vector regression* dalam Konsumsi daya Listrik di Kota Malang Menggunakan Algoritma Genetika yang nantinya akan diimplementasikan.

1.6.5. BAB V IMPLEMENTASI

Pada bagian ini akan berisi penjelasan tentang lingkungan implementasi misalnya bahasa pemrograman, sistem operasi serta perangkat keras yang digunakan. Akan ditunjukkan pula algoritma operasi yang akan diimplementasikan pada penelitian ini.

1.6.6. BAB VI PENGUJIAN DAN ANALISIS

Bab ini menjelaskan tentang strategi pengujian (unit, integrasi dan validasi) dan teknik pengujian terhadap sistem yang telah direalisasikan.

1.6.7. BAB VII PENUTUP

Bagian penutup berisi tentang pengambilan kesimpulan yang diambil berdasarkan analisa, meliputi hasil, kelebihan dan kekurangan, serta saran-saran untuk penyempurnaan aplikasi yang dibuat.

BAB 2 LANDASAN KEPUSTAKAAN

Landasan kepustakaan terdiri atas kajian pustaka dan dasar teori. Kajian Pustaka adalah pembahasan tentang penelitian yang sudah dilakukan dan yang akan diusulkan untuk penelitian berikutnya. Dasar teori adalah pembahasan tentang teori yang diperlukan untuk perencanaan aplikasi.

2.1 Kajian Pustaka

Topik prakiraan kebutuhan listrik Kota Malang telah dibahas pada penelitian sebelumnya. Pada penelitian berjudul "Analisis Implementasi Jaringan Syaraf Adaptif untuk Peramalan Kebutuhan Energi Listrik Wilayah Malang" telah dilakukan oleh Donny Frans Pasman. Penelitian tersebut bertujuan untuk meramalkan kebutuhan energi listrik tahun 2009. Penulis menggunakan algoritma jaringan saraf adaptif untuk peramalan (Pasman et al., 2010). Hasil penelitian itu menyimpulkan bahwa jaringan saraf adaptif lebih lebih cepat dan presisi yang lebih tinggi daripada algoritma jaringan saraf tiruan dengan error rata-rata sangat kecil yaitu 1%.

Sedangkan untuk metode *Support Vector Regression* (SVR), telah banyak diterapkan pada kasus prakiraan kebutuhan listrik. Salah satunya penelitian oleh Haijiang Wang dan Shanlin Yang yang berjudul "*Electricity Consumption Prediction Based on SVR with Ant Colony Optimization*" . Penelitian tersebut bertujuan untuk meramalkan konsumsi listrik provinsi Jiangsu menggunakan metode ACO-SVR. Peneliti menggunakan metode GA-SVR dikarenakan metode tersebut mampu untuk mengatasi tipe data konsumsi listrik yang bersifat non-linear (Wang, 2013). Kesimpulan penelitian tersebut menunjukkan bahwa *Support Vector Regression* dapat memecahkan permasalahan non-linear pada kasus konsumsi listrik.

Algoritma genetika telah banyak diimplementasikan untuk kasus optimasi maupun prakiraan. Salah satu penelitian yang menggunakan algoritma genetika yaitu "Optimasi Cluster Pada *Fuzzy C-Means* Menggunakan Algoritma Genetika Untuk Menentukan Nilai Akhir" karya Putri Elfa Mas`udia dan Retantyo Wardoyo. Peneliti menggunakan algoritma genetika bertujuan untuk mengoptimasi titik pusat *cluster* pada *Fuzzy C-Means* menggunakan algoritma untuk menentukan nilai akhir mahasiswa. Penulis menggunakan algoritma genetika dengan harapan dapat menangani masalah tersebut karena algoritma genetika berbasis evolusi dapat mencari individu terbaik melalui operasi genetika (seleksi, *crossover*, mutasi) dan dievaluasi berdasarkan nilai *fitness* (Mas'udia, 2012). Dengan menggunakan algoritma genetika, nilai awal titik pusat cluster *V* dibangkitkan secara acak, kemudian nilai *V* tersebut digunakan untuk menghitung matriks *u*. Selanjutnya nilai *V* ini akan dievolusikan menggunakan seleksi, *crossover* dan mutasi untuk mendapatkan nilai *V* yang paling optimum (Mas'udia, 2012).

Penelitian sebelumnya juga sudah ada yang menggunakan metode *Support Vector Regression* dan algoritma genetika (GA-SVR). Salah satunya ditulis oleh Jin-Cherng Lin, Chu-Ting Chang dan Sheng-Yu Huang yang berjudul "*Research on*

Software Effort Estimation Combined with Genetic Algorithm and Support Vector Regression". Penelitian tersebut bertujuan untuk meramalkan *software effort estimation,* karena kesalahan dalam perhitungan *software effort* dapat menyebabkan konsekuesi yang serius, salah satunya adalah pelonjakan biaya yang berakibat defisit yang besar. Oleh karena itu, peneliti tersebut menggunakan model GA-SVR. Hasil dari penelitian tersebut menunjukkan bahwa SVR dapat mengetahui model yang paling cocok, dan GA dapat menemukan parameter terbaik dengan cepat dan tepat dengan nilai *Mean Magnitude of Relative Error* (MMRE) sebesar 0.2 (Lin et al., 2011).

Pada penelitian ini, akan dilakukan algoritma genentika adalah untuk mengoptimasi parameter-parameter dalam *Support Vector Regression* (SVR) dalam konsumsi listrik, model SVR yang dihasilkan nantinya akan dibuat menjadi suatu persamaan untuk mencari prakiraan konsumsi listrik di bulan selanjutnya. Perbandingan objek dan metode penelitian yang akan dilakukan dengan penelitian dari masing-masing referensi studi literatur ditunjukkan pada Tabel 2.1.

Tabel 2.1 Perbedaan Penelitian Sebelumnya dengan Usulan Penulis

No	Judul	Objek	Metode	Keluaran
		Masukan dan parameter	**Proses**	**Hasil**
1	Analisis Implementasi Jaringan Syaraf Adaptif untuk Peramalan Kebutuhan Energi Listrik Wilayah Malang (Pasman et al., 2010)	Kebutuhan energi listrik	Jaringan Saraf Tiruan – Adaptif	Nilai *Mean Square Error* (MSE)
		-Data latih -Parameter ANN -Teknik normalisasi	-Normalisasi data awal -Penentuan bobot awal -Penentuan bobot awal -Pengecekan MSE -Penginputan data target -Mengidentifikasi data target -Melakukan adaptasi bobot	Hasil analisis peramalan kebutuhan energi listrik wilayah Malang menggunakan jaringan syaraf adaptif lebih baik daripada jaringan syaraf tiruan dengan MSE 0,1 dan selisih 0,3%
2	*Electricity Consumption Prediction Based on SVR with Ant Colony Optimization* (Wang, 2013)	Konsumsi listrik	ACO-SVR	Nilai *Mean Square Error* (MSE)
		-Data konsumsi listrik	-Menempatkan semut secara acak -Evaluasi posisi semut menggunakan *transition rule*	*Support Vector Regression* (SVR) dapat menyelesaikan permasalahan peramalan

			-Mengumpulkan himpunan - Mengevaluasi kembali posisi, jika belum sesuai maka dilakukan *update pheromone*, jika sudah sesuai maka menghasilkan item terbaik yang akan digunakan untuk proses *Support vector regression* untuk melakukan peramalan konsumsi listrik	konsumsi listrik yang bersifat non-linear dengan baik, hal itu dibuktikan dengan hasil yang akurasinya tinggi
3	Optimasi Cluster Pada *Fuzzy C-Means* Menggunakan Algoritma Genetika Untuk Menentukan Nilai Akhir (Mas'udia, 2012)	Nilai mahasiswa	*Fuzzy C-Means* dengan Algoritma Genetika	*Class* hasil dari proses *clustering* menggunakan *Genetic Fuzzy System* (GFS) dan *Fuzzy C-Means* (FCM)
		-Nilai angka, jumlah *cluster*, Pc, Pm	-Representasi kromosom -Menghitung matriks *U* -Menghitung *fitness* -Seleksi -*Crossover* -Mutasi -Evaluasi *V*, jika belum sesuai kembali ke proses hitung matrik *U* -Jika sudah sesuai maka ditemukan *V* optimum	Hasil menunjukkan bahwa dengan menggunakan algoritma genetika menghasilkan fungsi objektif yang lebih kecil dibandingkan dengan menggunakan *Fuzzy C-Means* untuk menentuan nilai akhir mahasiswa
4	Research on Software Effort Estimation	*Software*	*Support Vector Regression* dan algoritma genetika	*Mean Magnitude of Relative Error* (MMRE)

	Combined with Genetic Algorithm and Support Vector Regression (Lin et al., 2011)	- (*Constructive Cost Model*) COCOMO Parameter: - Tipe SVM - Tipe kernel - *Cost* - Gamma (*Ɣ*) - Epsilon (*ε*)	- Insialisasi parameter - Representasi kromosom - *Crossover* - Mutasi - Mengihitung SVR - Menghitung nilai *fitness* - Seleksi - Ulangi sampai mencapai nilai terbaik	Hasil dari penelitian tersebut menunjukkan bahwa SVR dapat mengetahui model yang paling cocok, dan GA dapat menemukan parameter terbaik dengan cepat dan tepat
5	Usulan penulis: Prakiraan Kebutuhan Energi Listrik Menggunakan *Support Vector Regression* (SVR) Dengan Optimasi Algoritma Genetika	Kebutuhan energi listrik	*Support Vector Regression* (SVR) dengan Algoritma genetika	Nilai *Mean Square Error* (MSE) dan *Mean Absolute Error* (MAE)
		-Data konsumsi listrik kota Malang Algoritma Genetika : - Popsize - *Crossover rate* - *Mutation rate* Generasi	-Inisalilasi kromosom secara random -Setiap kromosom dicari *error* serta *fitness*. -Proses Reproduksi dengan *extended intermediate crossover* dan *random mutation* -Proses Evaluasi dan Seleksi dengan model *elitism selection* -Diulang sebanyak generasi yang dibutuhkan sampai mendapatkan titik kromosom yang stabil dengan nilai *fitness* terbaik.	Penggunaan metode *Support Vector Regression* (SVR) yang dioptimasikan mengunakan algoritma genetika mampu menyelesaikan permasalahan prakiraan kebutuhan listrik Kota Malang

2.2 Daya Listrik

Daya adalah nilai dari energi listrik yang dikirimkan dan didistribusikan, dimana besarnya daya listrik tersebut sebanding dengan perkalian besarnya tegangan dan arus listriknya. Sistem suplai daya listrik dapat dikendalikan oleh kualitas dari tegangan, dan tidak dapat dikendalikan oleh arus listrik sebab arus listrik berada pada sisi beban yang bersifat individual, sehingga pada dasarnya kualitas daya merupakan kualitas dari tegangan itu sendiri.

Daya listrik merupakan suatu besarnya usaha yang dilakukan oleh sumber tegangan dalam 1 detik. Jika dalam waktu *t* sekon sumber tegangan telah melakukan usaha sebesar *W*. Daya listrik bisa juga disebut suatu kekuatan yang dikandung dalam aliran arus dan tegangan listrik melalui hambatan dengan besaran tertentu. Satuan ukuran daya listrik adalah Watt dan mempunyai simbol *P*. Maka persamaan untuk mencari besarnya daya adalah (Widodo, 2013) :

Persamaan (2.1) merupakan rumus mencari daya.

$$P = \frac{W}{t} \quad (2.1)$$

Persamaan (2-2) merupakan rumus mencari gaya/usaha.

Karena $W = V.I.t$ (2.2)

Persamaan (2-3) merupakan rumus formulasi untuk mencari daya berdasarkan persamaan (2-2).

Maka, $P = \frac{V\,I\,t}{t} = V\,.I$ (2.3)

Menurut hukum Ohm pada persamaan (2.4) :

$V = I\,.R$, sehingga (2.4)

Mencari daya berdasarkan persamaan (2.4) maka dihasilkan rumus beda potensial pada persamaan (2.5)

$P = I^2\,R$, dan karena $I = \frac{V^2}{R}$ (2.5)

Dimana :

P	= daya (joule/sekon) atau watt
W	= usaha (joule)
t	= waktu (detik)
1 joule/sekon	= 1 watt
V	= beda potensial (volt)
I	= kuat arus yang mengalir (ampere)

Misalnya, jika pada suatu peralatan listrik didapati tulisan : 220V 50 W, artinya bahwa alat tersebut dapat bekerja dengan stabil dan baik jika dipasang pada tegangan 220 V dan daya 50 watt sendiri dalam tiap sekon alat tersebut menggunakan listrik sebesar 50 joule (Widodo, 2013).

2.2.1 kWh Listrik

KWh bisa juga disebut kilowatt-jam atau *killowatt-hour* merupakan ukuran satuan energi listrik yang dikirim oleh peralatan elektronik yang terhubung dan membutuhan listrik dan diberi biaya. KWh adalah produk tenaga listrik dalam satuan kilowatt dikalikan dengan waktu dalam jam, bukan kilowatt per jam (Kw per hour).

Sedangkan untuk mengukur besarnya kWh pada sebuah alat elektronik bisa menggunakan kWh meter yang mengukur secara langsung hasil kali tegangan, arus faktor kerja, dikalikan waktu yang bekerja selama jangka waktu tersebut (Oktiyanti, 2011). KWh meter adalah kumparan tegangan, kumparan arus, piringan aluminium, magnet tetap yang bertugas menetralkan piringan aluminium dari induksi medan magnet dan gear mekanik yang mencatat jumlah perputaran piringan aluminium.

Gambar 2. 1 kWh Meter Analog

KWh meter analog bekerja menggunakan metode induksi medan magnet dimana medan magnet tersebut menggerakkan piringan yang terbuat dari aluminium. Putaran piringan tersebut akan menggerakkan *counter* digit sebagai tampilan jumlah kWhnya. Gambar kWh meter akan ditunjukkan pada Gambar 2.1.

2.2.2 Perhitungan Pemakaian Listrik Rumah Tangga

Alat ukur yang biasa disebut meteran yang dipakai sebagai pengukur energi listrik yang digunakan di rumah, diukur dengan satuan *killowatt-hour* atau kWh. Angka yang ditunjukkan oleh meteran itulah yang dipakai sebagai dasar pembayaran rekening listrik di rumah.

Karena satuan daya *P* adalah watt dan satuan waktu *t* adalah sekon, maka satuan usaha atau energi adalah *watt-second* atau WS. Satuan ini dianggap terlalu kecil dalam pengukuran pemakaian listrik sehari-hari. Oleh sebab itu, perlu ditentukan satuan yang lebih besar yaitu *watt-hour* atau Wh.

1 Wh = 3600s, pada umumnya meteran pada rumah penduduk menggunakan satuan kilowatt-hour (kWh) (Bali, 2000).

1 kWh = 1000 x 3600 Ws

1 kWh = 3,6 x 106 Ws

1 kWh = 3,6 x 106 joule

Sebagai contoh telah diamati posisi meteran bulan menunjukan 55.341 kWh dan posisi meteran bulan ini 55.439 kWh. Berarti selama sebulan telah memakai energi listrik sebesar 55.439 – 55.341 = 98 kWh.

Contoh perhitungan konsumsi listrik rumah tangga dalam waktu satu bulan (Kompas, 2008):

A. RUANG KELUARGA

- TV 21 inci. Konsumi daya 68 watt. Asumsi pemakaian 8 jam per hari.
- AC Split ½ PK. Konsumsi daya 430 watt. Asumsi pemakaian 8 jam per hari.
- Komputer. Konsumsi daya 140 watt. Asumsi pemakaian 5 jam per hari.
- Game Player. Konsumsi daya 20 watt. Asumsi pemakaian 5 jam per hari.
- Lampu Bohlam 60 Watt. Konsumsi daya 60 watt. Asumsi pemakaian 8 jam.
- Kipas Angin. Konsumsi daya 103 watt. Asumsi pemakaian 8 jam per hari.

B. RUANG DAPUR

- Kompor Listrik. Konsumsi daya 380 watt. Asumsi pemakaian 4 jam per hari.
- Magic Jar. Konsumsi daya: Menanak nasi 465 watt; Menghangatkan nasi 65 watt. Asumsi pemakaian 9 jam per hari (1 jam mananak nasi, 8 jam menghanagtkan nasi).
- Kulkas 120 Liter. Konsumsi daya: Compressor standby 12 watt; Compressor menyala 50 watt. Asumsi pemakaian 24 jam per hari (12 jam compressor standby, 12 jam compressor menyala).
- Setrika. Konsumsi daya 300 watt. Asumsi pemakaian 1 jam per hari.
- Dispenser. Konsumsi daya: Menyala 250 watt, standby 6 watt. Asumsi pemakaian 24 jam per hari (standby 22 jam, menyala 2 jam).

C. KAMAR MANDI

- Mesin Cuci. Konsumsi daya: Mencuci/membilas 250 watt; mengeringkan 300 watt. Asumsi pemakaian 1,5 jam per hari (mencuci dan membilas 2 jam, mengeringkan ½ jam).
- Pompa Air. Konsumsi daya 650 watt. Asumsi pemakaian 3 jam per hari.

Jawab :

- TV 21 inci
  68 watt x 8 jam x 30 hari = 16.320 Wh = 16,32 kWk
- AC Split ½ PK
  430 watt x 8 jam x 30 hari = 103.200 Wh = 103 kWh
- Komputer
  140 watt x 5 jam x 30 hari = 15.600 Wh = 15,6 kWh
- Game Player
  20 watt x 5 jam x 30 hari = 3.000 Wh = 3 kWh

- Lampu Bohlam 60 Watt
  60 watt x 8 jam x 30 hari = 14.400 Wh = 14,4 kWh
- Kipas Angin
  103 watt x 8 jam x 30 hari = 24,720 Wh = 24,72 kWh
- Kompor Listrik
  380 watt x 4 jam x 30 hari = 45.600 Wh = 45,6 kWh
- Magic Jar
  465 watt x 1 jam x 30 hari = 13.950 Wh = 13,95 kWh
  65 watt x 8 jam x 30 jam = 15.600 Wh = 15,6 kWh
- Kulkas 120 Liter
  12 watt x 12 jam x 30 hari = 4.320 Wh = 4,32 kWh
  50 watt x 12 jam x 30 hari = 18.000 Wh = 18 kWh
- Setrika
  300 watt x 1 jam x 30 hari = 9.000 Wh = 9 kWh
- Dispenser
  250 watt x 2 jam x 30 hari = 15.000 Wh = 15 kWh
  6 watt x 22 jam x 30 hari = 3.960 Wh = 1,96 kWh
- Mesin Cuci
  250 watt x 2 jam x 30 hari = 15.000 Wh = 15 kWh
  300 watt x ½ jam x 30 hari = 4.500 Wh = 4,5 kWh
- Pompa Air
  650 watt x 3 jam x 30 hari = 58.500 Wh = 58,5 kWh

Tarif listrik per kWh (PLN, 2015) = Rp 1.547,-

Bea Beban tetap per bulan = Rp 3500,-

Maka jumlah energi yang harus dibayar adalah :

= (16,32+103+15,6+3+14,4+24,72+45,6+13,95+15,6+4,32+18+9+15+1,96+15 +4,5+58,5) kWh

= 378,47 kWh

Jadi rekening yang harus dibayar adalah :

= Rp 3500,- + (378,47 x Rp 1.547,-)

= Rp 3500,- + Rp 585.493,-

= Rp 588.993,-.

2.3 Normalisasi dan Denormalisasi

Sebelum melakukan pelatihan dalam sistem *Radial Basis Function* data *input* dan data uji akan di normalisasi. Normalisasi ini bertujuan untuk mendapatkan data dengan ukuran yang lebih kecil yang mewakili data yang asli tanpa kehilangan karakteristik sendirinya. Rumus dari normalisasi yaitu (Patro, 2015):

$$Normalisasi = \frac{x - \min}{maks - \min} \quad \text{.....(2.6)}$$

X = data
Min = data minimum
Max = data maksimum

Sedangkan denormalisasi adalah mengembalikan ukuran data yang telah dinormalisasi sebelumnya untuk mendapatkan data yang asli. Denormalisasi dilakukan pada hasil keluaran dari pelatihan berupa prakiraan konsumsi listrik. Adapun rumus dari denormalisasi yaitu sebagai berikut :

$$Denormalisasi = y(maks - \min) + \min \quad \text{.....(2.7)}$$

Y = hasil keluaran dari pelatihan
Min = data minimum
Max = data maximum

2.4 *Support Vector Regression* (SVR)

2.4.1 Kernel RBF

Secara umum, kernel *Radial Basic Function* (RBF) paling sering digunakan dalam *Support Vector Regression* (SVR). Kernel ini memetakan data non-linear ke dimensi yang lebih tinggi, tidak seperti kernel linier, kernel RBF dapat menangani kasus ketika relasi antara kelas label dan atribut non linier. Selanjutnya, kernel linier termasuk *case* spesial karena kernel linier dengan parameter $\tilde{C}$ memiliki kinerja yang sama seperti kernel RBF dengan beberapa parameter (C, γ) (Hsu et al., 2010). Parameter tersebut digunakan untuk mentransformasi data *training* ke *feature space*.

Alasan kedua karena jumlah *hyperparameter* yang mempengaruhi kompleksitas pemilihan model. Kernel polinominal memiliki *hyperparameter* dibanding dengan kernel RBF.

Rumus untuk kernel RBF sebagai berikut (Kuo et al., 2014):

$$K(x, x') = Exp(-\|x - x'\|^2 / 2\sigma^2) \quad (2.8)$$

dimana $\|x - x'\|^2; x, x' \in R^d$ adalah *euclidean distance* antara dua data *trainning* dan σ adalah nilai varian data yang sudah diinisialisasi sebelumproses pelatihan SVR.

2.4.2 Persamaan *Support Vector Regression* (SVR)

SVR merupakan penerapan *support vector machine* (SVM) untuk kasus regresi. Dalam kasus regresi *output* berupa bilangan riil atau kontinyu. SVR merupakan metode yang dapat mengatasi *overfitting*, sehingga akan menghasilkan performansi yang bagus (Smola dan Scholkopf, 2004).

2.4.2.1 Formula Alternatif untuk Bias

Nilai bias dapat diformulasikan ke dalam bentuk lain yaitu dengan menggunakan penambahan dimensi untuk vektor masukan tambahan didefinisikan sebagai berikut (Vijayakumar, 1999).

$$f(x) = \sum_{i=1}^{l} (\alpha_i^* - \alpha_i)(K(x_i, x) + \lambda^2). \quad (2.9)$$

α_i, α_i^* =merupakan suatu nilai formulasi.
K = merupakan nilai kernel
λ = *augmenting factor*
x_i = merupakan data ke-*i*
x = merupakan data

2.4.2.2 Algoritma Sequential 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* pada baris ke-*i* dan kolom ke-*j*.
E_i merupakan nilai *error* ke-*i*.
y_i merupakan nilai aktual data latih.
α_i, α_i^* merupakan suatu nilai formulasi.
C merupakan nilai kompleksitas.
ε merupakan nilai deviasi
λ *augmenting factor*
x_i merupakan data ke-i
x merupakan data
γ merupakan *learning rate* yang didapatkan dari

$$\frac{cLR}{\max(\text{diagonal matriks Rij})} \quad (2.10)$$

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

1. Inisialisasi $\alpha_i = 0$ dan $\alpha_i^* = 0$. Menghitung

   $$Rij = K(x_i, x_j) + \lambda^2 \quad (2.11)$$

   Untuk $i, j = 1, \ldots, l$

2. Untuk setiap data latih, $i = 1\ to\ l$, hitung:

   2.1. $E_i = y_i - \sum_{j=1}^{l} (\alpha_i^* - \alpha_i) R_{ij}$

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

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

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

   $\alpha_i = \alpha_i + \delta\alpha_i \quad (2.12)$

3. Kondisi berhenti jika sudah mencapai iterasi maksimum atau $\max(|\delta\alpha_i|) < \varepsilon$ dan $\max(|\delta\alpha_i *|) < \varepsilon$. Jika belum, kembali ke step 2.

4. Selesai

2.4.3 *Mean Absolute Percentage Error* (MAPE)

Mean Absolute Percentage Error merupakan metode analisa untuk membandingkan hasil nilai prakiraan dengan nilai asli. Semakin kecil nilai error maka semakin kecil pula perbedaan nilai prakiraan dan nilai asli. MAPE sejauh ini telah digunakan untuk mengevaluasi sebuah model (Wu et al, 2009; Ahmad et al, 2011). Berikut persamaan *Mean Absolute Percentage Error* (MSE):

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{y_i - y'_i}{y_i}\right| * 100\% \quad (2.13)$$

Dimana:

n = Jumlah konsumsi yang diprakirakan

y_i = Nilai data latih/asli

y'_i = Nilai hasil prakiraan

i = indeks

2.5 Algoritma Genetika

Pada pertengahan 1970-an, John Holland adalah orang pertama yang ketat menyajikan konsep utama Algoritma Gentika (Ahmed et al., 2012). Algoritma genetika adalah metode ilmiah heuristik yang didasarkan pada teori evolusi biologis Darwin (Jiang et al., 2015; Ahmed et al., 2012), yang telah banyak diterapkan untuk memecahkan masalah optimasi parameter dalam bidang teknik dan ilmu pengetahuan, seperti konstruksi bangunan dan bioteknologi (Kucharzyka et al., 2012). Menurut Lin (2015), algoritma genetika adalah metode untuk memecahkan masalah optimasi baik dibatasi dan tidak dibatasi yang didasarkan pada seleksi alam, proses yang mendorong evolusi biologis. Algoritma genetika berulang kali memodifikasi sebuah populasi dari solusi individual . Algoritma genetika memilih individu secara acak dari populasi saat ini untuk menjadi orang tua dan menggunakan mereka untuk menghasilkan anak-anak untuk generasi berikutnya. Selama generasi-generasi, populasi "berkembang" menuju solusi yang optimal. Algoritma genetika dapat diterapkan untuk memecahkan berbagai masalah optimasi yang tidak cocok menggunakan algoritma optimasi standar, termasuk masalah di mana fungsi objektif todak kontinyu, tidak terdiferensiasi, *stochastic*, atau non linier.

Algoritma genetika dimulai dengan menentukan populasi awal *P* yang mengandung jumlah tertentu dari solusi dikodekan dengan benar atau sering disebut kromosom (Cichenski et al., 2015). Algoritma genetika menggunakan tiga tipe aturan utama yang digunakan untuk membentuk generasi selanjutnya dadi populasi saat ini (Lin et al., 2015), yaitu:

1. *Selection*: memilih individu yang disebut *parents* yang menyumbang populasi pada generasi berikutnya.

2. *Crossover*: menggabungkan dua *parents* untuk membentuk *children* untuk generasi berikutnya.
3. Mutasi: menerapkan perubahan secara random kepada *parent* untuk membentuk *children*.

Nilai *fitness* suatu individu menggunakan ukuran seberapa bagus solusi yang direoresentasikan oleh individu tersebut. Nilai *fitness* yang semakin tinggi menunjukkan semakin bagus solusi yang diberikan oleh individu tersebut (Sbeity et al., 2014).

2.5.1 Representasi Kromosom

Langkah awal dalam penerapan algoritma genetika adalah merepresentasikan permasalahan menjadi bentuk kromosom, yakni biasa disebut pengkodean. Penentuan jenis pengkodean yang digunakan bergantung pada bentuk permasalahan yang akan diselesaikan. Macam-macam pengkodean dalam algoritma genetika yaitu (Malhotra et al., 2011):

1. *Binary encoding*

   Binary encoding adalah tipe pengkodean yang paling umum dimana nilai data berbentuk biner. *Binarry e*ncoding memberikan banyak kemungkinan kromosom dengan jumlah alel yang kecil. Representasi kromosom dalam *binarry encoding* ditunjukkan dalam Gambar 2.2

Kromosom 1	1	0	1	0	1	0	0	1
Kromosom 2	1	1	0	0	1	0	1	1

Gambar 2. 2 *Binarry encoding*
Sumber: (Malhotra et al., 2011)

2. *Permutation encoding*

   Permutation encoding paling cocok untuk permasalahan antrian. Salah satu contoh penerapan permutation encoding adalah pada algoritma *Travelling Salaesman Problem* (TSP). Di dalam *permutation encoding*, setiap kromosom adalah angka yang disusun berurutan seperti pada Gambar 2.3

Kromosom 1	3	4	2	7	1	5	6	8
Kromosom 2	8	3	6	1	2	7	4	5

Gambar 2. 3 *Permutation encoding*
Sumber: (Malhotra et al., 2011)

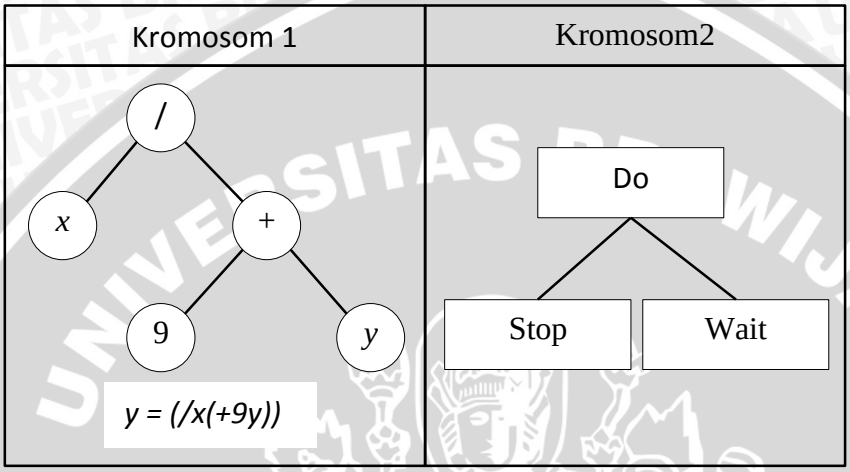
3. *Value encoding*

   Value encoding dapat berbentuk angka, bilangan real maupun karakter unik. *Value encoding* merupakan teknik dimana setiap kromosom berupa nilai dan digunakan bilamana nilai tersebut lebih kompleks dari yang dibutuhkan. Contoh *value encoding* terdapat pada Gambar 2.4

Kromosom 1	2,3123	4,7699	3,2123	5,3535
Kromosom 2	North	South	East	West

Gambar 2. 4 *Value encoding*
Sumber: (Malhotra et al., 2011)

4. *Tree e*ncoding



Gambar 2. 5 *Tree encoding*
Sumber: (Malhotra et al., 2011)

Teknik *tree encoding* paling cocok digunakan untuk pemngembangan program seperti *genetic programming*. Di dalam tree encoding, setiap kromosom adalah *tree* dari objek, fungsi atau perintah dalam *genetic programming*. *Locator/identifier separation protocol* (LISP) adalah bahasa pemrograman yang digunakan untuk encoding ini. Program LISP dapat ditunjukkan dalam bentuk struktur *tree* untuk *crossover* dan mutasi. Dalam *tree encoding*, kromosom ditunjukkan dalam Gambar 2.5.

2.5.2 *Fitness*

Fungsi *fitness* digunakan untuk proses evaluasi kromosom agar memperoleh kromosom yang diinginkan. Fungsi ini membedakan kualitas dari kromosom untuk mengetahui seberapa baik kromosom yang dihasilkan. Fungsi *fitness* tersebut ditungjukkan pada persamaan (2.19) (Suyanto, 2014).

$$fitness = \frac{100}{1+penalty} \quad (2.14)$$

Dari persamaan (2.19) nilai *fitness* ditentukan oleh nilai penalti. Penalti tersebut menunjukkan jumlah pelanggaran kendala pada suatu kromosom. Semakin tinggi nilai *fitness* akan semakin besar kemungkinan kromosom tersebut terpilih ke generasi berikutnya. Jadi nilai penalti berbanding terbalik dengan nilai fitness, semakin kecil penalti (jumlah pelanggaran) semakin besar nilai *fitness*-nya seperti ditunjukkan persamaan (2.20)

$$fitness = \frac{1}{1+\sum Bp+\sum Np} \quad (2.15)$$

Keterangan:

Bp = Bobot pelanggaran

Np = Indikator pelanggaran

2.5.3 Seleksi

Tahap seleksi menentukan individu yang dipilih untuk kawin (reproduksi) dan berapa banyak keturunan masing-masing individu yang dihasilkan. Prinsip utama dari strategi seleksi adalah "lebih baik individu, maka lebih tinggi kesempatan untuk menjadi orang tua". Ini adalah proses yang menentukan solusi yang harus dipertahankan dan diperbolehkan untuk mereproduksi dan mana yang pantas untuk mati. Tujuan utama dari seleksi adalah untuk menekankan pada solusi yang baik dan menghilangkan solusi yang buruk dalam suatu populasi, sekaligus mempertahankan ukuran populasi konstan. (Shukla et al., 2015).

A. Elitism Selection

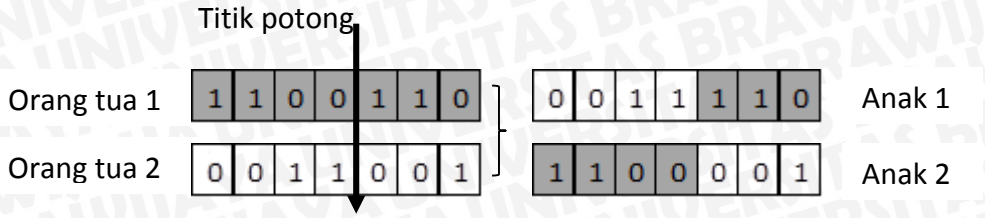
Karena seleksi dilakukan secara acak, maka tidak ada jaminan bahwa suatu individu bernilai *fitness* tertinggi akan selalu terpilih. Kalaupun individu bernilai *fitness* tertinggi terpilih, mungkin saja individu tersebut akan rusak karena proses pindah silang. Untuk menjaga agar individu bernilai *fitness* tertinggi tersebut tidak hilang selama evolusi, perlu dibuat satu atau dua salinannya. Prosedur ini dikenal dengan *elitism*. Metode elitism selection dilakukan dengan mengumpulkan semua individu baik parent maupun offspring, kemudian barulah diseleksi dengan melihat nilai fitness tertingginya untuk dipertahankan ke generasi selanjutnya. Metode seleksi elitism menjamin individu yang terbaik akan selalu lolos (Suyanto, 2014 dan Pramesti et al, 2015).

2.5.4 *Crossover*

Crossover atau sering disebut pindah silang merupakan proses perkawinan dua kromosom *parent* menjadi kromosom baaru (*offspring*). Tidak semua kromosom mengalami persilangan. Jumlah kromosom dalam populasi yang mengalami persilangan ditentukan oleh parameter yang disebut dengan *crossover rate* (Suyanto, 2014). Jenis operator persilangan antara lain:

A. One point crossover

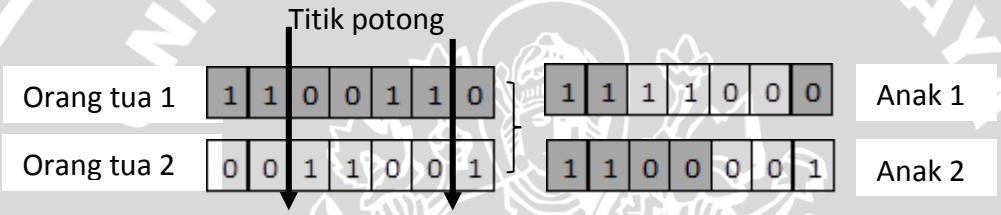
Sebuah titik *crossover* dipilih, selanjutnya string biner mulai dari awal kromosom sampai dengan titik tersebut disalin dari salah satu orangtua ke keturunannya, kemudian sisa bit keturunan disalin dari orangtua yang kedua (Suyanto, 2014).



Gambar 2. 6 Contoh pindah silang satu titik
Sumber: (Suyanto, 2014)

B. Two point crossover

Dua titik *crossover* dipilih, selanjutnya string biner mulai dari awal kromosom sampai dengan titik *crossover* pertama disalin dari salah satu orangtua ke keturunannya kemudian mulai dari titik *crossover* pertama sampai dengan titik kedua disalin dari orangtua kedua (Suyanto, 2014). Sisanya disalin dari orangtua pertama seperti pada Gambar 2.9

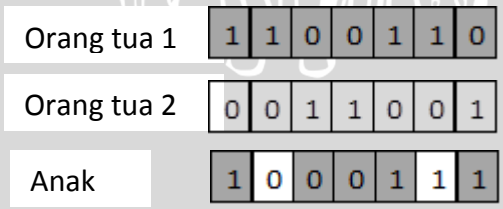


Gambar 2. 7 Contoh pindah silang dua titik
Sumber: (Suyanto, 2014)

C. Position-based crossover

Prosedur *position-based cossover*:

1. Pilih posisi gen dari induk satu secara random
2. Buat keturunan dengan menyalin nilai gen induk 1 yang menglami persilangan
3. Tempatkan gen induk 2 ke dalam gen keturunan yang masih kosong



Gambar 2. 8 Contoh pindah silang *position based crossover*
Sumber: (Suyanto, 2014)

D. Extended Intermediate Crossover

Extended intermediate crossover adalh asalah satu metode *crossover* yang menghasilkan offspring dari kombinasi nilai dua induk. Kombinasi dua induk(parent) untuk menghasilkan offspring tersebut dipilih secara acak dari populasi. Misalkan P1 dan P2 adalah dua induk yang telah diseleksi untuk

melakukan crossover, maka offspring C1 dan C2 bisa dibangkitkan sebagai berikut (Pramesti, 2015):

$$C1 = P1 + \alpha(P2 - P1)$$
$$C2 = P2 + \alpha(P1 - P2) \quad (2.16)$$

α dipilih dibangkitkan secara random misalkan dengan range [0,1].

2.5.5 Mutasi

Proses reproduksi dilanjutkan dengan mutasi. Mutasi adalah proses merubah gen dari keturunan secara random. Mutasi diperlukan untuk mengembalikan informasi bit yang hilang akibat *crossover*. Jika mutasi dilakukan terlalu sering, maka akan menghasilkan individu yang lemah karena konfigurasi gen pada individu yang unggul akan rusak. Berdasarkan bagian yang termutasi, proses mutasi dapat dibedakan atas tiga bagian:

- Mutasi pada tingkat kromosom, semua gen dalam kromosom berubah

Gambar 2. 9 Contoh mutasi tingkat kromosom.
Semua gen dalam kromosom berubah
Sumber: (Suyanto, 2014)

- Mutasi pada tingkat gen, semua bit dalam satu gen berubah. Misal 2 gen yang mengalami mutasi

Gambar 2. 10 Contoh mutasi tingkat gen.
Semua bit dalam suatu gen berubah
Sumber: (Suyanto, 2014)

- Mutasi pada tingkat bit, hanya satu bit yang berubah

Gambar 2. 11 Contoh mutasi tingkat bit.
Hanya satu bit yang berubah
Sumber: (Suyanto, 2014)

A. Random Mutation

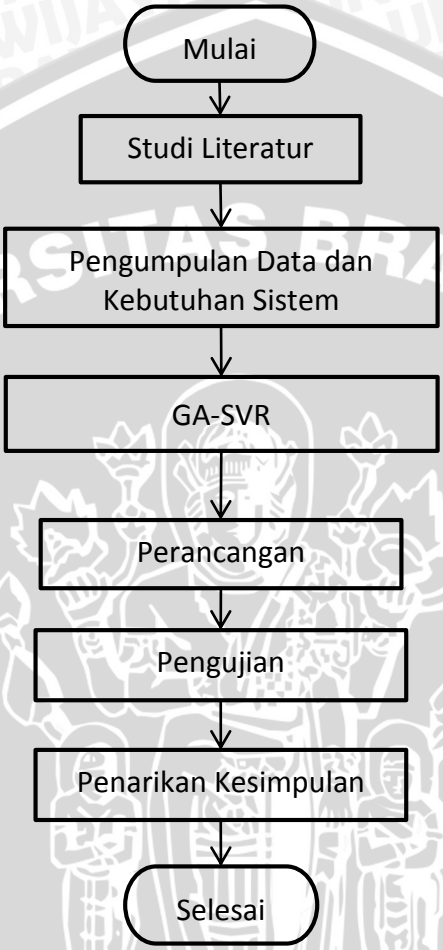
Metode *random mutation* adalah metode mutasi yang menghasilkan offspring (anak) yang dilakukan dengan menambah atau mengurangi nilai gen terpilih dengan bilangan random yang kecil. Mutasi dilakukan dengan memilih satu induk secara acak dari populasi. Misalkan domain variabel xj adalah [minj, maxj] dan offspring yang dihasilkan adalah $C=[\ x'_1..x'_n\ ]$, maka nilai gen offspring bisa dibangkitkan sebagai berikut (Pramesti, 2015):

$$x'_i = x'_i + r(\max_i - \min_j) \quad (2.17)$$

range *r* dibangkitkan secara random misalkan dengan range [-0,1, 0,1].

BAB 3 METODOLOGI

Metodologi penelitian menjelaskan metode yang digunakan dalam pembuatan sistem prakiraan kebutuhan energi listrik menggunakan *Support Vector Regression* (SVR) dengan optimasi algoritma genetika. Tahapan metodologi penelitian dapat dilihat pada gambar 3.1



Gambar 3.1 Diagram Alir Metode Penelitian

3.1 Studi Literatur

Studi literatur mempelajari mengenai dasar teori yang digunakan untuk menunjang penulisan serta pengerjaan tugas akhir. Teori-teori pendukung penulisan serta pemahaman tentang tugas akhir diperoleh dari buku, jurnal, e-book dan penelitian sebelumnya yang berkaitan tentang topik tugas akhir ini. Referensi utama yang diperlukan untuk menunjang penulisan ini diantaranya:

1. Metode *Support Vector Regression* (SVR)
2. Algoritma Genetika
3. Rekayasa Perangkat Lunak
4. Pemrograman menggunakan bahasa Java
5. DBMS MySQL

6. Proses pengujian sistem
7. Berbagai macam konsumsi listrik dan biayanya

3.2 Pengumpulan Data dan Kebutuhan Sistem

Analisa kebutuhan sistem merupakan tahap menganalisis hal-hal yang dibutuhkan untuk menjalankan sistem, sehingga sistem dapat berjalan secara optimal.

3.2.1 Pengumpulan Data

Data yang digunakan dalam penelitian ini adalah:

1. Data yang digunakan adalah data pemakaian kWh PLN Rayon Kota Malang pertahun 2011-2014, yang diperoleh dari data real pengusahaan PLN Rayon Kota Malang (terlampir).
2. Parameter Algoritma Genetika yang diperlukan berupa jumlah populasi (*popSize*), jumlah generasi, *crossover rate* (*cr*), dan *mutation rate* (*mr*).

3.2.2 Spesifikasi Kebutuhan Sistem

Analisis kebutuhan bertujuan untuk mendapatkan kebutuhan yang diperlukan untuk sistem prakiraan kebutuhan energi listrik menggunakan *Support Vector Regressio*n (SVR) dengan optimasi algoritma genetika pada PLN Kota Malang. Analisis kebutuhan diterapkan sesuai dengan lokasi penelitian, variabel penelitian dan mempersiapkan kebutuhan penelitian.

Spesifikasi lingkungan implementasi yang digunakan dalam pembuatan sistem prakiraan kebutuhan energi listrik menggunakan *Support Vector Regressio*n (SVR) dengan optimasi algoritma genetika diantaranya :

1. Spesifikasi Kebutuhan *Hardware*

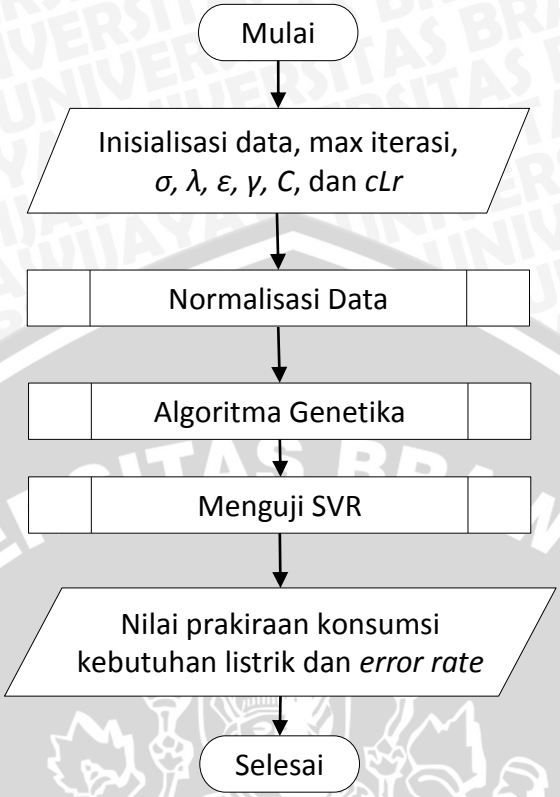
- Laptop Samsung NP355V4X-A02ID AMD A6
- RAM 2 GB
- DDR3
- VGA AMD RADEON HD GRAPHIC

2. Spesifikasi Kebutuhan *Software*

- Microsoft windows 8.1 sebagai sistem operasi yang digunakan
- Microsoft Office 2013 sebagai pengolah dokumentasi dan manual perhitungan
- Java merupakan bahasa pemrograman untuk menjalankan aplikasi developer

3.3 GA-SVR

Pada bagian ini awalnya akan dibahas proses perancangan sistem untuk memperoleh solusi keluaran yang baik. Langkah awal proses ini adalah penentuan parameter awal dan memasukkan data yang akan digunakan.

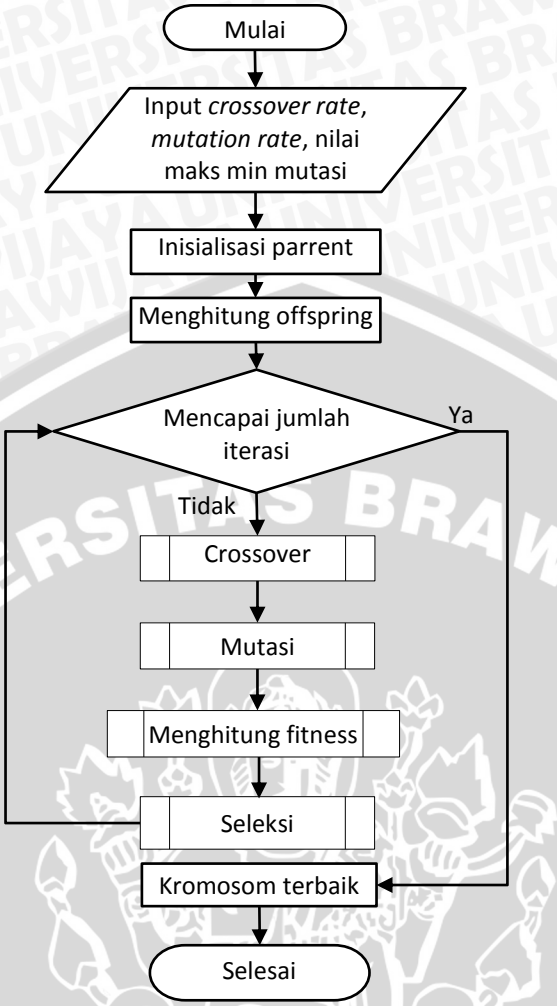


Gambar 3.2 Diagram Alir Optimasi GA-SVR

Langkah selanjutnya adalah melakukan normalisasi data. Kemudian melakukan proses GA-SVR sesuai dengan parameter yang telah didefinisikan dan data yang telah dinormalisasi. Rancangan sistem optimasi SVR dengan algoritma evolusi digambarkan pada diagram alir pada Gambar 3.12.

3.3.1 Proses Pencarian Parameter Terbaik Menggunakan Algoritma Genetika

Sebelum melakukan proses ini, sistem mengambil hasil pengkodean partikel yaitu nilai *C, ε, λ dan cLR*. Langkah selanjutnya adalah memilih fitur yang akan digunakan berdasarkan fitur yang dipilih menggunakan optimasi algoritma genetika. Proses ini dilakukan sebanyak *K* kali. Secara garis besar dapat dilihat pada Gambar 3.13 berikut:



Gambar 3.3 Diagram Alir Proses GA

3.3.1.1 Inisialisasi *Parent*

Kromosom tersusun atas bilangan real yang menyatakan parameter-parameter dalam perhitungan SVR. Panjang kromosom dalam 1 individu sesuai dengan parameter yang dibutuhkan dalam SVR yaitu 5. Pada Tabel 3.8 index ke-0 menyatakan lambda (λ), index ke-1 merupakan epsilon (ε), indeks ke-2 merupakan C dan indeks terakhir yaitu indeks ke-3 merupakan konstanta gama (*cLR*). Detail pembentukan kromosom akan ditampilkan pada Tabel 3.12.

Tabel 3.1 Pembentukan Kromosom

5	0,00000001	5	0,1

Pada Tabel 3.12 sel pertama adalah nilai parameter *lambda* (λ), sel kedua menunjukkan parameter *epsilon* (ε), sel ketiga adalah parameter *C*, dan sel terakhir menunjukkan parameter *cLr*.

Proses selanjutnya adalah inisialisasi populasi awal. Insialisasi populasi awal merupakan pembentukan kromosom-kromosom sebanyak jumlah populasi yang telah ditentukan pada awal pembuatan paramater. Kromosom dibangkitkan secara *random* pada rentang pada Tabel 3.14.

Tabel 3. 2 Batas Minimum dan Maksimum Kromosom

	λ	*E*	*C*	*cLR*
Minimum	1	0,000000001	1	0,01
Maksimum	10	0,009	50	12

Ukuran populasi atau sering disebut dengan *popSize* pada permasalahan kali ini dibuat sebanyak 5 populasi Representasi kromosom awal akan ditunjukkan pada Tabel 3.15.

Tabel 3.3 Data Populasi Awal

No	*λ*	*E*	*C*	*cLR*
P1	4	0,0000001	4	0,03
P2	5	0,00000001	5	0,1
P3	8	0,00001	1	0,006
P4	2	0,000000001	7	2
P5	10	0,0000001	8	12

3.3.1.2 Crossover

Pada kasus ini karena representasi kromosom bertipe data real, maka metode *crossover* yang dipilih adalah metode *extended intermediate crossover*, yaitu dengan memilih dua induk secara random dari populasi yang telah dibentuk dan nantinya akan menghasilkan anak (*offspring*) dari kombinasi kedua induk tersebut.

Populasi awal sebanyak 5 individu dengan *crossover rate* (*cr*) sebesar 0,4 maka offspring yang dihasilkan adalah 0,4 x 5 = 2. Akan dijelaskan proses *crossover* dalam diagram alir *flowchart* pada Gambar 3.12.

Proses *extended intermediate exchange* akan dijabarkan pada perhitungan manual sebagai berikut :

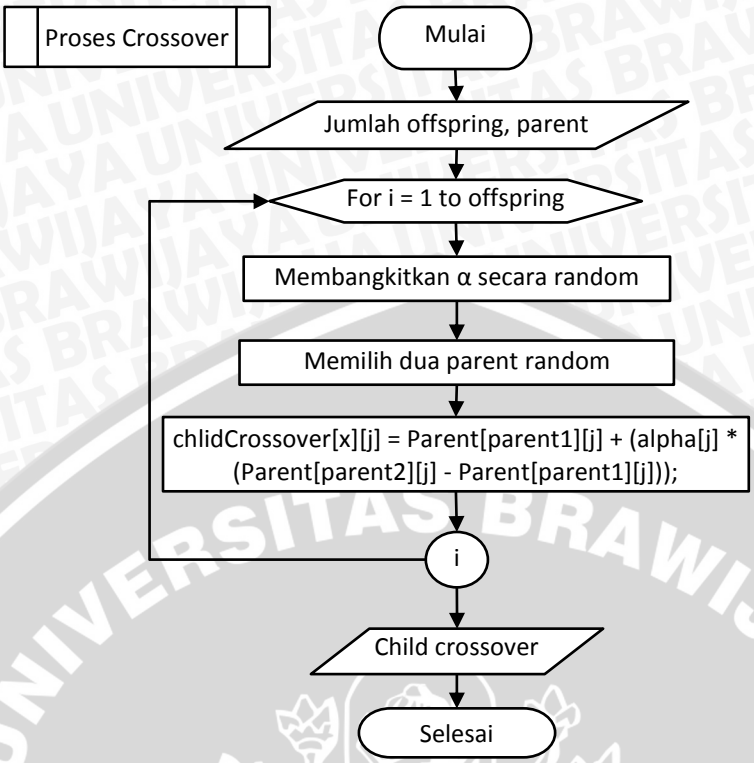
1. *α* dibangkitkan secara *random* pada interval [0,1]

α1	α2	α3	α4
0,1649848	0,157545219	0,125165	0,35305

P3	8	0,00001	1	0,006
P4	2	0,000000001	7	2
C1	7,010091	8,42471E-06	1,750991	0,709981
C2	2,989909	1,57629E-06	6,249009	1,296019

Gambar 3.4 Hasil *Crossover*

Nilai *α* pada index/kolom pertama digunakan untuk menghitung semua anak hanya pada index/kolom pertama. Hasil anak dari proses perhitungan *crossover* didapatkan dari rumus *Extended Intermediate Crossover*.



Gambar 3.5 *Flowchart* Proses *Crossover*

C1 dan C2 :

α = 0,1649848

C_1 = $P_1 + \alpha (P_2 - P_1)$

= 8 + 0,1649848 (2-8)

= 7,010091

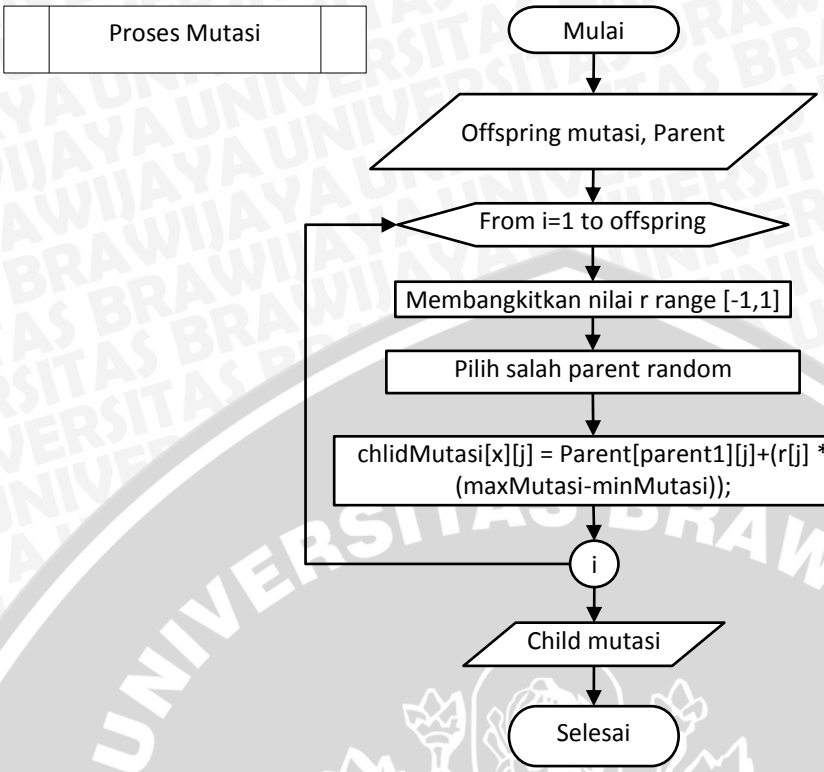
C_2 = $P_2 + \alpha (P_1 - P_2)$

= 2 + 0,1649848 (8-2)

= 2,989909

3.3.1.3 Mutasi

Proses mutasi dilakukan dengan memilih satu induk secara *random* dari populasi. Karena representasi kromosom bertipe data real, maka metode mutasi yang digunakan adalah *random mutation* yang *dilakukan* dengan menambah atau mengurangi nilai gen terpilih dengan bilangan random yang kecil. Dengan mutation rate sebesar 0,4 maka banyaknya *offspring* yang dihasilkan adalah 0,4 x 5 = 2. Parent dipilih secara *random*, pada tiap kromosom yang mengalami mutasi, setiap gen yang terpilih akan mengalami mutasi. Misalkan domain variabel x_j adalah [min_j,max_j] dengan *range r* dibangkitkan pada interval [-1 , 1]. Berikut flowchart untuk melakukan proses mutasi akan ditunjukkan pada Gambar 3.15.



Gambar 3.6 Diagram Alir Proses Mutasi

Proses *random mutation* akan dijabarkan dalam perhitungan manual sebagai berikut:

Dimana :

max = 0,6

min = -0,6

Nilai maksimum dan minimum ditentukan pada rentang kromosom awal yang telah dibuat, karena representasi kromosom dibangkitakan dalam rentang [-0.6, 0.6] maka nilai maksimumnya adalah -0.6 dan nilai minimumnya adalah 0.6.

Proses mutasi akan ditunjukkan pada Gambar 3.16.

r1	r2	r3	r4
-0,95835	-0,246777	-0,42306	0,401344

P2	5	0,00000001	5	0,1
C3	3,849980	0,000000001	4,492332	0,581613

P5	10	0,0000001	8	12
C4	8,84998	0,000000001	7,492332	12,48161

Gambar 3.7 Hasil Mutasi

Perhitungan proses mutasi dimulai dengan memilih salah satu gen pada induk secara random. Pada gen kedua C3 C4 sebelumnya bernilai negativ, namun sesuai pembatasan pada tabel 3.14, jika nilai lebih kecil dari batas bawah maka akan digantikan oleh nilai batas bawah gen tersebut. Pada proses mutasi di atas gen

pertama terpilih secara acak dan menghasilkan C3 dan C4 yang didapatkan dengan cara :

C3 :

x'_i $= x'_i + r\,(max_i - min_j)$
$= 5 + -0{,}95835\ (0{,}6 - (-0{,}6))$
$= 3{,}849980$

C4 :

x'_i $= x'_i + r\,(max_i - min_j)$
$= 30{,}3916 + 0{,}4503(100 - (-100))$
$= 120{,}454$

3.3.1.4 Menghitung *Fitness*

Proses evaluasi merupakan proses untuk menghitung nilai *fitness* dari setiap *chromosom* yang terbentuk, proses evaluasi di mencakup individu induk (*parent*) serta *offspring* yang terbentuk dari proses reproduksi sebelumnya. *Chromosome* yang terpilih untuk menjadi solusi berikunya harus mempunyai nilai *fitness* yang lebih baik, semakin besar nilai *fitness* yang diperoleh maka semakin baik pula untuk dijadikan solusi pada generasi berikutnya.

Sebelum menghitung nilai fitness, maka sebelumnya diperlukan menghitung *error rate*. Error rate disini didapatkan dari proses pada sub bab **3.3.2**. Sebagai contoh, P2 digunakan sebagai parameter untuk SVR, maka akan mengembalikan nilai error rate sebesar 2,366398.

P2	5	0,00000001	5	0,1

Selanjutnya menghitung nilai *fitness*. Pada kasus error yang digunakan adalah MAPE pada perhitungan sebelumnya. Rumus perhitungan *fitness* yaitu (Alves, 2013):

$$fitness = \frac{100}{1 + error}$$

$$= \frac{100}{1 + 2{,}366398}$$

$$= 29{,}7053$$

Hasil perhitungan *fitness* dari pembentukan kromosom sebelumnya dapat dilihat pada Tabel 3.13.

Tabel 3.4 Hasil Perhitungan *Fitness*

5	0,00000001	5	0,1	29,7053

Hasil evaluasi semua individu (*Parent* dan *Offspring*) akan ditampilkan pada tabel 3.16.

Tabel 3.5 Hasil Evaluasi

PARENT	GENERASI BARU				FITNESS (1/1+error)
	λ	ϵ	C	cLR	
P1	4	0,0000001	4	0,03	8,3902923
P2	5	0,00000001	5	0,1	29,7053
P3	8	0,00001	1	0,006	2,9911649
P4	2	0,000000001	7	2	5,72589E-03
P5	10	0,0000001	8	12	2,43795E-04
C1	7,01009104	8,42471E-06	1,750991	0,7099808	2,24404E-03
C2	2,98990896	1,57629E-06	6,249009	1,2960192	0,1187934
C3	3,849980	0,000000001	4,492332	0,581613	2,78347E-03
C4	8,84998	0,000000001	7,492332	12,48161	3,31534E-04

3.3.1.5 Seleksi

Proses seleksi ini bertujuan untuk mempertahankan individu yang mempunyai nilai *fitness* yang tinggi agar dapat hidup untuk generasi selanjutnya. Pada penelitian ini metode seleksi yang akan digunakan adalah metode *elitism selection*, yaitu dengan mengumpulkan semua individu *parent* maupun *offspring* dalam suatu populasi satu penampungan. Individu dengan nilai *fitness* yang besar merupakan *chromosome* terbaik dalam penampungan dalam populasi tersebut akan lolos untuk masuk dalam generasi selanjutnya, individu terbaik diambil sesuai dengan *popSize* yang ditentukan sebelumnya. Metode *elitism selection* ini akan menjamin individu yang terbaik akan selalu lolos. Berikut Tabel 3.17 adalah hasil dari reproduksi setelah diurutkan berdasarkan fitness terbesar

Tabel 3.6 Hasil Sortir *Fitness*

PARENT	GENERASI BARU				FITNESS
	λ	ϵ	C	cLR	
P2	5	0,00000001	5	0,1	29,7053
P1	4	0,0000001	4	0,03	8,3902923
P3	8	0,00001	1	0,006	2,9911649
C2	2,98990896	1,57629E-06	6,249009	1,2960192	0,1187934
P4	2	0,000000001	7	2	5,72589E-03
C3	3,84998	0,000000001	4,492332	0,581613	2,78347E-03
C1	7,01009104	8,42471E-06	1,750991	0,7099808	2,24404E-03
C4	8,84998	0,000000001	7,492332	12,48161	3,31534E-04
P5	10	0,0000001	8	12	2,43795E-04

Hasil akhir berupa hasil seleksi individu-individu terbaik sejumlah populasi awal untuk dilanjutkan ke generasi selanjutnya. Hasil seleksi individu terbaik ditampilkan pada Tabel 3.18.

Tabel 3.7 Hasil Seleksi Menggunakan *Elitism Selection*

P(t-1)	Asal P	Fitness
P1	P2	29,7053
P2	P1	8,3902923
P3	P3	2,9911649
P4	C2	0,1187934
P5	P4	5,72589E-03

Setelah dilakukan proses seleksi, maka dilanjutkan dengan memilih kromosom terbaik yang dapat ditentukan berdasarkan nilai *fitness* terbesar. Pada Tabel 3.13, kromosom yang memiliki *fitness* terbesar adalah *Parent* ke-2 dengan nilai 29,7053. Dari hasil perhitungan Algoritma Genetika didapatkan parameter SVR terbaik yaitu *Parent* ke-2. Detail *Parent* ke-2 ditunjukkan pada Tabel 3.19.

Tabel 3.8 Detail Kromosom Terbaik

Parent	Kromosom				Fitness= (1/error)
	1	2	3	4	
	λ	ϵ	C	cLR	
P2	5	0,00000001	5	0,1	29,7053

3.3.2 Proses Pelatihan SVR

Terdapat sebuah studi kasus yaitu memprakiraan konsumsi listrik menggunakan SVR. Estimasi dilakukan dengan membentuk model regresi, yang kemudian diuji untuk mendapatkan nilai error. Nilai error tersebut mengindikasikan bagaimana tingkat akurasi model regresi yang dibentuk. Tujuan utamanya adalah meminimalkan tingkat error estimasi.

Proses prakiraan konsumsi listrik dengan SVR adalah berikut:

1. Menentukan data latih dan data uji.
2. Normalisasi data
3. Proses *sequential learning*
   a. Inisialisasi parameter.
   b. Inisialisasi nilai α dan α^*.
   c. Membangun model regresi.
   d. Menghitung *learning rate*.
   e. Menghitung nilai *error*.
   f. Menghitung nilai $\delta\alpha$ dan $\delta\alpha^*$.
   g. Memperbarui nilai α dan α^*.
4. Menguji model regresi.
5. Denormalisasi
6. Menghitung tingkat error.

Sebelum data pemakaian kWh dihitung menggunakan SVR perlu diketahui bahwa data dengan periode tertentu (pertahun) (x) membutuhkan data sebanyak lebih dari 1 periode. Sehingga, dengan data pemakaian listrik 3 tahun terakhir (3x)

membutuhkan data dengan periode yang sama sebanyak 3 macam. Data pemakaian konsumsi selama 4 tahun akan ditampilkan pada Tabel 3.9.

Tabel 3.9 ***Pencapaian Bulanan Per 4 Tahun Terakhir***
(data dibuat dalam satuan juta kWh)

THN	PENCAPAIAN PERBULAN DALAM 3 TAHUN TERAKHIR											
	JAN	FEB	MAR	APR	MEI	JUNI	JULI	AGS	SEPT	OKT	NOV	DES
2010	14,654	9,754	13,558	13,153	14,489	14,457	14,577	14,105	14,385	14,380	14,465	14,691
2011	14,354	14,782	14,706	14,049	14,682	15,056	14,430	14,193	13,965	14,580	14,868	13,666
2012	14,976	15,506	15,368	15,069	15,565	15,888	15,706	14,981	14,498	16,101	16,427	16,664
2013	32,042	15,723	15,938	16,467	16,825	16,668	16,047	15,693	16,287	17,030	16,851	16,746
2014	16,481	16,658	16,654	16,844	17,680	17,169	16,584	16,615	16,744	17,448	17,883	17,624

1. Menentukan data latih

Tabel 3.2 diformulasikan untuk untuk mendapatkan prakiraan pemakaian pada bulan berikutnya berdasarkan pemakaian 4 bulan ke belakang. Permasalahan tersebut diformulasikan pada Tabel 3.10.

Tabel 3.10 Data Set

No	X1	X2	X3	X4	Aktual
1	16480785	16658338	16654060	16844239	17680031
2	16658338	16654060	16844239	17680031	17169211
3	16654060	16844239	17680031	17169211	16584303
4	16844239	17680031	17169211	16584303	16614758

Pada tabel 3.10, ". X4 menurpakan data konsumsi listrik pada bulan tersebut, X3 merupakan data konsumsi pada bulan sebelumnya dan seterusnya sampai X1.

2. Normalisasi data

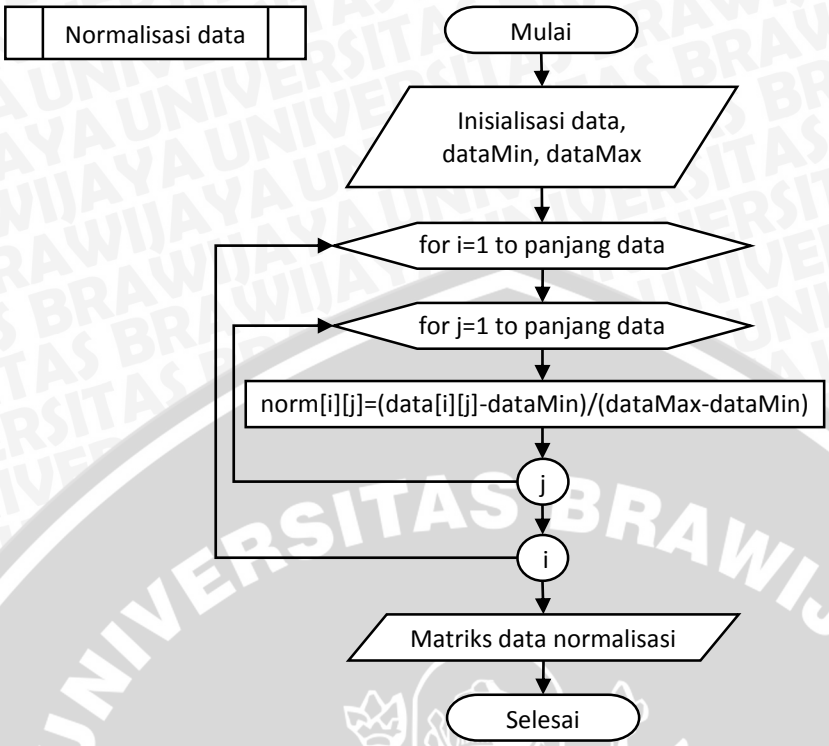
Diagram alir dari proses normalisasi seperti pada Gambar 3.8.

Dalam proses normalisasi, langkah pertama yang dilakukan adalah mencari nilai maksimum dan minimum dari data.

Tabel 3.11 Range Normalisasi

Minimum	9753559
Maksimum	31348538

Tabel 3.3 merupakan tabel yang berisi hasil pencarian nilai maksimum dan minimum data. Kedua menormalisasi data dengan Persamaan 2.6 Berikut adalah contoh perhitungannya, misalnya diambil 1 data dari dataset dengan atribut X1 yaitu nomer 3 yang bernilai 16654060. Tabel 3.12 berisi hasil normalisasi untuk pengaturan nilai fitur.



Gambar 3.8 Diagram Alir Proses Normalisasi

$$x_1 = \frac{x - x_{\min}}{x_{\max} - x_{\min}} = \frac{16654060 - 9753559}{31348538 - 9753559} = \frac{6900501}{21594979} = 0{,}3195419$$

Tabel 3.12 Hasil Normalisasi

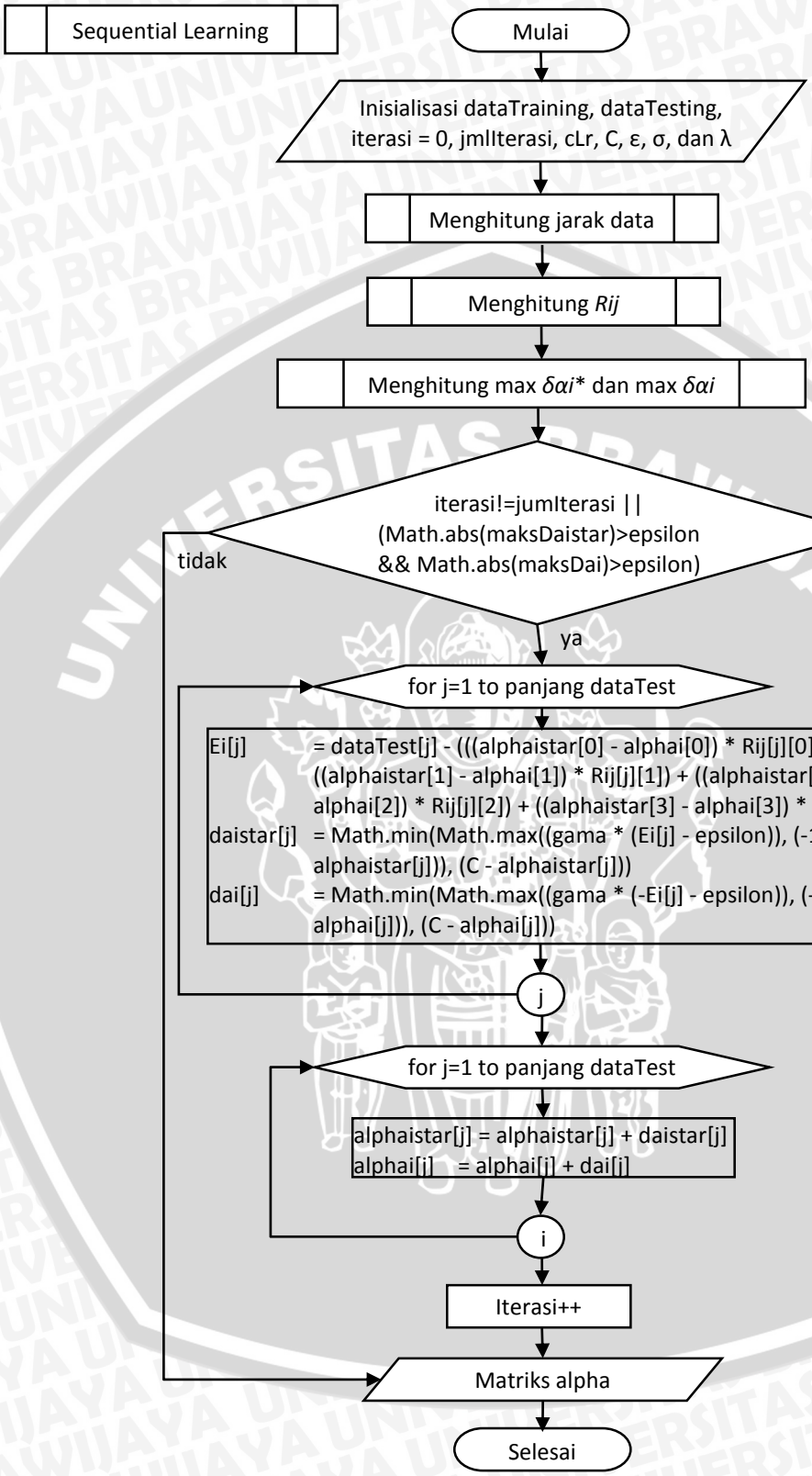
No	X1	X2	X3	X4	Aktual
1	0,3115181	0,3197400	0,3195419	0,3283485	0,3670516
2	0,3197400	0,3195419	0,3283485	0,3670516	0,3433970
3	0,3195419	0,3283485	0,3670516	0,3433970	0,3163117
4	0,3283485	0,3670516	0,3433970	0,3163117	0,3177220

3. Proses *sequential learning*

Diagram alir dari proses *sequential learning* seperti pada Gambar 3.9. Pada proses pembelajaran algoritma konsekutif, langkah pertama yang dilakukan adalah melakukan inisialisasi parameter SVR. Tabel 3.13 berisi nilai inisialisasi tiap parameter yang juga merupakan kromosom dari P2 pada sub bab 3.3.1:

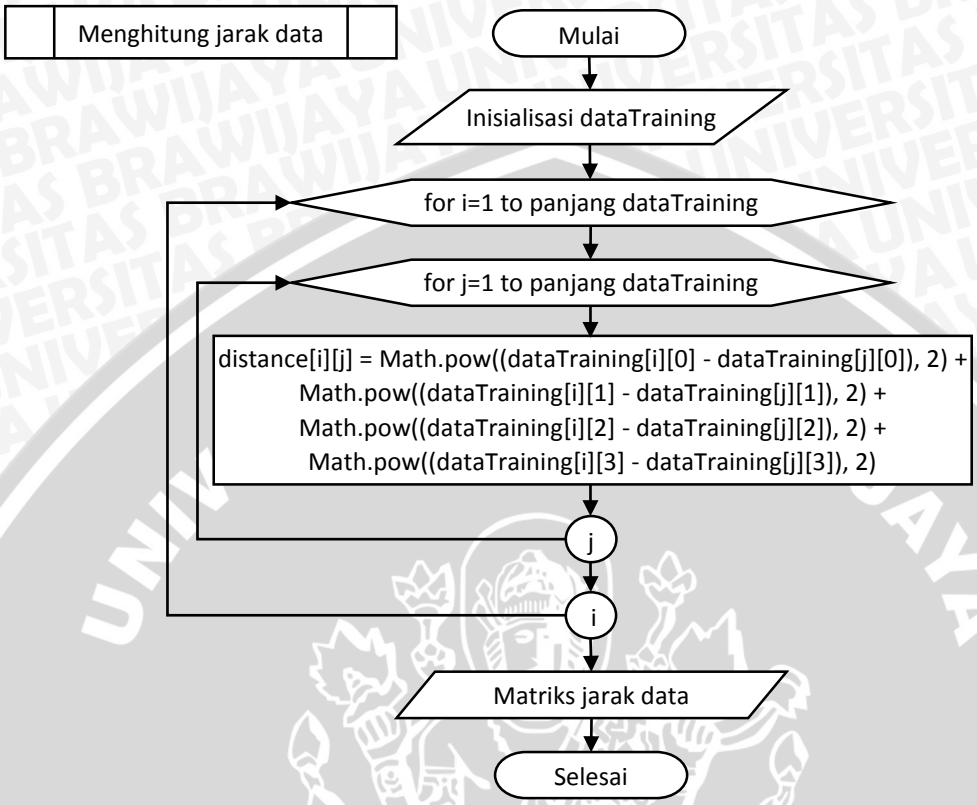
Tabel 3.13 Nilai Parameter SVR

σ	λ	ε	C	cLR
0,1	5	0,00000001	5	0,1



Gambar 3.9 Diagram Alir *Sequential Learning*

Langkah kedua, inisialisasi nilai α dan α* yaitu memberikan nilai 0 pada α dan α* sebanyak jumlah data latih. Langkah ketiga, menghitung jarak tiap data latih. Tahapan proses perhitungan jarak data latih dapat dilihat pada Gambar 3.10.



Gambar 3.10 Diagram Alir Proses Perhitungan Jarak Data Latih

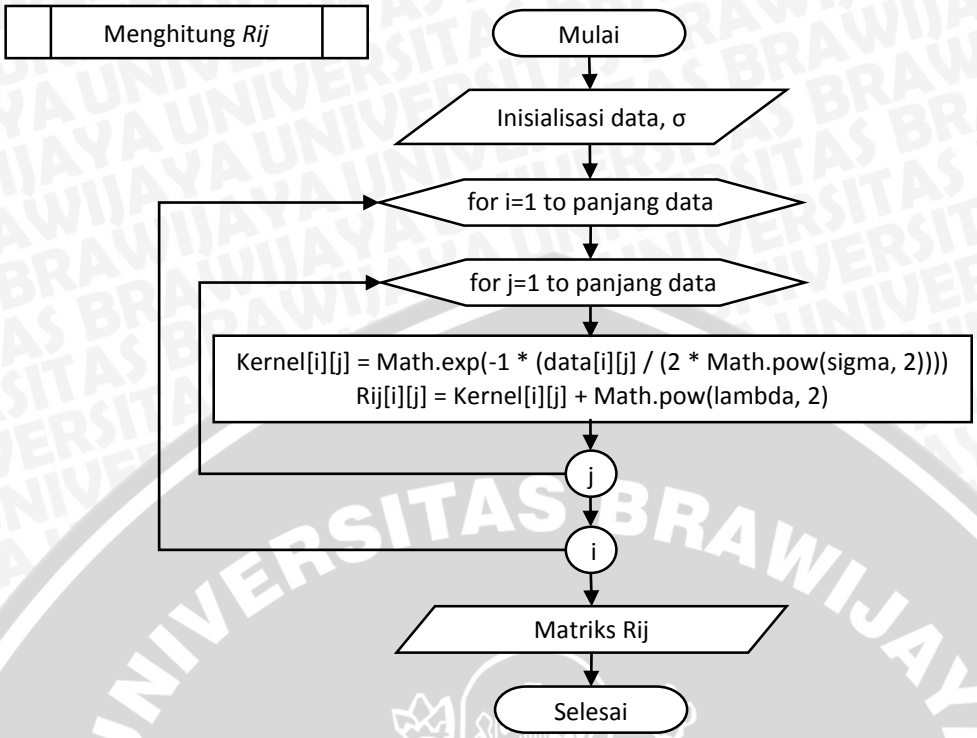
Berikut ini contoh perhitungan dengan data latih ke-1 dengan data latih ke-2. Pertama, menghitung jarak antar data. Tabel 3.14 berisi hasil perhitungan jarak data latih.

$$\|x_1 - x_2\|^2 = (0{,}3115 - 0{,}3197)^2 + (0{,}3197 - 0{,}3195)^2 + (0{,}3195 - 0{,}3283)^2 + (0{,}3283 - 0{,}36705)^2$$
$$= 0{,}001643124$$

Tabel 3.14 Jarak Data Latih

Data ke	1	2	3	4
1	0	0,001643124	0,00262212	0,00323561
2	0,001643124	0	0,00213506	0,00513228
3	0,002622118	0,002135062	0	0,00286864
4	0,003235606	0,005132278	0,00286864	0

Keempat, membentuk model regresi dalam bentuk matriks *Rij* berdasarkan jarak data yang telah diperoleh sebelumnya menggunakan persamaan (2.15). Tahapan proses perhitungan matrik *Rij* dapat dilihat pada Gambar 3.11.



Gambar 3.11 Diagram Alir Proses Perhitungan Matriks Rij

Berikut ini contoh perhitungan menggunakan data latih ke-1 dan 2. Tabel 3.15 berisi hasil model regresi yang telah dibentuk.

$$R_{1,2} = K(x_1, x_2) + \lambda^2 = \exp\left(-\frac{0{,}001643124}{2\times(0{,}1)^2}\right) + 5^2 = 25{,}92112856$$

Tabel 3. 15 Matriks Rij

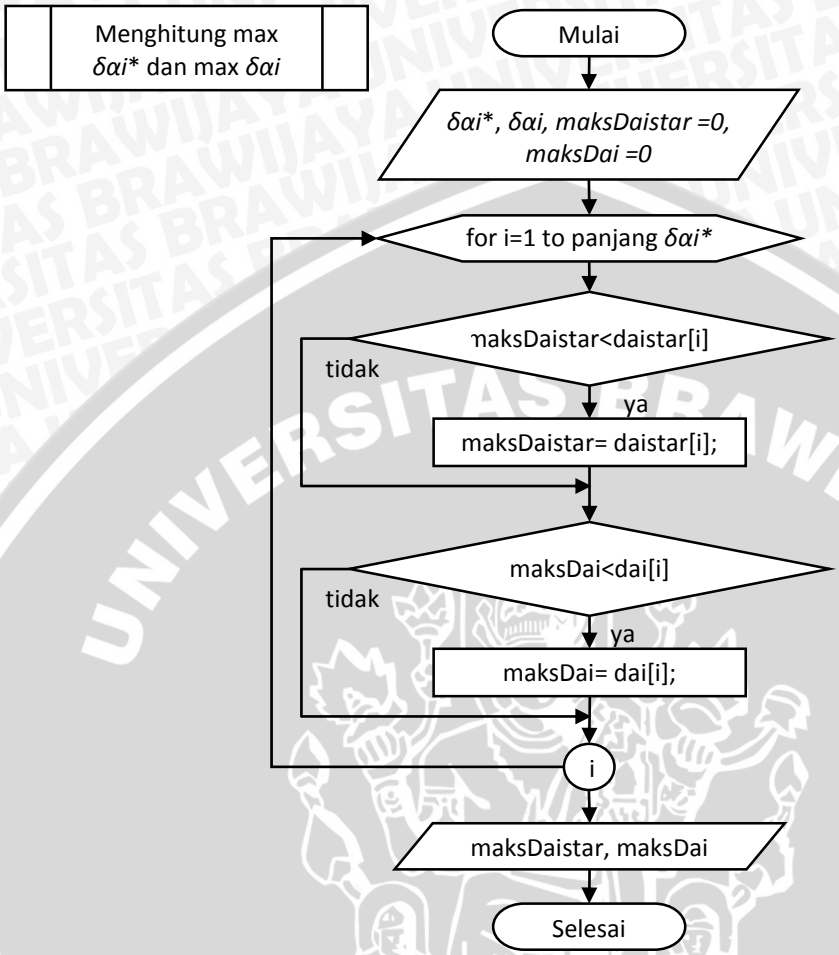
Data ke	1	2	3	4
1		25,921128056	25,877124880	25,850628059
2	25,921128056	26	25,898747525	25,773666885
3	25,877124880	25,898747525	26	25,866379690
4	25,850628059	25,773666885	25,866379690	26

Kelima menentukan learning rate berdasarkan konstanta gama dan nilai maksimal dari matriks *Rij* yang diperoleh pada tahap sebelumnya menggunakan persamaan (2.11).

$$\gamma = \frac{\text{konstanta } learning\ rate}{\max(\text{diagonal matriks Rij})} = \frac{0{,}1}{26} = 0{,}0038462$$

Pada tahap keenam, pada setiap data latih menghitung nilai *error* E_i, $\delta\alpha_i$ dan $\delta\alpha_i$* dan mencari α_i dan α_i* yang baru menggunakan rumus (2.12). Sebelumnya dicari terlebih dahulu nilai maksimum dari $\delta\alpha i$* dan maksimum $\delta\alpha i$ untuk pengecekan kondisi berhenti selain mencapai maksimum iterasi, yaitu jika

$\max(|\delta\alpha_i|) < \varepsilon$ dan $\max(|\delta\alpha_i *|) < \varepsilon$. Tahapan proses pencarian nilai maksimum *δαi** dan *δαi* dapat dilihat pada Gambar 3.12



Gambar 3. 12 Diagram Alir Proses Menghitung Maksimum *δαi dan *δαi***

Berikut ini contoh menggunakan data latih pertama:

$$E_1 = y_1 - \sum_{j=1}^{l} (\alpha_1 * - \alpha_1) R_{1j}$$

$$= 0.36705 - ((0-0)26 + (0-0)25.92112 + (0-0)25.87712) + (0-0)25.85062$$

$$= 0.36705 - 0 = 0.36705$$

$$\delta\alpha_1 * = \min\{\max[\gamma(E_1 - \varepsilon), -\alpha_1 *], C - \alpha_1 *\}$$

$$= \min\{\max[0{,}0038462(0.36705 - 0{,}00000001), -0], 5 - 0\}$$

$$= \min\{\max[0.0014117377, 0], 5\}$$

$$= \min\{0.0014117377, 5\}$$

$$= 0.0014117377$$

$$\delta\alpha_1 = \min\{\max[\gamma(-E_1 - \varepsilon), -\alpha_1], C - \alpha_1\}$$

$= \min\{\max[0{,}0038462(-0.36705-0{,}00000001),-0],5-0\}$

$= \min\{\max[-0.001411737,0],5\}$

$= \min\{0,5\}$

$= 0$

Langkah terakhir yaitu menghitung α_i dan α_i* dari hasil yang didapatkan pada perhitungan sebelumnya menggunakan rumus (2.12).

$\alpha_1 *$ $= \delta\alpha_1 * + \alpha_1 *$

$= 0{,}0014117377 + 0$

$= 0{,}001411737$

α_1 $= \delta\alpha_1 + \alpha_1$

$= 0 + 0$

$= 0$

Langkah tersebut dilakukan berulang-ulang sampai mencapai kondisi (iterasi) tertentu. Pada kasus ini dilakukan sebanyak 10 iterasi. Nilai α_i dan α_i* setelah iterasi ke 10 yaitu pada Tabel 3.16.

Tabel 3.16. Hasil αi* dan αi setelah iterasi ke 10

*αi**	*αi*
0,004429860	0,000000000
0,003525824	0,000000000
0,002483473	0,000148578
0,002552891	0,000118867

4. Menguji model regresi

Pada proses pengujian model regresi dilakukan untuk mencari prakiraan konsumsi listrik dari perhitungan *sequential learning*. Diagram alir proses menguji model regresi ditunjukkan pada Gambar 3.13 Pengujian model regresi menggunakan rumus (2.19).

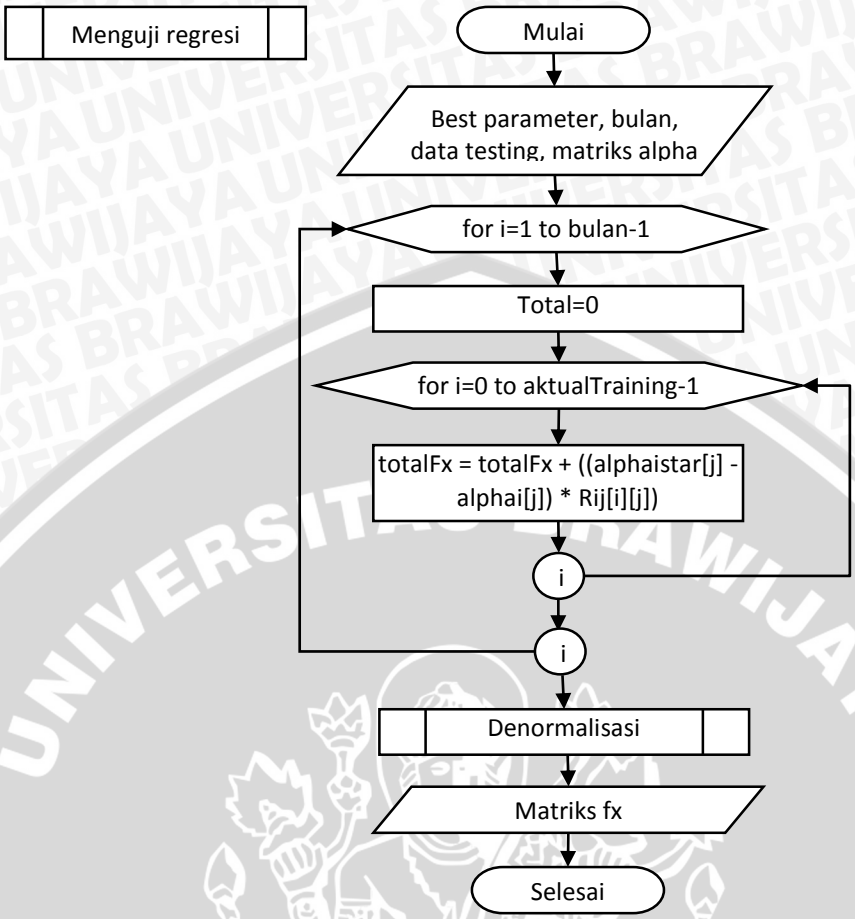
$f(x_1)$ $= \sum_{i=1}^{l} (\alpha_i^* - \alpha_i)(K(x_i, x_1) + \lambda^2).$

$= \sum_{i=1}^{l} (\alpha_i^* - \alpha_i)Rij$

$= (0{,}0044298 - 0)(26) + (0{,}003525824 - 0)(25{,}921128)$
$(0{,}002483 - 0{,}000148)(25{,}877124) + (0{,}002552 - 0{,}000118)(25{,}850628)$

$= 0{,}115176361 + 0{,}091393335 + 0{,}060420377 + 0{,}062921042$

$= 0{,}329911114494$



Gambar 3.13 Diagram Alir Proses Menguji Model Regresi

Tabel 3.17 Hasil model regresi yang terbentuk

No	Aktual	α_i*	α_i	*f(x)*
1	0,36705162	0,004429860	0	0,329911114494
2	0,34339705	0,003525824	0	0,329702972702
3	0,31631168	0,002483473	0,000148578	0,329613125484
4	0,31772196	0,002552891	0,000118867	0,329067981528

Nilai *f(x)* harus dilakukan denormalisasi terlebih dahulu agar menjadi prakiraan dari konsumsi listrik pada bulan berikutnya. Diagram alir denormalisasi ditunjukkan pada Gambar 3.14.

Proses denormalisasi menggunakan persamaan 2.7. Berikut adalah contoh denormalisasi $f(x_1)$. Tabel 3.18 berisi hasil denormalisai *f(x)*.

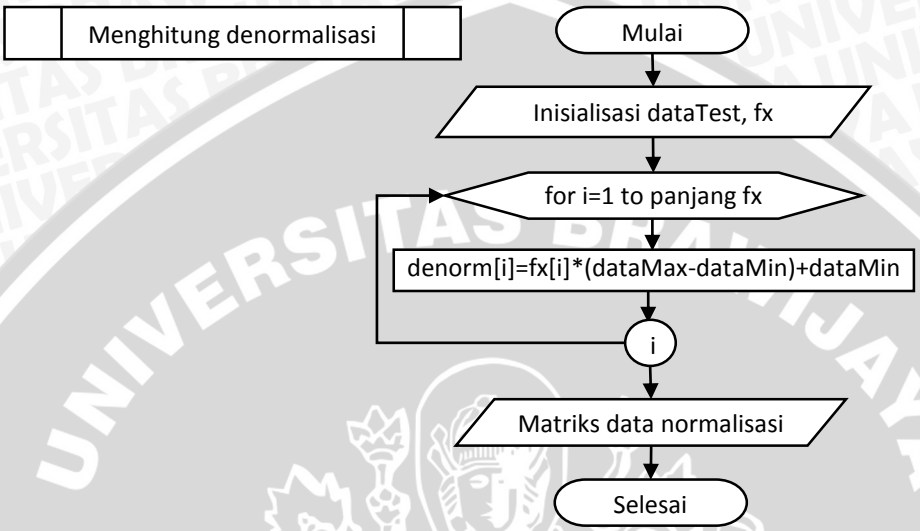
$$f'(x_1) = f(x_1)(maks - \min) + \min$$

$$= 0{,}32991111\,4494(31348538 - 9753559) + 9753559$$

$$= 16877982{,}59$$

Tabel 3.18. Hasil denormalisasi *f(x)*

f(x)	*f'(x)*
0,329911114494	16877982,59
0,329702972702	16873487,77
0,329613125484	16871547,52
0,329067981528	16859775,15



Gambar 3.14 Diagram Alir Proses Denormalisasi

5. Menghitung nilai error

Pada proses penghitungan error, penelitian ini menggunakan *Mean Absolute Percentage Error* (MAPE). Diagram alir MAPE ditunjukkan pada Gambar 3.15.

Perhitungan *Mean Absolute Percentage Error* (MAPE) memerlukan nilai data aktual dan nilai *f(x)* seperti yang ditunjukkan pada Tabel 3.19.

Tabel 3.19 Perbandingan nilai aktual dengan nilai hasil peramalan

No	Aktual	Prakiraan
1	17680031	16877982,59
2	17169211	16873487,77
3	16584303	16871547,52
4	16614758	16859775,15

Proses perhitungan *Mean Absolute Percentage Error* (MAPE) menggunakan rumus (2.13)

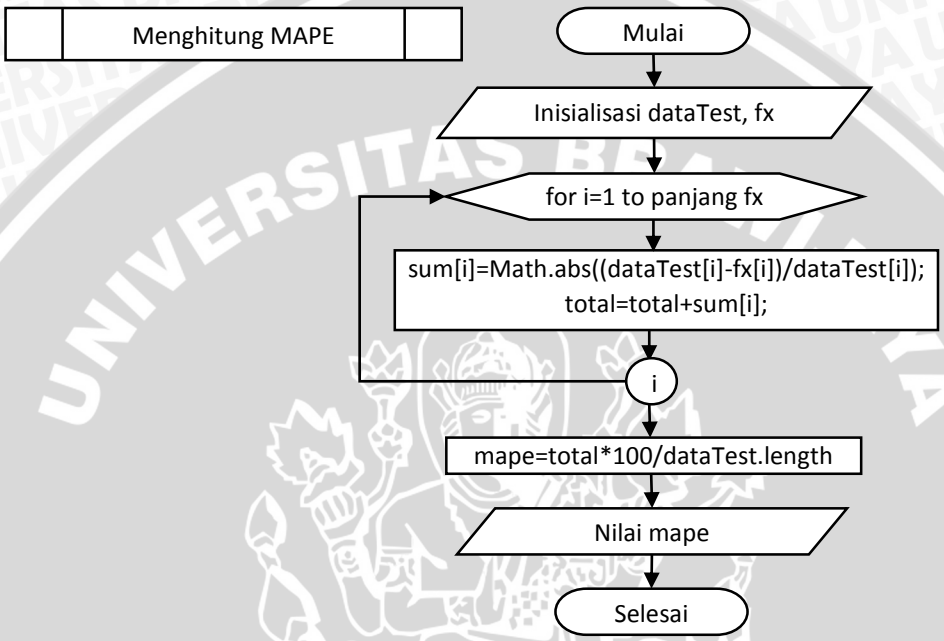
$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{y_i - y'_i}{y_i}\right| * 100$$

$$= \frac{1}{4}\left(\frac{|17680031 - 16877982|}{17680031} + \frac{|17169211 - 16873487|}{17169211}\right.$$

$$+\frac{|16584303 - 16871547|}{16584303}+\frac{|16614758 - 16859775|}{16614758}\Bigg)*100$$

$$= \frac{1}{4}(0{,}09465592)*100$$

$$= 2{,}366398$$

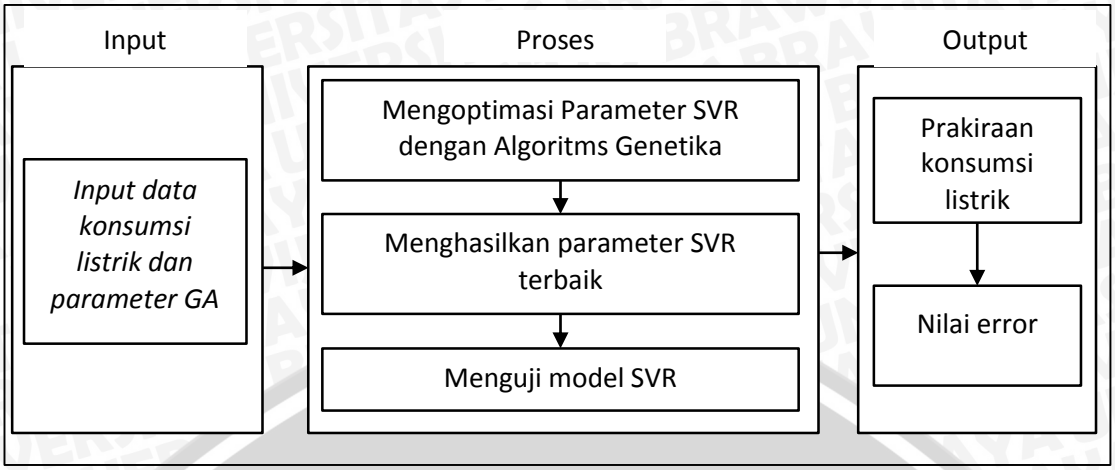


Gambar 3.15 Diagram Alir Proses Menghitung Error MAPE

3.4 Perancangan Sistem

Pada penelitian ini algoritma genetika berfungsi untuk mengoptimasi semua parameter SVR secara bersaman. Model ini mencari nilai-nilai optimal dari parameter SVR dan meningkatkan efisiensi prakiraan. Metode ini mengoptimasi parameter SVR secara dinamis, dan menggunakan parmeter yang didapatkan digunakan untuk membangun SVR yang telah dioptimasi yang berguna untuk melakukan prakiraan.

Pada fase training, sistem ini menggunakan Algoritma Genetika dan Support Vector Regression (SVR). Algoritma genetika untuk mengoptimasi parent, sedangkan SVR untuk menghitung fitnessnya. Setelah terlipih parameter terbaik, maka tinggal melakukan fase testing menggunakan SVR. Diagram model perancangan sistem dapat dilihat pada Gambar 3.16.



Gambar 3. 16 Diagram Blok Preancangan Sistem

3.5 Pengujian Sistem

Pada tahap ini dilakukan pengujian kerja sistem yang telah dibuat untuk mengetahui bahwa perangkat lunak telah mampu bekerja sesuai dengan spesifikasi dan kebutuhan. Pengujian yang dilakukan meliputi:

1. Uji coba untuk menentukan parameter yang paling optimal dalam implementasi sistem.
2. Uji coba untuk mengetahui *running time* proses pada sistem yang telah diimplementasikan
3. Uji coba untuk menentukan populasi terbaik untuk proses algoritma sistem

3.6 Penarikan Kesimpulan

Kesimpulan dilakukan setelah semua tahapan perancangan, implementasi dan pengujian sistem terhadap metode yang digunakan. Kesimpulan diambil dari hasil pengujian dan analisis metode yang diterapkan. Tahapan terakhir adalah penulisan saran berguna untuk memberikan pertimbangan atas hasil yang telah dilakukan dan memperbaiki kesalahan – kesalahan yang terjadi.

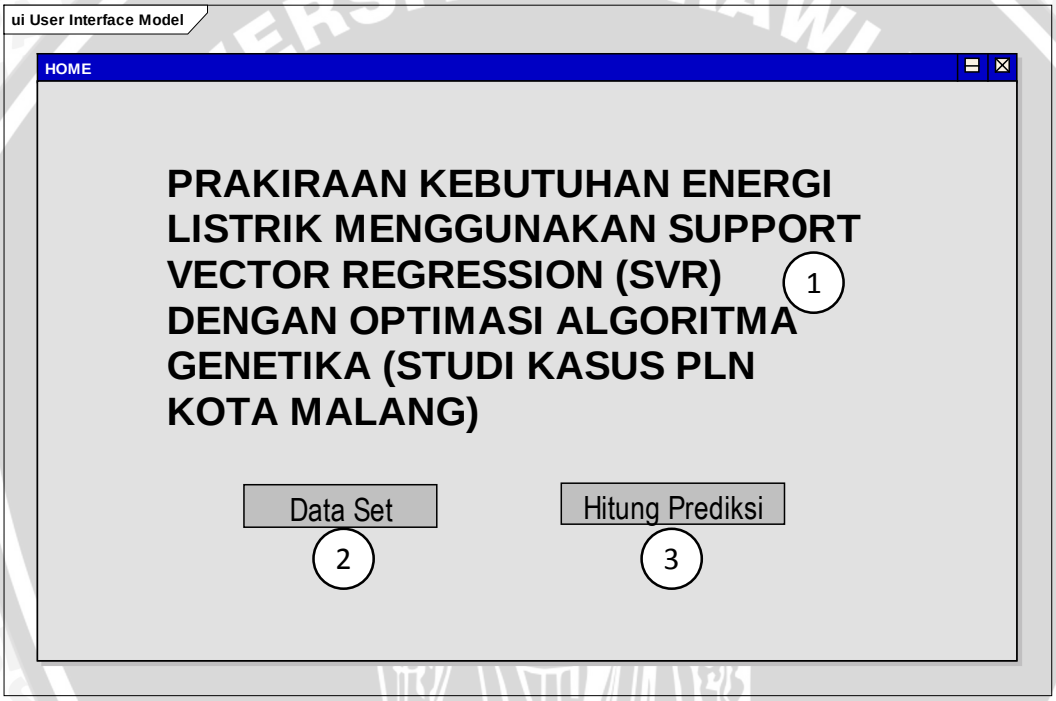
BAB 4 PERANCANGAN

4.1 Perancangan User Interface

Antarmuka sistem ini terdiri dari dua halaman. Halaman pertama adalah halaman daftar data konsumsi listrik 5 tahun terakhir. Halaman kedua adalah halaman masukan parameter algoritma genetika (GA), parameter Support Vector Regression (SVR) dan hasil optimasi yang dihasilkan sistem.

4.1.1 Rancangan Halaman Home

Halaman home merupakan tampilan awal ketika memulai aplikasi, halaman ini bersisi nama aplikasi, button start, button help, dan button about aplication. Rancangan halaman home akan ditampilkan pada Gambar 4.1.



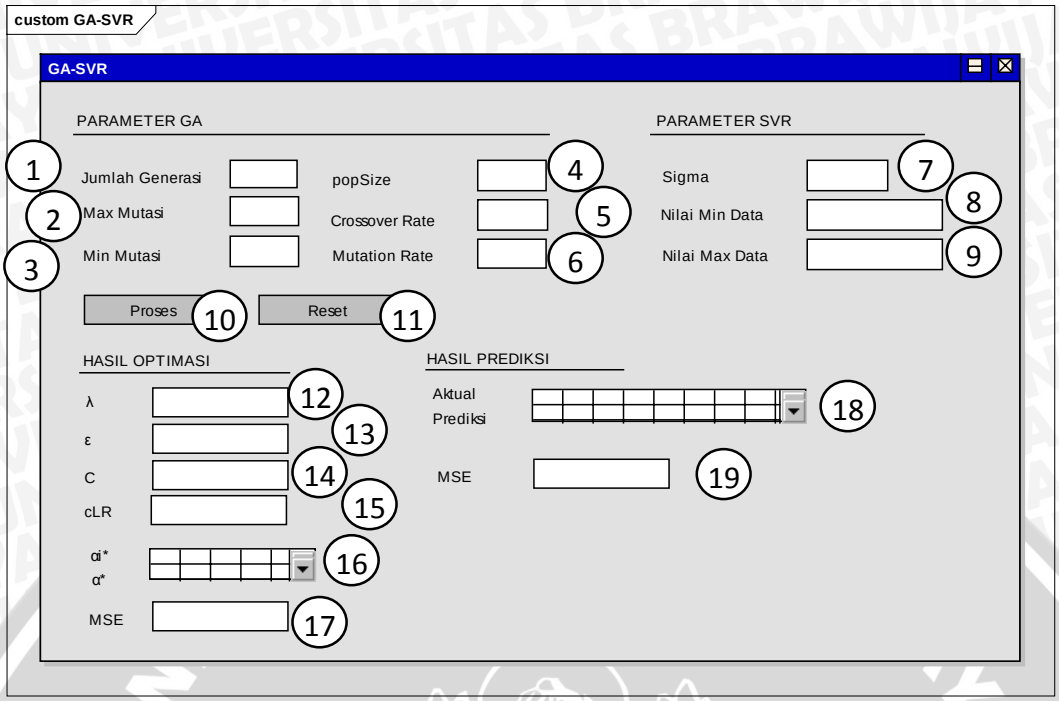
Gambar 4.1 Halaman Home

Keterangan:

1. Judul aplikasi
2. *Button* Data Set untuk menampilkan data yang digunakan
3. *Button* Hitung Prakiraan untuk memulai perhitungan prakiraan

4.1.2 Rancangan Halaman Parameter Algoritma dan Hasil Optimasi

Halaman ketiga merupakan halaman untuk input parameter GA-SVR dan menampilkan hasil optimasi dan prakiraan akan ditampilkan pada Gambar 4.3.



Gambar 4.2 Halaman Parameter Algoritma dan Hasil Optimasi

Keterangan:

1. *Text box* untuk memasukkan jumlah generasi algoritma genetika
2. *Text box* untuk memasukkan nilai max mutasi
3. Te*ks box* untuk memasukkan nilai min mutasi
4. *Text box* untuk memasukkan jumlah populasi algoritma genetika
5. *Text box* untuk memasukkan crossover rate
6. *Text box* untuk memasukkan mutation rate
7. *Text box* untuk memasukkan nilai sigma
8. *Text box* untuk memasukkan nilai minimun data set
9. *Text box* untuk memasukkan nilai maksimum data set
10. *Button* Proses untuk memproses masukan parameter algoritma yang nantinya akan menghasilkan hasil optimasi parameter SVR, fitur yang terpilih, dan hasil prakiraannya
11. *Button Reset* untuk me-*reset* ulang seluruh isi *box*
12. *Text box* untuk menampilkan nilai *λ* yang sudah dioptimasi
13. *Text box* untuk menampilkan nilai ε yang sudah dioptimasi
14. *Text box* untuk menampilkan nilai *C* yang sudah dioptimasi
15. *Text box* untuk menampilkan nilai *cLR* yang sudah dioptimasi
16. *Table* untuk menampilkan α_i^* dan α_i
17. *Text box* untuk menampilkan nilai *Mean Square Error* (MSE) data set saat *fase trainning*
18. *Table* untuk menampilkan data aktual dengan data hasil prakiraan
19. *Text box* untuk menampilkan nilai *Mean Square Error* (MSE) data set saat *fase testing*

4.2 Perancangan Uji Coba dan Evaluasi

Dibuatnya perancangan uji coba dan evaluasi disebabkan karena tidak adanya metode yang pasti untuk menentukan prakiraan pemakaian kWh dan menentukan parameter Algoritma Genetika. Maka untuk mengevaluasi sistem yang dibuat akan dilakukan uji coba antara lain :

1. Uji coba untuk menentukan popSize yang optimal
2. Uji coba untuk menentukan banyaknya generasi GA yang optimal
3. Uji coba untuk mencari nilai *crossover rate* dan *mutation rate* terbaik agar menghasilkan *range* yang optimal.
4. Uji coba menentukan range λ
5. Uji coba menentukan range ε
6. Uji coba menentukan range C
7. Uji coba menentukan range *cLr*
8. Uji coba untuk menentukan banyaknya iterasi SVR
9. Uji coba untuk mencari banyaknya periode pemakaian kWh yang akan diprakirakan

4.2.1 Uji Coba PopSize

Uji coba range merupakan uji coba yang dilakukan untuk mengetahui seberapa panjang nilai range agar dapat menghasilkan nilai koefisien yang optimal untuk menemukan persamaan regresi terbaik. Rancangan uji coba range popSize ditunjukkan pada Tabel 4.1.

Tabel 4.1 Rancangan Uji Coba *PopSize*

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

4.2.2 Uji Coba Banyaknya Jumlah Generasi GA

Uji coba banyaknya jumlah generasi dilakukan untuk mengetahui banyaknya generasi yang dibutuhkan agar dapat menghasilkan nilai koefisien regresi yang optimal untuk mendapatkan nilai prakiraan yang terbaik. Banyaknya generasi yang akan di uji coba adalah kelipatan 100. Rancangan uji coba banyaknya generasi ditunjukkan pada Tabel 4.2.

Tabel 4.2 Rancangan Uji Coba Banyaknya Generasi GA

Jumlah Generasi	Nilai *Fitness*										Rata-rata nilai *fitness*
	Percobaan ke-*i*										
	1	2	3	4	5	6	7	8	9	10	
10											
50											
100											
500											
1000											
1250											
1500											
1750											
2000											

4.2.3 Uji Coba Kombinasi *Crossover Rate* dan *Mutation Rate*

Uji coba kombinasi crossover rate dan mutation rate dilakukan untuk mengetahui kombinasi cr dan mr terbaik agar mendapatkan hasil koefisien regresi untuk mendapatkan hasil prakiraan konsumsi kWh listrik yang optimal. Uji coba kombinasi *cr* dan *mr* ini menggunakan nilai *cr* dan *mr* yang berbeda dalam range 0 – 1. Rancangan uji coba kombinasi crossover rate dan mutation rate ditunjukkan pada Tabel 4.3.

Tabel 4.3 Rancangan Uji Coba Crossover Rate dan Mutation Rate

Kombinasi		Nilai *Fitness*										Rata-rata
cr	*mr*	Percobaan ke-*i*										Nilai
		1	2	3	4	5	6	7	8	9	10	*Fitness*
0	1											
0,1	0,9											
0,2	0,8											
0,3	0,7											
0,4	0,6											
0,5	0,5											
0,6	0,4											
0,7	0,3											
0,8	0,2											
0,9	0,1											
1	0											

4.2.4 Uji Coba Range λ

Uji coba range merupakan uji coba yang dilakukan untuk mengetahui seberapa panjang nilai range λ agar dapat menghasilkan nilai parameter SVR yang optimal. Rancangan uji coba range minimum dan maksimum ditunjukkan pada Tabel 4.1.

Tabel 4.4 Rancangan Uji Coba Range λ

Nilai Min	Nilai Max	Nilai *Fitness*										Rata-rata nilai *fitness*
		Percobaan generasi ke-*i*										
		1	2	3	4	5	6	7	8	9	10	
0.0001	1											
	10											
	100											
0.01	1											
	10											
	100											
0.1	1											
	10											
	100											

4.2.5 Uji Coba Range ε

Uji coba range merupakan uji coba yang dilakukan untuk mengetahui seberapa panjang nilai range ε agar dapat menghasilkan nilai parameter SVR yang optimal. Rancangan uji coba range minimum dan maksimum ditunjukkan pada Tabel 4.1.

Tabel 4.5 Rancangan Uji Coba Range ε

Nilai Min	Nilai Max	Nilai *Fitness*										Rata-rata nilai *fitness*
		Percobaan generasi ke-*i*										
		1	2	3	4	5	6	7	8	9	10	
0.0000001	0.0001											
	0.001											
	0.01											
0.000001	0.0001											
	0.001											
	0.01											
0.00001	0.01											
	0.1											
	1											

4.2.6 Uji Coba Range *C*

Uji coba range merupakan uji coba yang dilakukan untuk mengetahui seberapa panjang nilai range *C* agar dapat menghasilkan nilai parameter SVR yang optimal. Rancangan uji coba range minimum dan maksimum ditunjukkan pada Tabel 4.1.

Tabel 4.6 Rancangan Uji Coba Range *C*

Nilai Min	Nilai Max	Nilai *Fitness* Percobaan generasi ke-*i*										Rata-rata nilai *fitness*
		1	2	3	4	5	6	7	8	9	10	
0.0001	0.1											
	10											
	100											
0.001	0.1											
	10											
	100											
0.01	0.1											
	10											
	100											

4.2.7 Uji Coba Range *cLr*

Uji coba range merupakan uji coba yang dilakukan untuk mengetahui seberapa panjang nilai range *cLr* agar dapat menghasilkan nilai parameter SVR yang optimal. Rancangan uji coba range minimum dan maksimum ditunjukkan pada Tabel 4.1.

Tabel 4.7 Rancangan Uji Coba Range *cLr*

Nilai Min	Nilai Max	Nilai *Fitness* Percobaan generasi ke-*i*										Rata-rata nilai *fitness*
		1	2	3	4	5	6	7	8	9	10	
0.00001	0.01											
	0.1											
	1											
0.0001	0.01											
	0.1											
	1											
0.001	0.01											
	0.1											
	1											

4.2.8 Uji Coba Banyaknya Jumlah Iterasi SVR

Uji coba banyaknya jumlah iterasi dilakukan untuk mengetahui banyaknya iterasi yang dibutuhkan agar dapat menghasilkan nilai α_i^* dan α_i yang optimal untuk mendapatkan nilai prakiraan yang terbaik. Banyaknya generasi yang akan di uji coba adalah kelipatan 100. Rancangan uji coba banyaknya generasi ditunjukkan pada Tabel 4.2.

Tabel 4.8 Rancangan Uji Coba Banyaknya Jumlah Iterasi SVR

Banyak Iterasi	Nilai *Fitness*										Rata-rata nilai *fitness*
	Percobaan ke-*i*										
	1	2	3	4	5	6	7	8	9	10	
10											
50											
100											
500											
1000											
5000											
10000											
50000											
100000											

4.2.9 Uji Coba Banyaknya Periode Pemakaian kWh

Uji coba banyaknya periode pemakaian kWh yang akan diprakirakan merupakan uji coba yang dilakukan untuk mengetahui seberapa berpengaruh periode terhadap nilai fitness. Dengan begitu bisa diketahui lama periode yang optimal untuk diprakirakan panjang nilai range agar dapat menghasilkan nilai koefisien yang optimal untuk menemukan persamaan regresi terbaik. Rancangan uji coba range popSize ditunjukkan pada Tabel 4.1.

Tabel 4.9 Rancangan Uji Coba *Pemakaian kWh*

Banyak Periode	Nilai *Fitness*										Rata-rata nilai *fitness*
	Percobaan ke-*i*										
	1	2	3	4	5	6	7	8	9	10	
1 bulan											
2 bulan											
3 bulan											
4 bulan											
5 bulan											
6 bulan											
7 bulan											
8 bulan											
9 bulan											
10 bulan											
11 bulan											
12 bulan											

BAB 5 IMPLEMENTASI

Pembahasan berfungsi untuk menerjemahkan makna dari hasil yang diperoleh untuk menjawab pertanyaan atau masalah penelitian. Fungsi lainnya adalah untuk menjelaskan pemahaman baru yang didapatkan dari hasil penelitian, yang diharapkan berguna dalam pengembangan keilmuan. Dalam penelitian tingkat lanjut, fungsi pembahasan yang kedua ini sangat penting karena dapat menunjukkan kontribusi penulis terhadap pengembangan keilmuan. Akan tetapi, dalam penelitian tingkat skripsi, fungsi yang kedua ini dapat diterapkan secara terbatas karena pendidikan S1 tidak dituntut untuk pengembangan keilmuan secara substansial, tetapi cukup terhadap pemahaman personal dalam implementasi konsep atau teori.

5.1 Implementasi Program

5.1.1 Proses Normalisasi Data

Proses normalisasi data bertujuan untuk merubah dimensi data menjadi range [0,1]. Proses ini diawali dengan melakukan normalisasi terhadap seluruh data yang akan diolah untuk standarisasi data, proses keseluruhan normalisasi dapat dilihat pada Kode Program 5.1.

```
1     public double[][] normalisasi2Dimensi(double[][] data){
2          double norm[][] = new
3           double[data.length][data.length];
4          for (int i = 0; i < data.length; i++) {
5              for (int j = 0; j < data.length; j++) {
6                  norm[i][j]=(data[i][j]-dataMin)/(dataMax-
7                  dataMin);
8                 }
9             }
10         return norm;
11    }
```

Kode Program 5. 1 Proses Normalisasi Data

Penjelasan dari Kode Program 5.1 adalah sebagai berikut :

1. Baris 2-3 merupakan inisialisasi array untuk menampung hasil normalisasi
2. Baris 6 merupakan proses normalisasi data
3. Baris 10 merupakan proses pengembalian parameter

5.1.2 Proses *Sequential Learning*

Proses *sequential learning* ini membutuhkan parameter berupa *data training*, *data testing* dan jumlah iterasi, proses *sequential learning* gdapat dilihat pada Kode Program 5.2

```
1     public double[][] seqLearning(double[][] dataTraining,
2     double[] aktualTraining, int jumIterasi) {
3         double[][] Kernel = Kernel(dataTraining);
4         double[][] Rij = Rij(Kernel);
5
6         this.gama = cLR / Rij[0][0];
```

```
7
8           double Ei[] = new double[aktualTraining.length];
9           double          alphaistar[]          =          new
10           double[aktualTraining.length];
11          double alphai[] = new double[aktualTraining.length];
12          double daistar[] = new double[aktualTraining.length];
13          double dai[] = new double[aktualTraining.length];
14
15          double maksDaistar = cekDeltaAlpha(daistar, dai)[0];
16          double maksDai = cekDeltaAlpha(daistar, dai)[1];
17
18          int iterasi =0;
19          while              (iterasi!=jumIterasi              ||
20              (Math.abs(maksDaistar)>epsilon                    &&
21              Math.abs(maksDai)>epsilon)){
22              for (int j = 0; j < aktualTraining.length; j++) {
23                  double jumlahRij = 0;
24                  for (int k = 0; k < aktualTraining.length;
25                   k++) {
26                      jumlahRij = jumlahRij + ((alphaistar[k] -
27                       alphai[k]) * Rij[j][k]);
28                  }
29                  Ei[j] = aktualTraining[j] - jumlahRij;
30                  daistar[j] = Math.min(Math.max((gama * (Ei[j]
31                   - epsilon)), (-1 * alphaistar[j])), (C -
32                   alphaistar[j]));
33                  dai[j] = Math.min(Math.max((gama * (-Ei[j] -
34                   epsilon)), (-1 * alphai[j])), (C -
35                   alphai[j]));
36              }
37              for (int j = 0; j < alphaistar.length; j++) {
38                  alphaistar[j] = alphaistar[j] + daistar[j];
39                  alphai[j] = alphai[j] + dai[j];
40              }
41              for (int j = 0; j < alphaistar.length; j++) {
42             }
43             Iterasi++;
44          }
45          for (int i = 0; i < alphaistar.length; i++) {
46              alpha[0][i] = alphaistar[i];
47              alpha[1][i] = alphai[i];
48          }
49          return alpha;
50      }
```

Kode Program 5. 2 Proses Sequential Learning

Penjelasan dari Kode Program 5.2 adalah sebagai berikut :

1. Baris 3 merupakan proses untuk inisialisai matriks untuk menampung nilai kernel
2. Baris 4 merupakan proses untuk inisialisai matriks untuk menampung nilai matrik *Rij*
3. Baris 6 merupakan inisialisai nilai γ
4. Baris 8 merupakan proses untuk inisialisai matriks untuk menampung nilai E
5. Baris 9-11 merupakan proses untuk inisialisai matriks untuk menampung nilai α_i^* dan α_i

6. Baris 12-13 merupakan proses untuk inisialisai matriks untuk menampung nilai $\delta\alpha_i^*$ dan $\delta\alpha_i$
7. Baris 15-16 merupakan proses untuk memanggil fungsi untuk mencari nilai kasimum dari $\delta\alpha_i^*$ dan $\delta\alpha_i$
8. Baris 19-21 merupakan kondisi berhenti yaitu ketika mencapai iterasi maksimum atau $\max(|\delta\alpha_i|) < \varepsilon$ dan $\max(|\delta\alpha_i *|) < \varepsilon$
9. Baris 24-29 merupakan proses untuk menghitung nilai *Ei*
10. Baris 30-32 merupakan proses untuk menghitung nilai $\delta\alpha_i^*$
11. Baris 33-35 merupakan proses untuk menghitung nilai $\delta\alpha_i$
12. Baris 38 merupakan proses meng-*update* nilai α_i^*
13. Baris 39 merupakan proses meng-*update* nilai α_i
14. Baris 46-47 merupakan proses menggbungkan α_i^* dan α_i kedalam satu matriks
15. Baris 49 merupakan proses pengembalian variabel alpha

5.1.3 Proses Perhitungan Kernel

Perhitungan kernel yang dipergunakan adalah Kernel RBF. Kernel RBF ini bertugas memetakan data ke dimensi yang lebih tinggi. Proses perhitungan Kernel dapat dilihat pada Kode Program 5.3

```
1  public double[][] Kernel(double[][] data) {
2          double Kernel[][] = new
3           double[data.length][data.length];
4          double distance[][] = new
5           double[data.length][data.length];
6          for (int i = 0; i < data.length; i++) {
7              for (int j = 0; j < data.length; j++) {
8                  distance[i][j] = Math.pow((data[i][0] -
9                   data[j][0]), 2) + Math.pow((data[i][1] -
10                  data[j][1]), 2) + Math.pow((data[i][2] -
11                  data[j][2]), 2) + Math.pow((data[i][3] -
12                  data[j][3]), 2);
13             }
14         }
15         for (int i = 0; i < data.length; i++) {
16             for (int j = 0; j < data.length; j++) {
17                 Kernel[i][j] = Math.exp(-1 * (distance[i][j]
18                  / (2 * Math.pow(sigma, 2))));
19             }
20         }
21         return Kernel;
22     }
```

Kode Program 5. 3 Proses Perhitungan Kernel

Penjelasan dari Kode Program 5.3 adalah sebagai berikut :

1. Baris 2 merupakan proses inisialisasi array untuk menampung nilai Kernel
2. Baris 3 merupakan proses inisialisasi array untuk menampung jarak antar data

3. Baris 8-12 merupakan proses perhitungan jarak antar data
4. Baris 17-18 merupakan proses untuk menghitung kernel
5. Baris 21 merupakan proses pengembalian variabel nilai kernel

5.1.4 Proses Perhitungan *Rij*

Perhitungan matriks *Rij* memerlukan parameter berupa nilai Kernel. Proses perhitungan Kernel dapat dilihat pada Kode Program 5.4

```
1       public double[][] Rij(double[][] Kernel){
2           for (int i = 0; i < Kernel.length; i++) {
3               for (int j = 0; j < Kernel.length; j++) {
4                   Rij[i][j] = Kernel[i][j] + Math.pow(lambda,
5                   2);
6               }
7           }
8           return Rij;
9       }
```

Kode Program 5. 4 Proses Perhitungan *Rij*

Penjelasan dari Kode Program 5.4 adalah sebagai berikut :

1. Baris 4 merupakan proses untuk menghitung nilai *Rij*
2. Baris 8 merupakan proses pengembalian nilai variable *Rij*

5.1.5 Proses Pencarian Nilai Maksimum *δαi** dan *δαi*

Pencarian nilai maksimum *δαi** dan *δαi* memerlukan parameter berupa matriks *δαi** dan *δαi*. Proses pencarian nilai maksimum *δαi** dan *δαi* dapat dilihat pada Kode Program 5.5

```
1       public double[] cekDeltaAlpha(double[] daistar, double[]
2       dai){
3               double[] maksDelta = {0,0};
4
5               for (int i = 0; i < daistar.length; i++) {
6                   if (maksDelta[0]<daistar[i]) {
7                       maksDelta[0] = daistar[i];
8                   }
9                   if (maksDelta[1]<dai[i]) {
10                      maksDelta[1] = dai[i];
11                  }
12              }
13              return maksDelta;
14      };
```

Kode Program 5. 5 Pencarian Nilai Maksimum *δαi dan *δαi***

Penjelasan dari Kode Program 5.5 adalah sebagai berikut :

1. Baris 3 merupakan proses untuk inisialisasi nilai maksimum *δαi** dan *δαi*
2. Baris 6-8 merupakan proses pencarian nilai maksimum *δαi**
3. Baris 9-10 merupakan proses pencarian nilai maksimum *δαi*
4. Baris 8 merupakan proses pengembalian nilai maksimum *δαi** dan *δαi*

5.1.6 Proses Perhitungan *f(x)*

Proses *f(x)* disini adalah untuk menghitung nilai prakiraan. Proses ini membutuhkan parameter berupa nilai α_i^* dan α_i. Proses perhitungan Kernel dapat dilihat pada Kode Program 5.6

```
1     public double[] Fx(double[][] alpha) {
2             double fx[] = new double[aktualTraining.length];
3             double alphaistar[] = new
4              double[aktualTraining.length];
5             double alphai[] = new
6              double[aktualTraining.length];
7             for (int i = 0; i < aktualTraining.length; i++) {
8                  alphaistar[i] = alpha[0][i];
9                  alphai[i] = alpha[1][i];
10            }
11            for (int i = 0; i < aktualTraining.length; i++) {
12             double totalFx = 0;
13            for (int j = 0; j < aktualTraining.length;
14             j++) {
15                 totalFx += (alphaistar[j] - alphai[j]) *
16                  Rij[i][j];
17             } fx[i] = totalFx;
18            }       return fx;
19    }
```

Kode Program 5. 6 Proses Perhitungan *f(x)*

Penjelasan dari Kode Program 5.6 adalah sebagai berikut :

1. Baris 2 merupakan proses inisialisai matriks untuk menampung nilai *f(x)*
2. Baris 3-4 merupakan proses inisialisai matriks untuk menampung nilai α_i^*
3. Baris 5-6 merupakan proses inisialisai matriks untuk menampung nilai α_i^*
4. Baris 8-9 merupakan proses pemisahan matriks alpha menjadi α_i^* dan α_i^*
5. Baris 15-16 merupakan proses perhitungan nilai *f(x)*
6. Baris 19 merupakan proses pengembalian variable nilai *f(x)*

5.1.7 Proses Perhitungan *Error Rate*

Metode perhitungan error rate yang digunakan pada penelitian ini adalah *Mean Absolute Percentage Error* (MAPE). Proses perhitungan Kernel dapat dilihat pada Kode Program 5.7

```
1     public double MAPE(double[] fx, double[] aktual) {
2              double sum[] = new double[aktual.length];
3              double total = 0, mape;
4              for (int i = 0; i < aktual.length; i++) {
5                  sum[i] = Math.abs((aktual[i] - fx[i]) /
6                   aktual[i]);
7                  total = total + sum[i];
8              } mape = 100 * total / aktual.length;
9              return mape;
10     }
```

Kode Program 5. 7 Proses Perhitungan *Error Rate*

Penjelasan dari Kode Program 5.7 adalah sebagai berikut :

1. Baris 2 merupakan proses inisialisai matriks untuk menampung jumlah data
2. Baris 3 merupakan inisialisasi variabel total dan MAPE
3. Baris 5-7 merupakan proses perhitungan nilai MAPE
4. Baris 9 merupakan proses perhitungan nilai presentase MAPE
5. Baris 10 merupakan proses pengembalian nilai variabel MAPE

5.1.8 Proses Denormalsasi Data

Proses denormalisasi alah proses pengembalian data prakiraan ke dimensi awal. Proses perhitungan Kernel dapat dilihat pada Kode Program 5.8

```
1     public double[] denormalisasi(double[] fx){
2           double denorm[] = new double[fx.length];
3           for (int i = 0; i < fx.length; i++) {
4                 denorm[i]=fx[i]*(dataMax-
5                 dataMin)+dataMin;
6           }
7           return denorm;
8     }
```

Kode Program 5. 8 Proses Denormalisasi Data

Penjelasan dari Kode Program 5.8 adalah sebagai berikut :

1. Baris 2 merupakan proses inisialisai matriks untuk menampung nilai hasil denormalisasi
2. Baris 4-5 merupakan proses perhitungan denormalisasi
3. Baris 7 merupakan proses pengembalian nilai variabel denormalisasi

5.1.9 Proses Pembangkitan Parent Awal

Proses pada algoritma genetika dimulai dengan pembangkitan *parent*. Proses perhitungan Kernel dapat dilihat pada Kode Program 5.9

```
1     public double[][] inisialisasiParent() {
2          double[][] p = new double[popSize][4];
3          for (int i = 0; i < popSize; i++) {
4               p[i][0]=ThreadLocalRandom.current().nextDouble(
5                 bMin_lambda, bMax_lambda);
6               p[i][1]=ThreadLocalRandom.current().nextDouble(
7                 bMin_epsilon, bMax_epsilon);
8               p[i][2]=ThreadLocalRandom.current().nextDouble(
9                 bMin_C, bMax_C);
10              p[i][3]=ThreadLocalRandom.current().nextDouble(
11                bMin_cLr, bMax_cLr);
12         }
13         this.parent=p;
14         return parent;
15    }
```

Kode Program 5. 9 Proses Pambangkitan Parent Awal

Penjelasan dari Kode Program 5.9 adalah sebagai berikut :

1. Baris 2 merupakan proses inisialisai matriks untuk menampung parent awal
2. Baris 4-5 merupakan proses pembangkitan nilai λ dengan range yang sudah ditentukan sebelumnya

3. Baris 6-7 merupakan proses pembangkitan nilai *ε* dengan range yang sudah ditentukan sebelumnya
4. Baris 8-9 merupakan proses pembangkitan nilai *C* dengan range yang sudah ditentukan sebelumnya
5. Baris 10-11 merupakan proses pembangkitan nilai *cLr* dengan range yang sudah ditentukan sebelumnya
6. Baris 14 merupakan proses pengembalian nilai variable parent

5.1.10 Proses Perhitungan *Fitness*

Setelah menghitung error rate selanjutnya menghitung fitness. Proses ini memerlukan parameter nilai error. Proses perhitungan Kernel dapat dilihat pada Kode Program 5.10

```
1          public double Fitness(double lambda, double epsilon,
2           double C, double cLr) {
3              double error = svr.hitungSVR(jblIterasiSVR,
4               lambda, epsilon, C, cLr);
5              fitness = 100 / (1 + error);
6              return fitness;
7      }
```

Kode Program 5. 10 Proses Perhitungan *Fitness*

Penjelasan dari Kode Program 5.10 adalah sebagai berikut :

1. Baris 3 merupakan proses mendapatkan nilai error dari SVR
2. Baris 5 merupakan proses fitness
3. Baris 6 merupakan proses pengembalian variabel nilai fitness

5.1.11 Proses *Crossover*

Pada proses *crossover* ini jenis *crossover* yang digunakan adalah *extended intermediate crossover*. Proses perhitungan Kernel dapat dilihat pada Kode Program 5.11

```
1      public double[][] crossover(double[][] Parent, double
2      offspringCrossover){
3           int offspring = (int) offspringCrossover/2;
4           double chlidCrossover[][] = new
5           double[(int)offspringCrossover][4];
6           int x=0;
7           for (int i = 0; i < offspring; i++) {
8               double alpha[] = new double[4];
9               for (int k = 0; k < 4; k++) {
10                   alpha[k]=
11                   ThreadLocalRandom.current().nextDouble(0,
12                   1);
13              }
14              int parent1, parent2;
15              do{
16                  parent1 =
17                   ThreadLocalRandom.current().nextInt(0,paren
18                   t.length);
19                  parent2 =
20                   ThreadLocalRandom.current().nextInt(0,paren
21                   t.length);
```

```
22              }
23              while (parent1 == parent2);
24              for (int j = 0; j < 4; j++) {
25                  if (offspringCrossover==1) {
26                      chlidCrossover[x][j] =
27                       Parent[parent1][j] + (alpha[j] *
28                       (Parent[parent2][j] -
29                       Parent[parent1][j]));
30                  }
31                  else{
32                      chlidCrossover[x][j] =
33                       Parent[parent1][j] + (alpha[j] *
34                       (Parent[parent2][j] -
35                       Parent[parent1][j]));
36                      chlidCrossover[x+1][j] =
37                       Parent[parent2][j] + (alpha[j] *
38                       (Parent[parent1][j] -
39                       Parent[parent2][j]));
40                  }
41              }
42          x=x+2;
43          }
44          return chlidCrossover;
45      }
```

Kode Program 5. 11 Proses *Crossover*

Penjelasan dari Kode Program 5.11 adalah sebagai berikut :

1. Baris 3 merupakan proses perhitungan offspring
2. Baris 4-5 merupakan proses inisialisasi martiks untuk menampung anak hasil *crossover*
3. Baris 8 merupakan proses inisialisasi matriks untuk menampungnilai *α*
4. Baris-10-12 merupakan proses pembangkitan nilai *α* dengan range (1,0)
5. Baris 16-23 merupakan proses pemilihan 2 parent secara random untuk disilangkan
6. Baris 26-40 merupakan proses *corssover*
7. Baris 44 merupakan proses pengembalian nilai variable hasil crossover

5.1.12 Proses Mutasi

Pada proses mutasi ini jenis mutasi yang digunakan adalah *random mutation*. Proses perhitungan Kernel dapat dilihat pada Kode Program 5.12.

```
1       public double[][] mutasi(double[][] Parent, double
2       offspringMutasi){
3           int offspring = (int) offspringMutasi;
4           double chlidMutasi[][] = new
5           double[(int)offspringMutasi][4];
6           int x=0;
7           for (int i = 0; i < offspring; i++) {
8               double r[] = new double[4];
9               for (int k = 0; k < 4; k++) {
10                  r[k] =
11                   ThreadLocalRandom.current().nextDouble(-1,
12                   1);
13              }
14              int parent1;
```

```
15              parent1 =
16               ThreadLocalRandom.current().nextInt(0,parent.le
17               ngth);
18              for (int j = 0; j < 4; j++) {
19                  chlidMutasi[x][j] =
20                   Parent[parent1][j]+(r[j]*(maxMutasi-
21                   minMutasi));
22              }
23          x++;
24          }
25          return chlidMutasi;
26      }
```

Kode Program 5. 12 Proses Mutasi

Penjelasan dari Kode Program 5.12 adalah sebagai berikut :

1. Baris 3 merupakan proses perhitungan offspring
2. Baris 4 merupakan proses insialisasi matriks untuk menampung hasil mutasi
3. Baris 8 merupakan proses insialisasi matriks untuk menampung nilai *r*
4. Baris 10-12 merupakan proses pembangkitan nilai *r* dengan range (-1,1)
5. Baris 16-17 merupakan proses pemilihan parent secara random untuk dimutasi
6. Baris 19-21 merupakan proses perhitungan mutasi
7. Baris 25 merupakan proses mengembalian nilai variabel hasil mutasi

5.1.13 Proses Seleksi

Pada proses seleksi ini jenis seleksi yang digunakan adalah *elitism selection*. Seleksi ini memastikan kromosom terbaik lolos ke generasi berikutnya. Proses perhitungan Kernel dapat dilihat pada Kode Program 5.13.

```
1       public double[][] seleksi(double[][] child) {
2               Arrays.sort(child, new Comparator<double[]>() {
3                   @Override
4                   public int compare(double[] o1, double[] o2)
5       {
6                       return Double.compare(o2[4], o1[4]);
7                   }
8               });
9               double newParent[][] = new double[popSize][5];
10              for (int i = 0; i < popSize; i++) {
11                  for (int j = 0; j < 5; j++) {
12                      newParent[i][j] = child[i][j];
13                  }
14              }
15              return newParent;
16      }
```

Kode Program 5. 13 Proses Seleksi

Penjelasan dari Kode Program 5.13 adalah sebagai berikut :

1. Baris 2 merupakan method untuk untuk mengurutkan array
2. Baris 6 merupakan proses pembandingan dua buah nilai berdasarkan kolom yang ditentukan
3. Baris 9 merupakan proses inisialisasi matriks untuk menampung *parent* baru

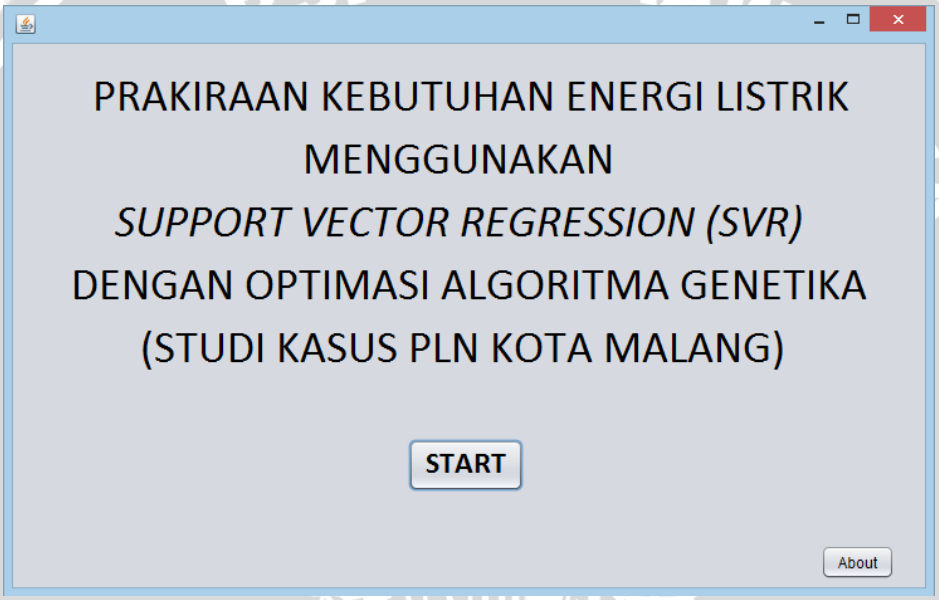
4. Baris 12 merupakan proses penentuan *parent* baru
5. Baris 15 merupakan proses pengembalian nilai variabel *parent* baru

5.2 Implementasi Antarmuka

Sub bab Implementasi Antarmuka merupakan pembahasan mengenai hasil implementasi antarmuka yang sebelumnya dirancangan pada Bab 4.

5.2.1 Implementasi Halaman Home

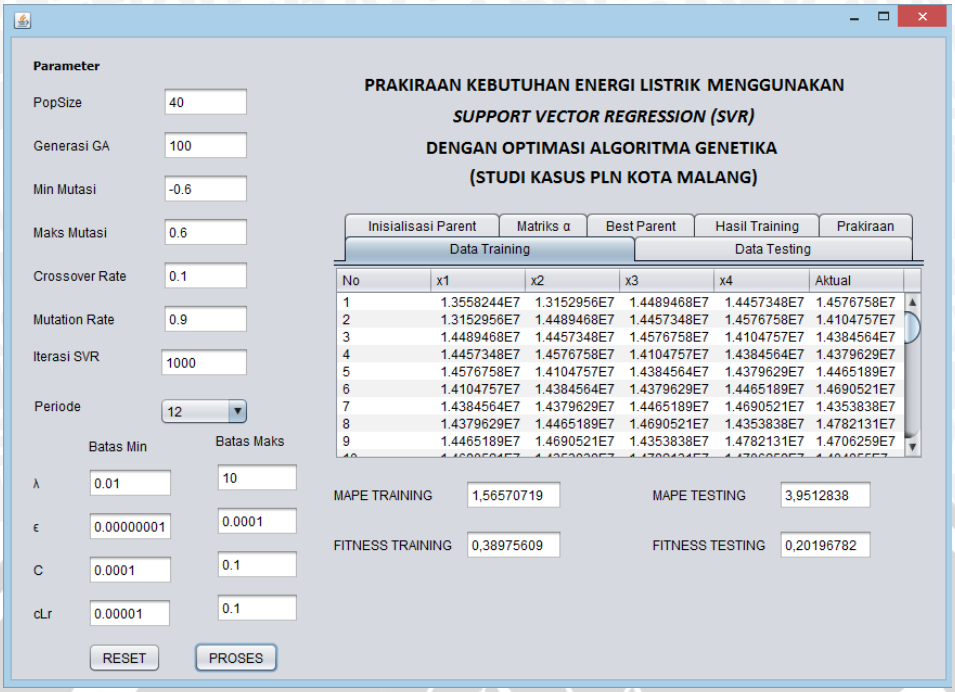
Halaman awal atau home page merupakan halaman default ketika aplikasi mulai dijalankan pertama kali. Terdapat button start untuk ke form selanjutnya dan melakukan input parameter, dan button about berisi tentang developer aplikasi. Gambar implementasi halaman awal ditunjukkan pada Gambar 5.1.



Gambar 5. 1 Implementasi Antarmuka Halaman Home

5.2.2 Implementasi Halaman Proses GASVR

Halaman Proses GASVR digunakan untuk melakukan *input* parameter yang digunakan yaitu jumlah iterasi SVR, GA, jumlah popSize, nilai mininimum dan maksimum mutasi, *crossover rate, mutation rate*, dan batas maksimum dan minimum parameter yang dioptimasi, terdapat *button*reset untuk mengosongkan nilai pada form yang tanpa perlu menghapus satu-persatu. Serta ada *button* proses untuk melakukan proses GASVR. Gambar implementasi halaman awal ditunjukkan pada Gambar 5.2.



Gambar 5. 2 Implementasi Antarmuka Halaman GASVR

5.2.3 Implementasi Halaman About

Halaman About merupakan halaman yang berisi tentang developer sistem Algoritma Genetika. Gambar implementasi halaman about ditunjukkan pada Gambar 5.3.

Gambar 5. 3 Implementasi Antarmuka Halaman About

BAB 6 PENGUJIAN DAN ANALISIS

Bab ini berisi hasil pengujian serta analisis dari hasil uji coba yang telah dilakukan dalam prakiraan kebutuhan energi listrik menggunakan SVR dengan optimasi algoritma genetika. Pada pengujian prakiraan kenutuhan energi listrik digunakan data training sebanyak 36 data yang diambil mulai bulan Juli 2010 hingga Juni 2013 dan data testing sebanyak 12 data yang diambil pada bulan Juli 2013 hingga Juni 2014.

6.1 Hasil dan Analisa Uji Coba *PopSize*

Untuk penujian jumlah *popSize* yang digunakan adalah 10, 20, 30, 40, 50, 60, 70, 80, 90, 100. Detail parameter yang digunakan pada proses pengujian *popSize* adalah sebagai berikut:

a. Range λ : 0.0001 - 10
b. Range ε : 0.00000001 - 0.0001
c. Range *C* : 0.0001 - 10
d. Range *cLr* : 0.000001 - 0.1
e. Generasi GA : 100
f. Iterasi SVR : 1000
g. *Crosover rate* : 0.4
h. *Mutation rate* : 0.6
i. Periode prakiraan : 12 bulan

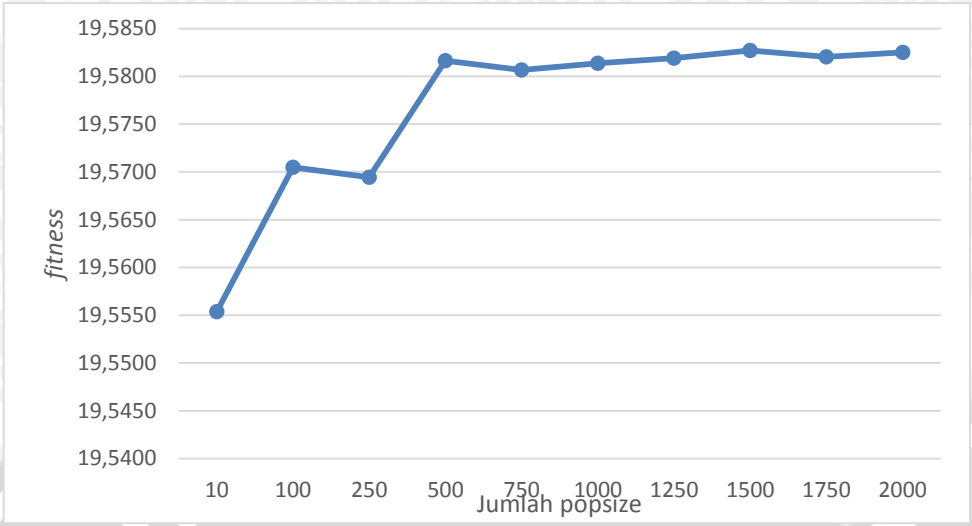
Pengujian *popSize* ini masing – masing dilakukan sebanyak 10 kali. Hasil pengujian dapat dilihat pada Tabel 6.1 berikut ini :

Tabel 6. 1 Hasil Uji Coba Jumlah *PopSize*

Banyak PopSize	Nilai *Fitness*										Rata-rata nilai *fitness*
	Percobaan ke-*i*										
	1	2	3	4	5	6	7	8	9	10	
10	19.5648	19.5742	19.5607	19.5725	19.5463	19.5570	19.5499	19.5185	19.5742	19.5356	19.5554
20	19.5682	19.5798	19.5783	19.5664	19.5762	19.5818	19.5627	19.5774	19.5518	19.5622	19.5705
30	19.5130	19.5824	19.5772	19.5799	19.5796	19.5765	19.5598	19.5727	19.5819	19.5715	19.5695
40	19.5823	19.5824	19.5809	19.5772	19.5839	19.5821	19.5832	19.5807	19.5822	19.5816	19.5817
50	19.5812	19.5773	19.5829	19.5752	19.5809	19.5778	19.5825	19.5834	19.5832	19.5823	19.5807
60	19.5814	19.5809	19.5800	19.5826	19.5827	19.5822	19.5757	19.5831	19.5830	19.5819	19.5814
70	19.5836	19.5822	19.5822	19.5819	19.5825	19.5830	19.5825	19.5788	19.5816	19.5808	19.5819
80	19.5816	19.5838	19.5803	19.5834	19.5815	19.5835	19.5839	19.5821	19.5841	19.5827	19.5827
90	19.5833	19.5819	19.5836	19.5839	19.5831	19.5820	19.5816	19.5834	19.5801	19.5774	19.5820
100	**19.5835**	**19.5824**	**19.5828**	**19.5836**	**19.5814**	**19.5831**	**19.5810**	**19.5833**	**19.5809**	**19.5832**	**19.5825**

Hasil uji coba jumlah *popSize* terhadap rata-rata nilai fitness pada Tabel 6.1 akan diformulasikan dalam sebuah grafik untuk mempermudah melihat

perbedaan dari hasil pengujian banyaknya ukuran populasi terhadap nilai fitness. Grafik hasil uji coba ukuran populasi ditunjukkan pada Gambar 6.1



Gambar 6. 1 Grafik Hasil Pengujian Jumlah *PopSize*

Pada grafik Gambar 6.1 dapat dilihat bahwa jumlah *popSize* berpengaruh terhadap perubahan nilai *fitness*. Secara garis besar semakin besar jumlah *popSize* maka nilai fitness akan semakin membesar. Untuk penambahan ukuran populasi tidak memberikan nilai fitness signifikan tetapi menunjukkan nilai fitness yang cenderung sama dengan ukuran populasi yang diindikasikan sudah mencapai konvergen. Populasi yang terlalu sedikit akan menyebabkan sedikitnya individu yang belum cukup baik akan terpilih sehingga membuat nilai fitness juga akan kurang optimal. Sebaliknya, jika semakin banyak populasi belum tentu juga didapat nilai fitness yang cukup tinggi dari jumlah populasi yang lebih sedikit (Priandani, 2015). Rata-rata nilai *fitness* yang paling besar didapatkan pada nilai *popSize* 100 yaitu 19.583. Namun semakin banyak jumlah *popSize* maka akan semakin lama waktu komputasinya. Pada Gambar 6.1 grafik menunjukan bahwa nilai *fitness* mulai stabil pada popSize 40.

6.2 Hasil dan Analisa Uji Jumlah Iterasi SVR

Untuk pengujian jumlah iterasi SVR yang digunakan adalah 10, 50, 100, 500, 1000, 5000, 10000, 50000, 100000. Detail parameter yang digunakan pada proses pengujian jumlah iterasi SVR adalah sebagai berikut:

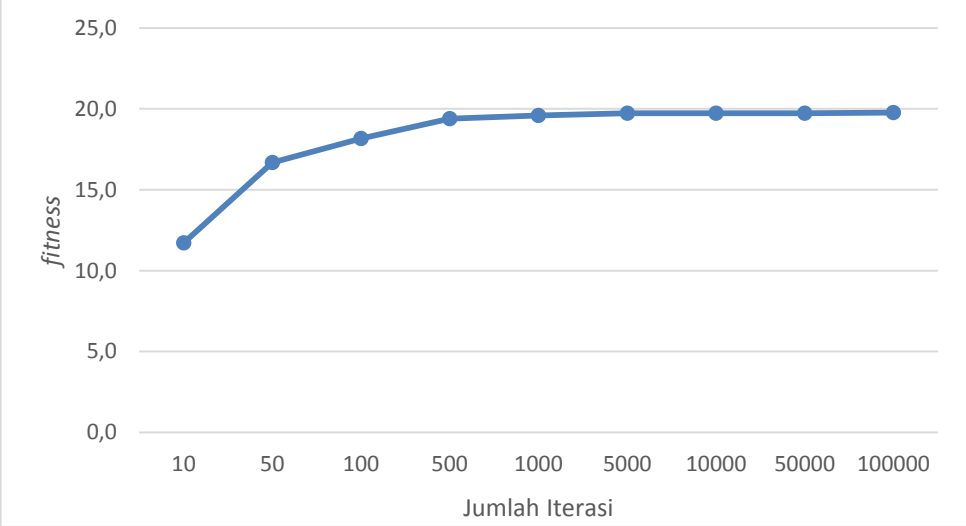
a. Range λ : 0.0001 - 10
b. Range ε : 0.00000001 - 0.0001
c. Range *C* : 0.0001 - 10
d. Range *cLr* : 0.000001 - 0.1
e. *popSize* : 40
f. Generasi GA : 100
g. *Crosover rate* : 0.4
h. *Mutation rate* : 0.6
i. Periode prakiraan : 12 bulan

Pengujian *popSize* ini masing – masing dilakukan sebanyak 10 kali. Hasil pengujian dapat dilihat pada Tabel 6.2 berikut ini :

Tabel 6. 2 Hasil Pengujian Jumlah Iterasi SVR

Iterasi	Nilai *Fitness*										Rata-rata nilai *fitness*
	Percobaan ke-*i*										
	1	2	3	4	5	6	7	8	9	10	
10	11.9052	11.9055	11.8964	11.8864	11.8358	11.7886	11.9013	10.1926	11.9055	11.8176	11.70348
50	16.6956	16.6816	16.6851	16.7042	16.6194	16.7026	16.6858	16.6313	16.7228	16.7026	16.68310
100	18.1661	18.1706	18.1633	18.1653	18.1641	18.1420	18.1705	18.1709	18.1670	18.1599	18.16396
500	19.3947	19.3920	19.3829	19.3949	19.3870	19.3897	19.3957	19.3958	19.3924	19.3903	19.39155
1000	19.5829	19.5820	19.5833	19.5826	19.5778	19.5797	19.5813	19.5800	19.5785	19.5824	19.58106
5000	19.7345	19.7295	19.7297	19.7309	19.7261	19.7391	19.7374	19.7354	19.7441	19.7441	19.73509
10000	19.7413	19.7308	19.7387	19.7416	19.7202	19.7400	19.7253	19.7385	19.7420	19.7403	19.73588
50000	19.7365	19.7285	19.7422	19.7457	19.7313	19.7445	19.7417	19.7318	19.7223	19.7377	19.73623
100000	**19.7223**	**19.6457**	**19.7392**	**20.3567**	**19.7216**	**19.7323**	**19.7239**	**19.6811**	**19.7049**	**19.7323**	**19.77601**

Hasil uji coba jumlah iterasi SVR terhadap rata-rata nilai fitness pada Tabel 6.2 akan diformulasikan dalam sebuah grafik untuk mempermudah melihat perbedaan dari hasil pengujian banyaknya ukuran populasi terhadap nilai fitness. Grafik hasil uji coba jumlah iterasi SVR ditunjukkan pada Gambar 6.2.



Gambar 6. 2 Grafik Hasil Pengujian Jumlah Iterasi SVR

Pada Gambar 6.2 dapat dilihat bahwa jumlah *iterasi* berperangaruh terhadap perubahan nilai fitness. Jumlah generasi/iterasi merepresentasikan lamanya proses pembelajaran yang dilakukan terhadap jaringan yang sedang diobservasi. Jumlah iterasi menentukan kapan proses pembelajaran dihentikan. Semakin besar nilai jumlah iterasi, maka semakin lama pula proses pembelajaran berlangsung. Jumlah iterasi yang terlalu sedikit, mengakibatkan *alpha* yang terbentuk bersifat terlalu general/umum (*overweigth*). Artinya kemampuan dalam mengenali pola

terlalu sedikit atau bahkan tidak ada sama sekali. Jumlah generasi/iterasi yang terlalu banyak, akan mengakibatkan jaringan mengalami kondisi overfit. Dikatakan bahwa kondisi overfit akan bagus dalam mengenali pola data pelatihan namun buruk dalam mengenali pola dari data pengujian (Alfredo, 2015). Rata-rata nilai *fitness* yang paling besar didapatkan pada nilai jumlah iterasi sebesar 100000 yaitu 20.357. Pada Gambar 6.1 grafik menunjukan bahwa nilai *fitness* mulai stabil pada jumlah iterasi 1000.

6.3 Hasil dan Analisa Uji Coba Kombinasi *Crossover Rate* dan *Mutation Rate*

Untuk penujian nilai *crossover rate* dan *mutation rate* yang digunakan adalah 0;1, 0.1;0.9, 0.2;0.8, 0.3;0.7, 0.4;0.6, 0.5;0.5, 0.6;0.4, 0.7;0.3, 0.8;0.2, 0.9;0.1, 1;0. Detail parameter yang digunakan pada proses pengujian nilai *crossover rate* dan *mutation rate* adalah sebagai berikut:

a. Range λ : 0.0001 - 10
b. Range ε : 0.00000001 - 0.0001
c. Range *C* : 0.0001 - 10
d. Range *cLr* : 0.000001 - 0.1
e. *popSize* : 40
f. Iterasi SVR : 1000
g. Generasi GA : 100
h. Periode prakiraan : 12 bulan

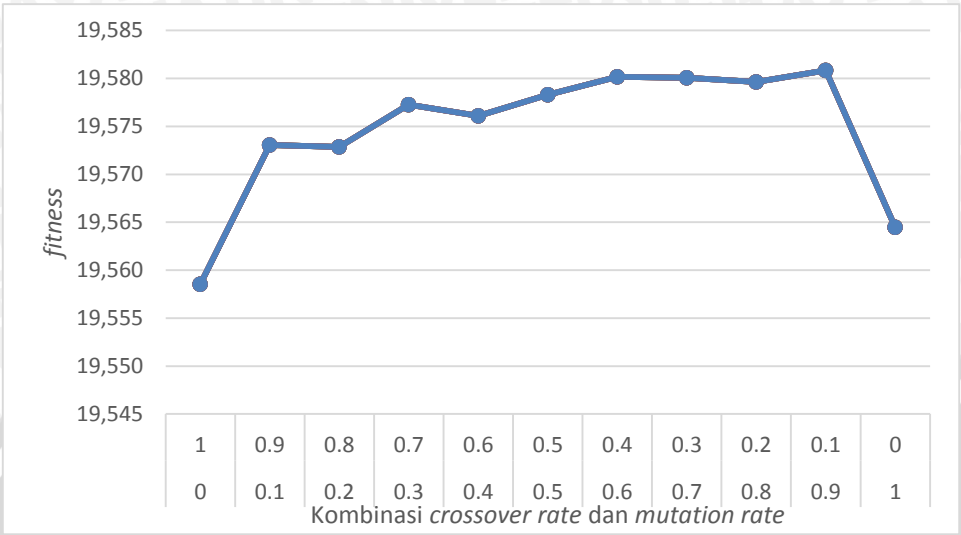
Pengujian *popSize* ini masing – masing dilakukan sebanyak 10 kali. Hasil pengujian dapat dilihat pada Tabel 6.3 berikut ini :

Tabel 6. 3 Hasil Pengujian Kombinasi *Crossover Rate* dan *Mutation Rate*

cr	*mr*	Nilai *Fitness* Percobaan ke-*i*										Rata-rata nilai *fitness*
		1	2	3	4	5	6	7	8	9	10	
1	0	19.5719	19.5751	19.5399	19.5809	19.5612	19.5759	19.5231	19.5480	19.5714	19.5381	19.55855
0.9	0.1	19.5804	19.5680	19.5786	19.5706	19.5797	19.5787	19.5748	19.5632	19.5748	19.5619	19.57308
0.8	0.2	19.5802	19.5645	19.5649	19.5774	19.5652	19.5626	19.5784	19.5831	19.5711	19.5813	19.57285
0.7	0.3	19.5757	19.5823	19.5687	19.5825	19.5739	19.5709	19.5835	19.5778	19.5819	19.5757	19.57728
0.6	0.4	19.5661	19.5617	19.5816	19.5802	19.5811	19.5792	19.5817	19.5744	19.5726	19.5822	19.57609
0.5	0.5	19.5828	19.5818	19.5722	19.5802	19.5806	19.5833	19.5754	19.5815	19.5777	19.5678	19.57831
0.4	0.6	19.5828	19.5772	19.5805	19.5811	19.5776	19.5788	19.5787	19.5833	19.5812	19.5805	19.58017
0.3	0.7	19.5812	19.5769	19.5776	19.5833	19.5780	19.5815	19.5841	19.5815	19.5772	19.5796	19.58009
0.2	0.8	19.5798	19.5821	19.5822	19.5810	19.5776	19.5824	19.5812	19.5794	19.5696	19.5809	19.57962
0.1	**0.9**	**19.5821**	**19.5827**	**19.5824**	**19.5813**	**19.5783**	**19.5818**	**19.5841**	**19.5746**	**19.5801**	**19.5811**	**19.58085**
1	0	19.5680	19.5486	19.5541	19.5710	19.5563	19.5618	19.5744	19.5800	19.5759	19.5547	19.56448

Hasil uji coba kombinasi *crossover rate* dan *mutation rate* terhadap rata-rata nilai fitness pada Tabel 6.3 akan diformulasikan dalam sebuah grafik untuk

mempermudah melihat perbedaan dari hasil pengujian banyaknya ukuran populasi terhadap nilai fitness. Grafik hasil uji kombinasi *crossover rate* dan *mutation rate* ditunjukkan pada Gambar 6.3.



Gambar 6. 3 Grafik Hasil Pengujian Kombinasi Crossover Rate dan Mutaion Rate

Pada Gambar 6.3 dapat dilihat bahwa kombinasi *crossover rate* dan *mutation rate* berperangaruh terhadap perubahan nilai fitness. Rata-rata nilai *fitness* yang paling besar didapatkan pada nilai *crossover rate* 0.1 dan *mutation rate* 0.9 yaitu 19.5809. Semakin besar nilai *crossover rate* dan semakin kecil nilai *mutation rate* maka nilai *fitness* akan semakin membesar. *Crossover rate* yang tinggi dan *mutation rate* yang rendah digunakan, maka akan mengalami penurunan kemampuan untuk menjaga diversitas pupolasi. *Crossover rate* yang tinggi akan menghasilkan offspring yang mempunyai kemiripan yang tinggi dengan induknya. Hal ini menyebabkan GA mengalami konvergensi dini hanya dalam beberapa generasi dan kehilangan kesempatan untuk mengeksplorasi area lain dalam ruang pencarian (Mahmudy, 2013).

6.4 Hasil dan Analisa Uji Coba *Range Lambda* (λ)

Untuk penujian nilai *range* λ yang digunakan adalah 0.0001-1, 0.001-1, 0.01-1, 0.0001-10, 0.001-10, 0.01-10, 0.0001-100, 0.001-100, 0.01-100. Detail parameter yang digunakan pada proses pengujian nilai *range* λ adalah sebagai berikut:

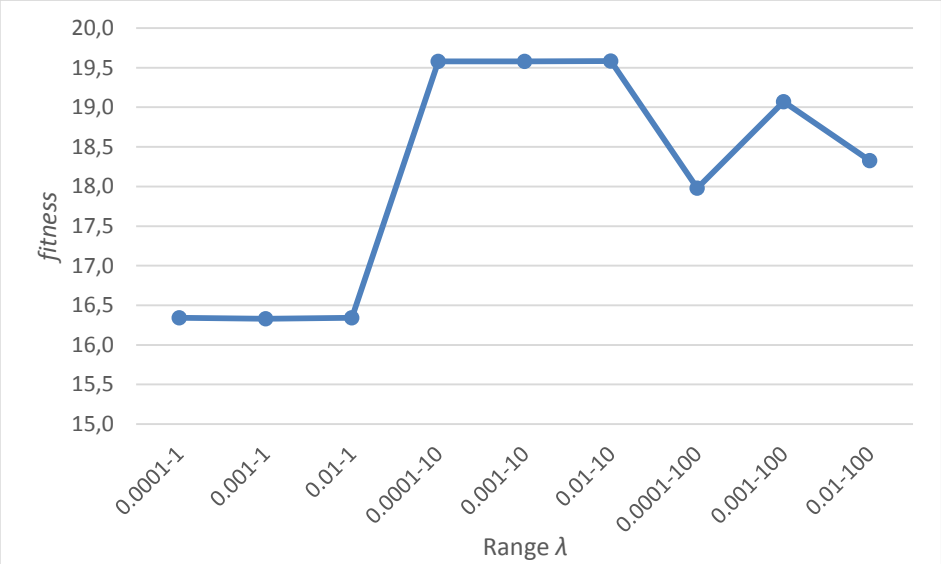
a. Range ε : 0.00000001 - 0.0001
b. Range *C* : 0.0001 - 10
c. Range *cLr* : 0.000001 - 0.1
d. *popSize* : 40
e. Iterasi SVR : 1000
f. Generasi GA : 100
g. *Crossover rate* : 0.1
h. *Mutation rate* : 0.9
i. Periode prakiraan : 12 bulan

Pengujian *popSize* ini masing – masing dilakukan sebanyak 10 kali. Hasil pengujian dapat dilihat pada Tabel 6.4 berikut ini :

Tabel 6. 4 Hasil Pengujian Nilai *Crossover Rate* dan *Mutation Rate*

min	*max*	Nilai *Fitness*										Rata-rata nilai *fitness*
		Percobaan ke-*i*										
		1	2	3	4	5	6	7	8	9	10	
0.0001	1	16.3424	16.3429	16.3429	16.3429	16.3391	16.3427	16.3424	16.3429	16.3429	16.3426	16.34238
	10	16.3430	16.3415	16.3424	16.3416	16.3400	16.2273	16.3427	16.3404	16.3428	16.3395	16.33011
	100	16.3404	16.3400	16.3422	16.3422	16.3429	16.3428	16.3403	16.3389	16.3429	16.3428	16.34155
0.001	1	19.5771	19.5825	19.5817	19.5822	19.5819	19.5809	19.5821	19.5818	19.5821	19.5804	19.58128
	10	19.5826	19.5839	19.5829	19.5831	19.5835	19.5810	19.5794	19.5778	19.5597	19.5752	19.57892
	100	**19.5812**	**19.5799**	**19.5801**	**19.5814**	**19.5832**	**19.5818**	**19.5840**	**19.5807**	**19.5821**	**19.5835**	**19.58180**
0.01	1	19.5812	19.5828	15.9404	14.8865	19.5493	19.5834	14.8097	16.7179	19.5728	19.5837	17.98076
	10	19.5799	19.5742	19.5803	19.4003	19.5800	19.5836	14.6850	19.5789	19.5821	19.5668	19.07110
	100	19.5835	19.5832	19.5818	17.1100	15.5715	16.6122	19.5839	16.4736	19.5811	19.5798	18.32607

Hasil uji coba *range* λ terhadap rata-rata nilai fitness pada Tabel 6.4 akan diformulasikan dalam sebuah grafik untuk mempermudah melihat perbedaan dari hasil pengujian banyaknya ukuran populasi terhadap nilai fitness. Grafik hasil uji *range* λ ditunjukkan pada Gambar 6.4.



Gambar 6. 4 Grafik Hasil Pengujian *Range* λ

Pada Gambar 6.4 dapat dilihat bahwa *range* λ cukup berperangaruh terhadap perubahan nilai *fitness*. Nilai fitness meningkat tajam pada saat nilai maksimum 10, namun kembali turun saat nila maksmum membesar di 100. Rata-rata nilai *fitness* yang paling besar didapatkan pada saat nilai minimum 0.01 dan nilai maksimum 10 yaitu sebesar 19.582.

6.5 Hasil dan Analisa Uji Coba *Range Epsilon* (*ε*)

Untuk penujian nilai *range* *λ* yang digunakan adalah 0.00000001-0.0001, 0.0000001-0.0001, 0.000001-0.0001, 0.00000001-0.001, 0.0000001-0.001, 0.000001-0.001, 0.00000001-0.01, 0.0000001-0.01, 0.000001-0.01. Detail parameter yang digunakan pada proses pengujian nilai *range* *ε* adalah sebagai berikut:

a. Range *λ* : 0.01 - 10
b. Range *C* : 0.0001 - 10
c. Range *cLr* : 0.000001 - 0.1
d. *popSize* : 40
e. Iterasi SVR : 1000
f. Generasi GA : 100
g. *Crossover rate* : 0.1
h. *Mutation rate* : 0.9
i. Periode prakiraan : 12 bulan

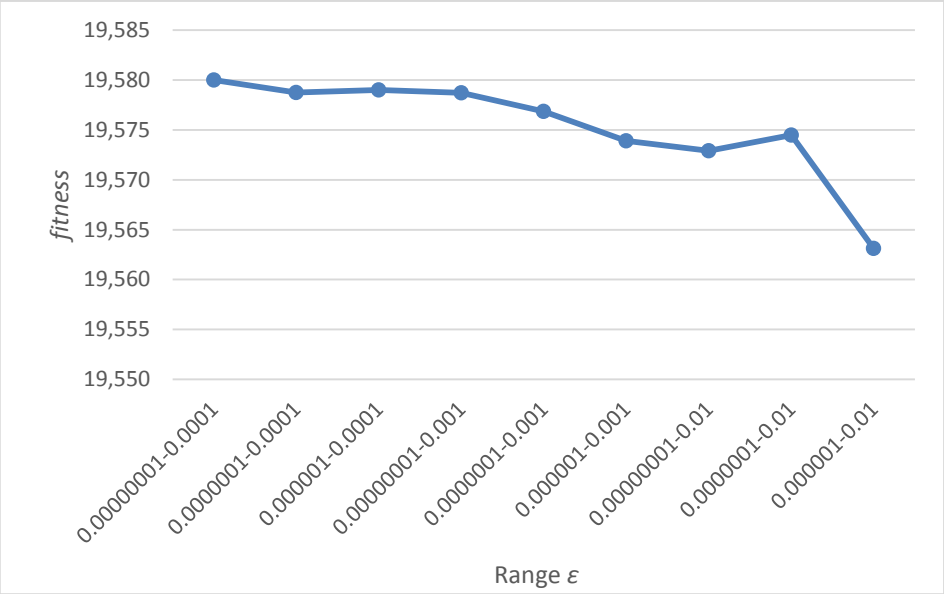
Pengujian *range* *ε* ini masing – masing dilakukan sebanyak 10 kali. Hasil pengujian dapat dilihat pada Tabel 6.5 berikut ini :

Tabel 6. 5 Hasil Pengujian *Range ε*

min	***max***	**Nilai *Fitness***										**Rata-rata nilai *fitness***
		Percobaan ke-*i*										
		1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	
1E-08	**0.0001**	**19.5694**	**19.5823**	**19.5822**	**19.5799**	**19.5823**	**19.5812**	**19.5796**	**19.5812**	**19.5811**	**19.5810**	**19.58001**
	0.001	19.5810	19.5791	19.5719	19.5743	19.5789	19.5757	19.5822	19.5803	19.5809	19.5831	19.57874
	0.01	19.5738	19.5834	19.5814	19.5823	19.5804	19.5760	19.5821	19.5753	19.5734	19.5819	19.57901
1E-07	0.0001	19.5812	19.5821	19.5820	19.5821	19.5747	19.5682	19.5783	19.5809	19.5791	19.5786	19.57872
	0.001	19.5778	19.5734	19.5816	19.5831	19.5785	19.5827	19.5588	19.5833	19.5748	19.5746	19.57686
	0.01	19.5787	19.5739	19.5670	19.5829	19.5701	19.5832	19.5820	19.5508	19.5776	19.5730	19.57391
1E-06	0.0001	19.5625	19.5663	19.5799	19.5707	19.5696	19.5799	19.5611	19.5834	19.5729	19.5826	19.57290
	0.001	19.5621	19.5822	19.5803	19.5783	19.5760	19.5796	19.5749	19.5750	19.5773	19.5592	19.57449
	0.01	19.5801	19.5702	19.5617	19.5807	19.5831	19.5468	19.5673	19.5404	19.5216	19.5795	19.56314

Hasil uji coba *range* *ε* terhadap rata-rata nilai *fitness* pada Tabel 6.5 akan diformulasikan dalam sebuah grafik untuk mempermudah melihat perbedaan dari hasil pengujian banyaknya ukuran populasi terhadap nilai fitness. Grafik hasil uji *range* *ε* ditunjukkan pada Gambar 6.5.

Pada Gambar 6.5 dapat dilihat bahwa *range* *ε* berperangaruh terhadap perubahan nilai fitness. Rata-rata nilai *fitness* yang paling besar didapatkan pada saat nilai minimum 0.00000001 dan nilai maksimum 0.0001 yaitu sebesar 19.58001. Semakin besar nilai epsilon ε, maka semakin jelek hasil regresinya sehingga menghasilkan nilai *fitness* yang semakin kecil.



Gambar 6. 5 Grafik Hasil Pengujian *Range ε*

6.6 Hasil dan Analisa Uji Coba *Range C*

Untuk penujian nilai *range C* yang digunakan adalah 0.0001-0.1, 0.001-0.1, 0.01-0.1, 0.0001-0.1, 0.001-0.1, 0.01-0.1, 0.0001-0.1, 0.001-0.1, 0.01-0.1. Detail parameter yang digunakan pada proses pengujian nilai *range ε* adalah sebagai berikut:

a. Range λ : 0.01 - 10
b. Range ε : 0.00000001 – 0.0001
c. Range *cLr* : 0.000001 - 0.1
d. *popSize* : 40
e. Iterasi SVR : 1000
f. Generasi GA : 100
g. *Crossover rate* : 0.1
h. *Mutation rate* : 0.9
i. Periode prakiraan : 12 bulan

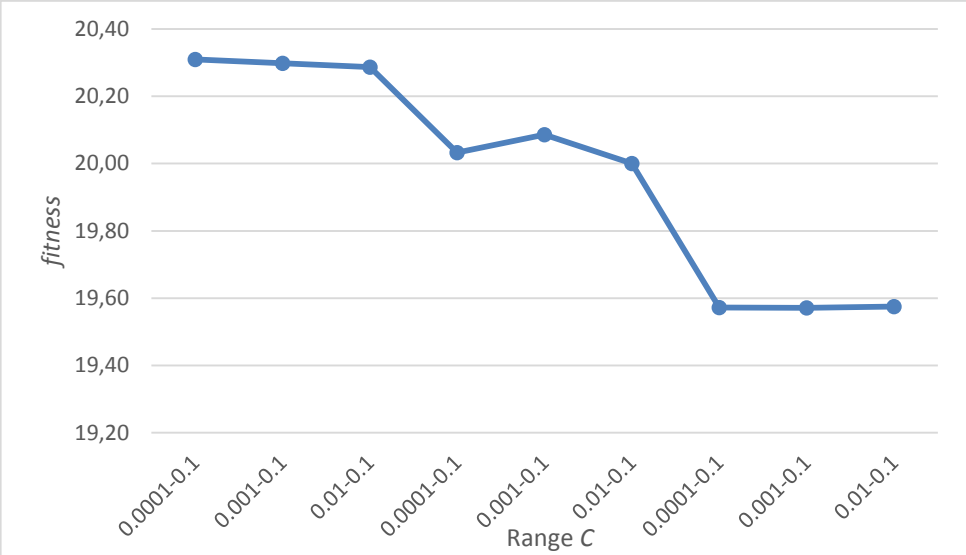
Pengujian *range C* ini masing – masing dilakukan sebanyak 10 kali. Hasil pengujian dapat dilihat pada Tabel 6.6 berikut ini :

Tabel 6. 6 Hasil Pengujian *Range C*

min	***max***	**Nilai *Fitness***										**Rata-rata nilai *fitness***
		Percobaan ke-*i*										
		1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	
0.0001	0.1	**20.3508**	**20.2565**	**20.3441**	**20.3416**	**20.3223**	**20.3654**	**20.2590**	**20.2609**	**20.3004**	**20.2948**	**20.30957**
	1	20.3149	20.3491	20.3371	20.3131	20.2703	20.3109	20.2939	20.2480	20.2945	20.2469	20.29787
	10	20.3247	20.3162	20.2728	20.2297	20.3350	20.1977	20.3068	20.3030	20.2817	20.2946	20.28623
0.001	0.1	20.3953	19.5817	20.0641	20.3836	19.5713	20.3509	19.5787	20.4020	20.4288	19.5664	20.03230
	1	20.4066	20.3226	20.4227	19.8614	20.3408	19.5686	20.2731	19.5804	20.3681	19.7151	20.08595

	10	19.7070	20.3016	20.0503	20.4205	19.5831	20.1505	20.2981	19.5614	19.5665	20.3654	20.00043
0.01	0.1	19.5698	19.5765	19.5388	19.5816	19.5793	19.5754	19.5783	19.5773	19.5762	19.5694	19.57226
	1	19.5711	19.5804	19.5606	19.5643	19.5543	19.5723	19.5820	19.5822	19.5618	19.5798	19.57089
	10	19.5798	19.5683	19.5806	19.5836	19.5531	19.5788	19.5767	19.5804	19.5798	19.5679	19.57490

Hasil uji coba *range C* terhadap rata-rata nilai fitness pada Tabel 6.6 akan diformulasikan dalam sebuah grafik untuk mempermudah melihat perbedaan dari hasil pengujian banyaknya ukuran populasi terhadap nilai fitness. Grafik hasil uji *range C* ditunjukkan pada Gambar 6.6.



Gambar 6. 6 Grafik Hasil Pengujian *Range C*

Pada Gambar 6.6 dapat dilihat bahwa *range C* berpengaruh terhadap perubahan nilai fitness. Rata-rata nilai *fitness* yang paling besar didapatkan pada saat nilai minimum 0.0001 dan nilai maksimum 0.1 yaitu sebesar 0.20309. Semakin besar nilai *epsilon C*, maka semakin jelek hasil regresinya sehingga menghasilkan nilai *fitness* yang semakin kecil.

6.7 Hasil dan Analisa Uji Coba *Range cLr*

Untuk penujian nilai *range cLr* yang digunakan adalah 0.0001-0.1, 0.001-0.1, 0.01-0.1, 0.0001-0.1, 0.001-0.1, 0.01-0.1, 0.0001-0.1, 0.001-0.1, 0.01-0.1. Detail parameter yang digunakan pada proses pengujian nilai *range ε* adalah sebagai berikut:

a. *Range λ* : 0.01 - 10
b. *Range ε* : 0.00000001 – 0.0001
c. *Range C* : 0.0001 - 0.1
d. *popSize* : 40
e. Iterasi SVR : 1000
f. Generasi GA : 100
g. *Crossover rate* : 0.1
h. *Mutation rate* : 0.9

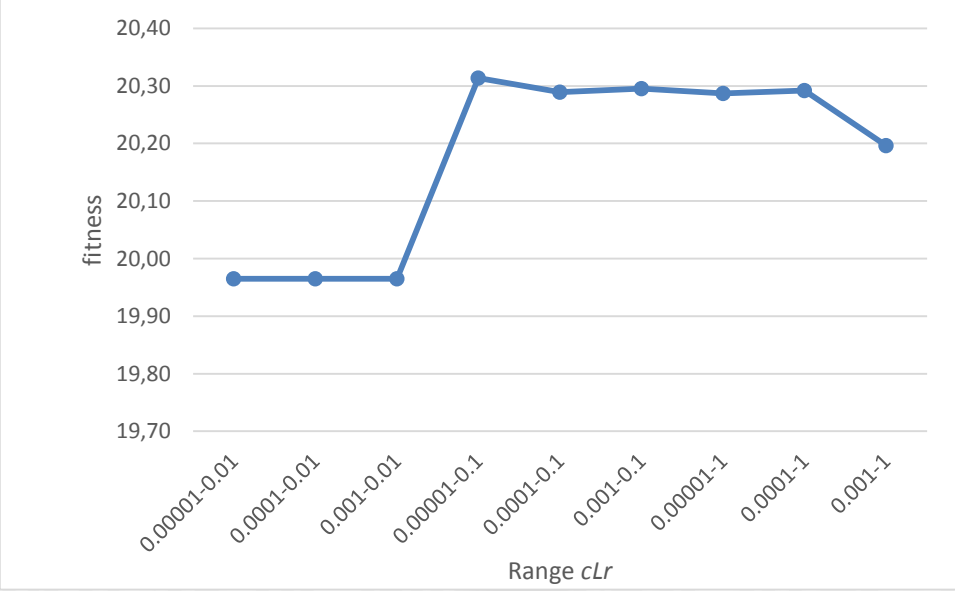
i. Periode prakiraan : 12 bulan

Pengujian *range cLr* ini masing – masing dilakukan sebanyak 10 kali. Hasil pengujian dapat dilihat pada Tabel 6.7 berikut ini :

Tabel 6. 7 Hasil Pengujian *Range cLr*

min	***max***	**Nilai *Fitness***										**Rata-rata nilai *fitness***
		Percobaan ke-*i*										
		1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	
0.00001	0.01	19.9651	19.9651	19.9651	19.9651	19.9651	19.9651	19.9651	19.9651	19.9651	19.9651	19.96510
	0.1	19.9651	19.9651	19.9651	19.9651	19.9651	19.9651	19.9650	19.9650	19.9651	19.9651	19.96508
	1	19.9651	19.9651	19.9651	19.9651	19.9651	19.9651	19.9651	19.9651	19.9651	19.9651	19.96511
0.**0001**	**0.01**	**20.3421**	**20.3057**	**20.3328**	**20.3552**	**20.2905**	**20.2570**	**20.3042**	**20.3469**	**20.3114**	**20.2897**	**20.31356**
	0.1	20.3008	20.2914	20.3170	20.2791	20.2987	20.2824	20.1993	20.2882	20.3302	20.3068	20.28940
	1	20.2813	20.3255	20.3374	20.2618	20.2699	20.3252	20.2819	20.2887	20.2979	20.2838	20.29535
0.001	0.01	20.3024	20.2805	20.2681	20.2826	20.2754	20.2740	20.2602	20.2817	20.3122	20.3353	20.28724
	0.1	20.3201	20.3470	20.2612	20.2545	20.3203	20.3222	20.3306	20.2997	20.1961	20.2690	20.29206
	1	20.3479	20.2589	20.2685	20.2217	20.2849	20.2780	20.3344	19.6546	20.0678	20.2487	20.19654

Hasil uji coba *range cLr* terhadap rata-rata nilai fitness pada Tabel 6.7 akan diformulasikan dalam sebuah grafik untuk mempermudah melihat perbedaan dari hasil pengujian banyaknya ukuran populasi terhadap nilai *fitness*. Grafik hasil uji *range cLr* ditunjukkan pada Gambar 6.7.



Gambar 6. 7 Grafik Hasil Pengujian *Range cLr*

Pada Gambar 6.7 dapat dilihat bahwa *range cLr* berpengaruh terhadap perubahan nilai fitness. Rata-rata nilai *fitness* yang paling besar didapatkan pada saat nilai minimum 0.00001 dan nilai maksimum 0.1 yaitu sebesar 20.3136. Semakin besar nilai *cLr*, maka semakin baik hasil regresinya sehingga menghasilkan

nilai *fitness* yang semakin besar. Namun nilai *fitness* mulai stabil pada saar range *cLr* antara 0.00001 dan 0.1.

6.8 Hasil dan Analisa Uji Coba *Jumlah Generasi GA*

Untuk pengujian jumlah generasi GA yang digunakan adalah 10, 100, 250, 500, 750, 1000, 1250, 1500, 1750, 2000. Detail parameter yang digunakan pada proses pengujian nilai *range ε* adalah sebagai berikut:

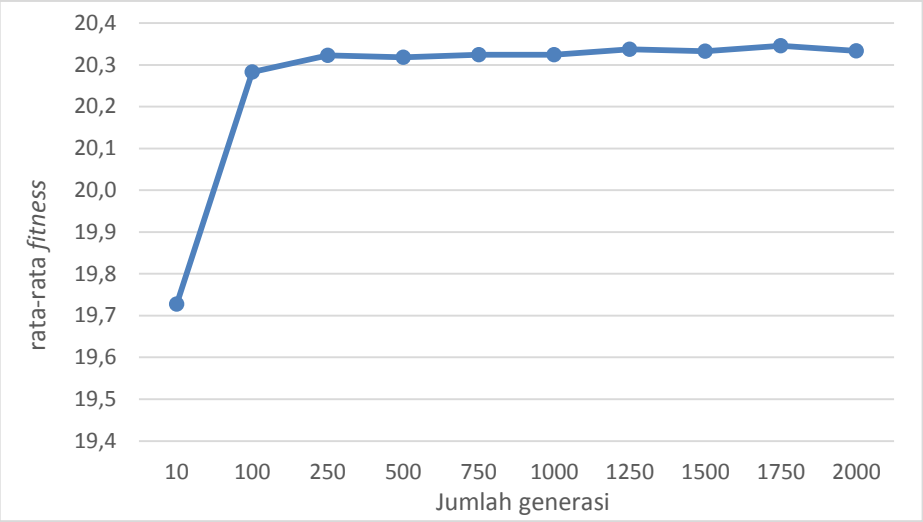
a. *Range λ* : 0.01 - 10
b. *Range ε* : 0.00000001 – 0.0001
c. *Range C* : 0.0001 - 0.1
d. *Range cLr* : 0.00001 – 0.1
e. *popSize* : 40
f. Iterasi SVR : 1000
g. *Crossover rate* : 0.1
h. *Mutation rate* : 0.9
i. Periode prakiraan : 12 bulan

Pengujian jumlah generasi GA ini masing – masing dilakukan sebanyak 10 kali. Hasil pengujian dapat dilihat pada Tabel 6.8 berikut ini :

Tabel 6. 8 Hasil Pengujian *Jumlah Generasi GA*

Generasi	Nilai *Fitness*										Rata-rata nilai *fitness*
	Percobaan ke-*i*										
	1	2	3	4	5	6	7	8	9	10	
10	19.9050	20.1684	19.8973	19.9311	20.0500	19.7260	19.6338	19.6797	18.3869	19.8970	19.72753
100	20.2741	20.3269	20.3011	20.2662	20.3038	20.2973	20.2338	20.2895	20.2824	20.2516	20.28266
250	20.2910	20.3204	20.3275	20.3051	20.2817	20.2982	20.3846	20.3312	20.3449	20.3386	20.32230
500	20.2876	20.2200	20.3069	20.3125	20.3204	20.3694	20.3416	20.3558	20.3388	20.3297	20.31829
750	20.3310	20.3607	20.2841	20.3133	20.3281	20.2524	20.3277	20.3363	20.3641	20.3429	20.32406
1000	20.2786	20.3388	20.3306	20.3236	20.3758	20.3448	20.2901	20.3269	20.2740	20.3584	20.32414
1250	20.2610	20.3540	20.3187	20.3297	20.3422	20.3348	20.3850	20.3760	20.3683	20.3027	20.33724
1500	20.3466	20.3665	20.3508	20.2812	20.3540	20.3647	20.3532	20.2633	20.3247	20.3189	20.33240
1750	**20.3297**	**20.2715**	**20.3383**	**20.3354**	**20.3969**	**20.4008**	**20.3476**	**20.3467**	**20.3381**	**20.3548**	**20.34599**
2000	20.3522	20.2789	20.3382	20.3617	20.3032	20.3238	20.3335	20.3490	20.3177	20.3765	20.33347

Hasil uji coba jumlah generasi GA terhadap rata-rata nilai *fitness* pada Tabel 6.8 akan diformulasikan dalam sebuah grafik untuk mempermudah melihat perbedaan dari hasil pengujian banyaknya ukuran populasi terhadap nilai fitness. Grafik hasil uji jumlah generasi GA ditunjukkan pada Gambar 6.8.



Gambar 6. 8 Grafik Hasil Pengujian Jumlah Generasi GA

Pada Gambar 6.8 dapat dilihat bahwa *jumlah generasi GA* berpengaruh terhadap perubahan nilai fitness. Rata-rata nilai *fitness* yang paling besar didapatkan pada saat jumlah generasi sebanyak 1750 yaitu sebesar 20.346. Nilai *fitness* terendah terdapat pada jumlah generasi terendah. Hal ini disebabkan karena algoritma genetika belum dapat memproses secara optimal karena kurangnya jumlah generasi. Jumlah generasi yang terlalu banyak, akan menyebabkan waktu proses yang menjadi lebih lama dan hasil nilai fitness yang dihasilkan juga belum tentu jauh lebih baik dari generasi yang lebih sedikit. Namun cenderung menunjukkan peningkatan nilai *fitness* karena memberikan eksplorasi terhadap ruang pencarian yang lebih besar (Subanar, 2010). Nilai *fitness* pada pengujian ini mulai konstan pada generasi ke 250.

6.9 Hasil dan Analisa Uji Coba Jumlah Periode Prakiraan

Untuk pengujian jumlah periode prakiraan yang digunakan adalah 3, 4, 5, 6, 7, 8, 9, 10, 11, dan 12 bulan. Detail parameter yang digunakan pada proses pengujian jumlah periode prakiraan adalah sebagai berikut:

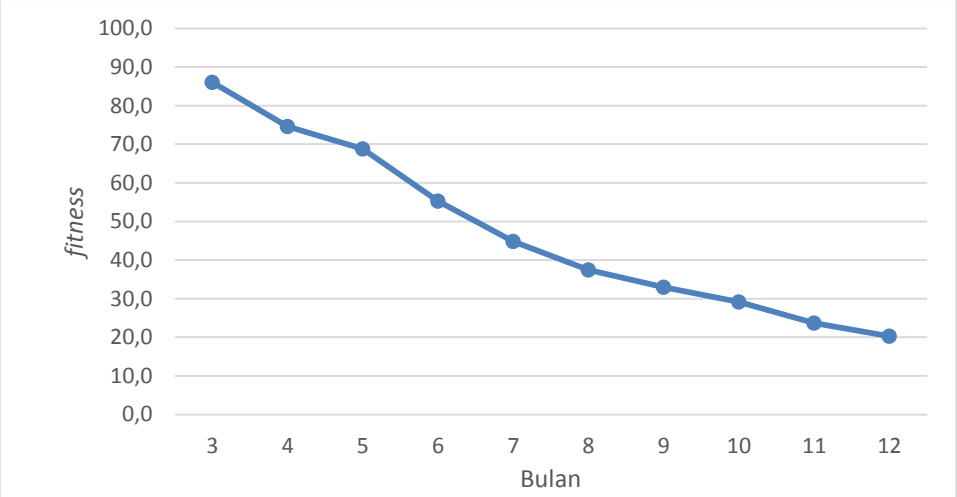
a. *Range λ* : 0.01 - 10
b. *Range ε* : 0.00000001 – 0.0001
c. *Range C* : 0.0001 - 0.1
d. *Range cLr* : 0.00001 – 0.1
e. *popSize* : 40
f. Iterasi SVR : 1000
g. Generasi GA : 250
h. *Crossover rate* : 0.1
i. *Mutation rate* : 0.9

Pengujian jumlah periode prakiraan ini masing – masing dilakukan sebanyak 10 kali. Hasil pengujian jumlah periode prakiraan dapat dilihat pada Tabel 6.9 berikut ini :

Tabel 6. 9 Hasil Pengujian Jumlah Periode Prakiraan

Periode (bulan)	Nilai *Fitness*										Rata-rata nilai *fitness*
	Percobaan ke-*i*										
	1	2	3	4	5	6	7	8	9	10	
3	**85.9909**	**86.0777**	**86.0302**	**86.0528**	**86.0112**	**85.9514**	**86.0192**	**85.9747**	**86.0014**	**86.0166**	**86.01261**
4	74.6195	74.4928	74.7074	74.7014	74.7080	74.5521	74.6689	74.6565	74.7096	74.4618	74.62780
5	68.7586	68.8905	68.8284	68.8715	68.6529	68.7565	68.9138	68.9476	68.9003	68.7974	68.83177
6	55.2628	55.3071	55.2297	55.2556	55.0568	55.3111	55.2479	55.3541	55.2653	55.2641	55.25546
7	44.9381	44.9956	44.9206	45.0157	44.9602	44.7596	44.9895	44.8856	44.0235	45.0056	44.84939
8	37.4444	37.5065	37.4982	37.4004	37.1236	37.4126	37.4847	37.5184	37.4066	37.4073	37.42027
9	33.0117	33.0293	33.0129	33.0130	33.0231	32.9987	32.9973	33.0154	32.9635	32.9976	33.00625
10	29.1791	29.2229	29.1490	29.2440	29.1177	29.1148	29.1183	29.1837	29.1380	29.1731	29.16406
11	23.6885	23.6827	23.6758	23.6475	23.6366	23.7016	23.6733	23.6431	23.6780	23.6139	23.66411
12	20.3255	20.3024	20.3298	20.2832	20.2819	20.3400	20.3199	20.3471	20.3080	20.2999	20.31378

Hasil uji coba jumlah periode prakiraan terhadap rata-rata nilai fitness pada Tabel 6.9 akan diformulasikan dalam sebuah grafik untuk mempermudah melihat perbedaan dari hasil pengujian banyaknya ukuran populasi terhadap nilai fitness. Grafik hasil uji jumlah periode prakiraan ditunjukkan pada Gambar 6.9.



Gambar 6. 9 Grafik Hasil Pengujian Jumlah Periode Prakiraan

Pada Gambar 6.9 dapat dilihat bahwa jumlah periode yang diprakiraan berpengaruh terhadap perubahan nilai fitness. Rata-rata nilai *fitness* yang paling besar didapatkan pada saat jumlah periode 3 bulan yaitu sebesar 86.013. Semakin besar jumlah periode yang diprakirakan, maka semakin kecil nilai *fitness*-nya. Ini membuktikan bahwa metode *Support Vector Regression* (SVR) tidak cocok untuk peramalan jangka jauh.

BAB 7 PENUTUP

7.1 Kesimpulan

Berikut adalah kesimpulan yang dapat diambil dari hasil yang telah didapatkan dari perancangan, implementasi dan pengujian yang telah dilakukan yaitu :

1. Proses awal yang harus dilakukan dalam mengimplementasikan metode SVR dengan algoritma genetika untuk prakiraan kebutuhan listrik Kota Malang adalah menentukan data latih dan data uji yang akan digunakan, peramalan yang digunakan adalah peralaman dengan data secara sekuensial, untuk mendapatkan parameter yang optimal digunakan metode algoritma genetika ketika fase training, sedangkan untuk *fase testing* digunakan metode SVR tanpa algoritma genetika dengan menggunakan parameter yang optimal ketika *fase training*.
2. Untuk mengukur *error rate* dari solusi untuk permasalahan peramalan nilai tukar ini digunakan perhitungan nilai fitness yang didapatkan dari rata- rata nilai error dari prakiraan kebutuhan listrik itu sendiri. Berdasarkan pengujian yang telah dilakukan, dengan data training = 36 bulan, *popSize* = 40, iterasi = 1000, *crossover rate* = 0.1, *mutation rate* = 0.9, λ = 0.01-10, ε = 0.00000001-0.0001, C = 0.0001-0.1, cLr = 0.00001-0.1, generasi = 1750, dan data testing = 3 bulan, nilai error terkecil didapatkan yaitu sebesar 0.161741.

7.2 Saran

Berikut merupakan saran yang dapat digunakan untuk penelitian selanjutnya yaitu :

1. Untuk penelitian selanjutnya, peneliti dapat menambahkan meode kernel yang digunakan untuk mendapatkan hasil yang optimal. Misalnya, seperti kernel linier, kernel polinomial, kernel sigmoid, dan lain sebagainya.
2. Penambahan parameter algoritma genetika untuk diuji untuk hasil yang lebih optimal. Misalnya nilai minimum dan makasimum mutasi.
3. Penambahan penggunaan jumlah data latih dan data uji dan juga variasi percobaan.

DAFTAR PUSTAKA

Ahmad, S., & Latif, H., 2011. Forecasting on the Crude Palm Oil and Kernel Palm Production: Seasonal ARIMA Approach. 2011 IEEE Colloquium on Humanities, Science and Engineering Research (CHUSER 2011), Dec 5-6 2011, Penang.

Ahmed, H. & Glasgow, J., 2012. *Swarm Intelligence: Concepts, Models and Applications*. Ontario: School of Computing, Queen's University, Kingston.

Alfredo, Jondri & Rismala R., 2014. Prediksi Harga Saham menggunakan *Support Vector Regression* dan *Firefly Algorithm*. Bandung: Departemen Informatika, Universitas Telkom.

Alves, G. & Silva, D., 2013. Data Envelopment Analysis for Selection of the Fitness Function in Evolutionary Algorithms Applied to Time Series Forecasting Problem. 2013 BRICS Congress on Computational Intelligence & 11th Brazilian Congress on Computational Intelligence. Brazil: Statistics and Informatics Department Federal Rural University of Pernambuco.

Bagnasco, A., Fresi, F., Saviozzi, M., Silvestro, F., dkk., 2015. *Electrical Consumption Forecasting in Hospital Facilities: An Application Case*. Energy and Buildings 103 (2015) 261–270.

Bali, P. P., 2000. *Teori Dasar Listrik.* Jawa Bali: PT. PLN (Persero) UBS P3B.

Chai, T. & Draxler, R., 2014. *Root Mean Square Error (RMSE) or Mean Absolute Error (MAE)? – Arguments Against Avoiding RMSE in the Literatur*. Geosci, Model Dev., 7, 1247–1250, 2014.

Cichenski, M., Jarus, M., Miszkiewicz, M., dkk., 2015. *Supporting Supply Process in Charitable Organizations by Genetic Algorithm*. Computers & Industrial Engineering 88. Poznan: Institute of Computing Science, Poznan University of Technology, Piotrowo 2.

Escriva, G., Alvarez-bel, C., Roldan-blay, C., dkk., 2011. *New Artificial Neural Network Prediction Method for Electrical Consumption Forecasting Based on Building End-Uses*. Valencia: Institute for Energy Engineering, Universitat Politècnica de València, Camino de Vera.

Hsu, C., Chang, C. & Lin, C., 2010. *A Practical Guide to Support Vector Classification*. Taipei: Department of Computer Science National Taiwan University.

Jiang, L., Xiao, H., He J., dkk., 2015. *Application of Genetic Algorithm to Pyrolysis of Typical Polymers*. Fuel Processing Technology 138 (2015) 48–55.

Kaytez, F., Taplamacioglu, M., Cam, E. dkk., 2015. *Forecasting Electricity Consumption: A Comparison Of Regression Analysis, Neural Networks and Least Squares Support Vector Machines*. Electrical Power and Energy Systems 67 (2015) 431–438.

Kucharzyka, K., Crawford, R., Paszczynskia, A., dkk., 2012. *Maximizing microbial degradation of perchlorate using a genetic algorithm: Media optimization*. Journal of Biotechnology 157 (2012) 189– 197. Moscow: University of Idaho.

Kuo, B., Ho, H., Li, C., dkk., 2014. *A Kernel-Based Feature Selection Method for SVM With RBF Kernel for Hyperspectral Image Classification*. IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, Vol. 7, No. 1, January 2014.

Lin, J., Shen P. & Wen, H., 2015. *Repetitive Control Mechanism of Disturbance Cancellation Using a Hybrid Regression and Genetic Algorithm*. Mechanical Systems and Signal Processing 62-63 (2015) 356–365. Taichung : Feng Chia University.

Lin, J., Chang, C. & Huang, S., 2011. *Research on Software Effort Estimation Combined with Genetic Algorithm and Support Vector Regression*. 2011 International Symposium on Computer Science and Society. Taipe: Deptartment of Computer Science & Engineering, Tatung University.

Lu, X., Leng, Y., Xie, P., dkk., 2014. *An Intelligent Task Scheduling Algorithm of Electricity Consumption for Reducing the Load Peak*. Proceeding of the 11th World Congress on Intelligent Control and Automation Shenyang, China, June 29 - July 4 2014. Shanghai: School of Economics and Management, Shanghai University of Electric Power.

Mahmudy, W. F. 2013. Algoritma Evolusi. Malang: Program Teknologi Informasi dan Ilmu Komputer, Universitas Brawijaya.

Mas'udia, P. dan Wardoyo R., 2012. Optimasi Cluster Pada Fuzzy C-Means Menggunakan Algoritma Genetika Untuk Menentukan Nilai Akhir. IJCCS, Vol.6, No.1, January 2012, pp. 101-110. Yogyakarta: Jurusan Ilmu Komputer dan Elektronika, FMIPA UGM.

Malhotra, R., Singh N. dan Singh, Y., 2011. *Genetic Algorithms: Concepts, Design for Optimization of Process Controllers*. Computer and Information Science Vol. 4, No. 2; March 2011. Punjab: Punjab Technical University, Jalandhar.

Ogcu, G., Demirel, O. F. & Zaim, S., 2012. *Forecasting Electricity Consumption with Neural Networks and Support Vector Regression*. Procedia - Social and Behavioral Sciences 58 (2012) 1576 – 1585. Turkey: Department of Industrial Engineering, Fatih University.

Pasman, D. F., Muslim, M. A. & Dhofir M., 2010. Analisis Implementasi Jaringan Syaraf Adaptif untuk Peramalan Kebutuhan Energi Listrik Wilayah Malang. Jurnal Nitro Vol. 2, No. 2 April 2010. Malang: Teknik Elektro, Universitas Brawijaya

Patro, S.G.K., Sahu, K.K., 2015. *Normalization: A Preprocessing Stage*. Department of CSE& IT, VSSUT, Burla, Odisha, India.

Pramesti, D., Mahmudy, W. & Indriati. 2015. Optimasi Komposisi Pakan Kambing Potong Menggunakan Algoritma Genetika. DORO: Repository Jurnal Mahasiswa PTIIK Universitas Brawijaya, vol. 5, no. 13. Malang.

Priandani, N. D. & Mahmudy, W. F. 2015. Optimasi Travelling Salesman Problem with Time Windows (TSP-TW) pada Penjadwalan Paket Rute Wisata di Pulau Bali Menggunakan Algoritma Genetika. Seminar Nasional Sistem Informasi Indonesia. Malang: Jurusan Ilmu Komputer, Universitas Brawijaya.

Sbeity, I., Dbouk, M. & Kobeissi H., 2014. *Combining The Analytical Hierarchy Process and The Genetic Algorithm to Solve The Timetable Problem*. International Journal of Software Engineering & Applications (IJSEA), Vol.5, No.4, July 2014. Lebanon: Department of Computer Sciences, Faculty of Sciences, Section I, Lebanese University.

Scholkopf, B. & Smola, A., 2004. *A tutorial on support vector regression*. Statistics and Computing 14: 199–222. Netherlands: Kluwer Academic Publishers.

Shukla, A., Pandey, H. & Mehrotra, D., 2015. *Comparative Review of Selection Techniques in Genetic Algorithm*. Noida: Amity University.

Subanar & Permadi, I. 2010. Penerapan Algoritma Genetika untuk Optimasi Penjadwalan Tebangan Hutan (*Applying of Genetic Algorithm for Scheduling Optimation Cuts Away Forest*). JUITA Vol. I Nomor 1.

Suyanto, 2014. *Artificial Intelligence: Searching, Reasoning, Planning, Learning* (Revisi Kedua). Penerbit: Informatika, Bandung

Vijayakumar S., & Wu, S., 1999. *Sequential Support Vector Classiers and Regression*. Proc. International Conference on Soft Computing (SOCO'99), Genoa, Italy, pp.610-619. Saitama: RIKEN Brain Science Institute, The Institute for Physical and Chemical Research, Hirosawa 2-1, Wako-shi

Wang, H. & Yang, S., 2013. *Electricity Consumption Prediction Based on SVR with Ant Colony Optimization*. TELKOMNIKA, Vol.11, No.11, November 2013, pp. 6928-6934. Anhui Hefei: School of Management of Hefei University of Technology.

Widodo, S., 2013. Dasar dan pengukuran Listrik (Semester 1). [e-book] Kementriaan Pendidikan dan Kebudayaan Republik Indinesia. Tersedia di: <http:// bse.kemdikbud.go.id> [Diakses 15 September 2015]

Wu, L. & Wang, Y., 2009. *Modelling DGM(1,1) Under The Criterion of The Minimization of Mean Absolute Percentage Error*. 2009 Second International Symposium on Knowledge Acquisition and Modeling. Wenzhou University, Wenzhou, China.

Yi-feng, J. & Shu-wen, W., 2010. *Village Electrical Load Prediction by Genetic Algorithm and SVR*. Wuhan: School of Electrical Engineering Wuhan University.

LAMPIRAN A DATA KONSUMSI KEBUTUHAN LISTRIK (KWH) KOTA MALANG PERIODE 2010-2014

A.1 Data Konsumsi Kebutuhan Listrik (kWh) Kota Malang Periode 2010-2014

Konsumsi Listrik Rayon Malang Kota dalam satuan kWh												
TAHUN	**JANUARI**	**FEBRUARI**	**MARET**	**APRIL**	**MEI**	**JUNI**	**JULI**	**AGUSTUS**	**SEPTEMBER**	**OKTOBER**	**NOVEMBER**	**DESEMBER**
2010	14654406	9753559	13558244	13152956	14489468	14457348	14576758	14104757	14384564	14379629	14465189	14690521
2011	14353838	14782131	14706259	14048550	14682443	15056219	14429970	14193229	13965342	14579500	14868337	13665752
2012	14976393	15505607	15368241	15068711	15565365	15888400	15705979	14980882	14497723	16101235	16426655	16663745
2013	16416151	15722762	15937932	16467328	16824804	16668235	16047387	15692999	16286564	17029736	16851238	16745668
2014	16480785	16658338	16654060	16844239	17680031	17169211	16584303	16614758	16743808	17447669	17883099	17624279

Data yang ada pada tabel diatas merupakan data konsumsi kebutuhan listrik (kWh) Kota Malang periode 2010-2014 dengan data bulanan.

LAMPIRAN B VISUALISASI HASIL PENGUJIAN

B.1 Nilai Parameter Optimal untuk Peramalan Konsumsi Listrik Kota Malang Menggunakan Metode GA-SVR

Hasil berikut merupakan nilai yang optimal untuk masing-masing variabel yang ada pada metode SVR dan metode Algoritma Genetika setelah beberapa uji coba yang dilakukan:

1. Batas pencarian parameter λ : 0.01 - 10
2. Batas pencarian parameter ε : 0.00000001 – 0.0001
3. Batas pencarian parameter *cLR* : 0.00001 – 0.1
4. Batas pencarian parameter *C* : 0.0001 – 0.1
5. Jumlah iterasi SVR : 100000
6. Jumlah generasi GA : 1750
7. Jumlah popSize : 40
8. *Crossover rate* : 0.1
9. *Muation rate* : 0.9
10. Jumlah periode prakiraan : 12
11. Nilai λ : 4.99107947841775
12. Nilai ε : 0.0001
13. Nilai *cLR* : 0.1
14. Nilai *C* : 0.027784589653607684

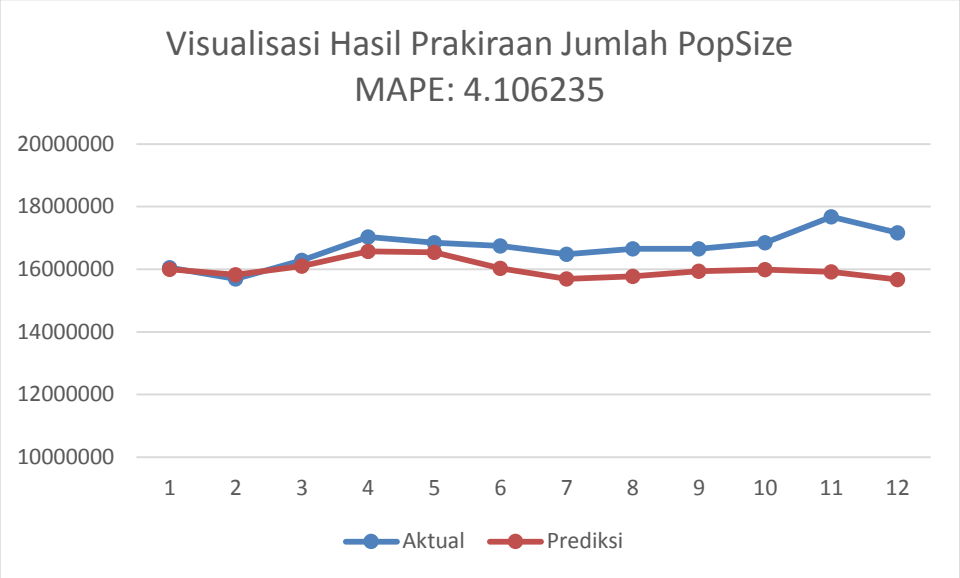
Hasil prakiraan menggunakan parameter diatas menghasilkan nilai MAPE sebesar 3.922479778.

B.2 Nilai α* dan α Optimal untuk Fungsi Regresi pada SVR

Hasil berikut merupakan nilai α* dan α yang optimal untuk permalan SVR yang didapatkan pada iterasi terakhir yaitu 1000000:

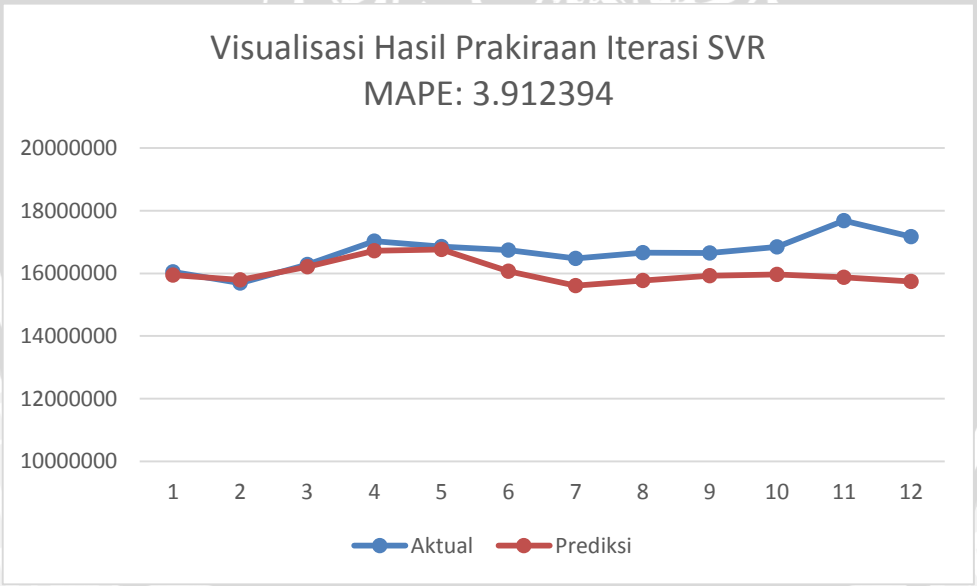
NO	α*	α	NO	α*	α	NO	α*	α
1	4.85E-04	0.086773	13	0	0.1	25	0.000133	0.062614
2	0	0.1	14	0	0.1	26	0	0.1
3	0	0.1	15	0	0.1	27	0	0.1
4	0	0.1	16	0	0.1	28	0.096516	0.001115
5	0.1	0	17	0.058275	0.004429	29	0.099672	0.000227
6	0.1	0	18	0	0.1	30	0.1	0
7	0	0.076175	19	0.00851	0.004575	31	0.089174	0.000272
8	0.1	0	20	0.011004	0.024677	32	0	0.1
9	0.059273	0	21	0.083486	0	33	0.025789	0.000176
10	0	0.1	22	0.001465	0.016083	34	0.066945	0.001109
11	0.1	0	23	0.1	0	35	0.1	0
12	0.1	0	24	0.012609	0.011054	36	0.1	0

B.3 Visualisasi Hasil Uji Coba Popsize



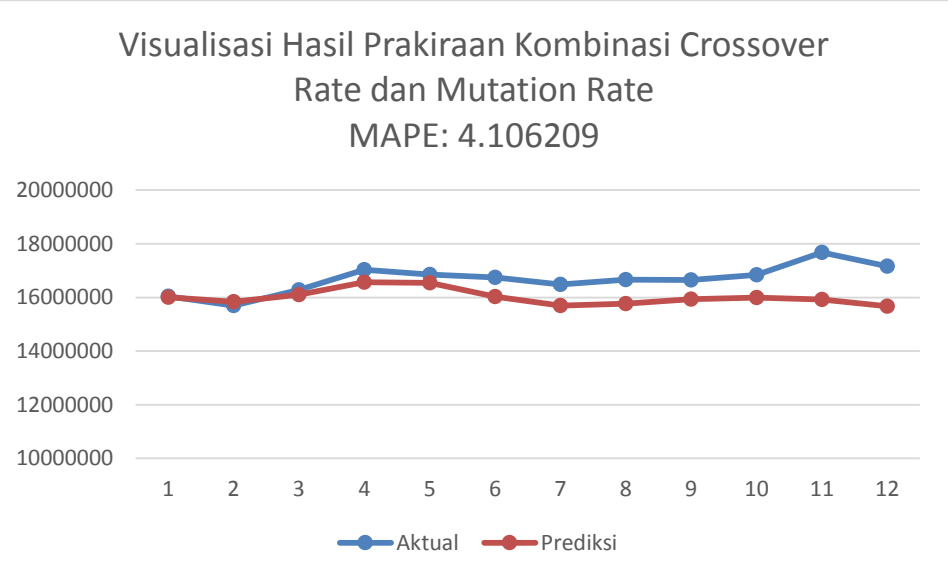
Grafik diatas merupakan visualisasi hasil prakiraan jumlah *popSize* yang terbaik dari 10 variasi dan 10 kali percobaan dengan MAPE sebesar 4.106235, yaitu menggunakan *popSize* sebesar 40.

B.4 Visualisasi Hasil Uji Jumlah Iterasi SVR



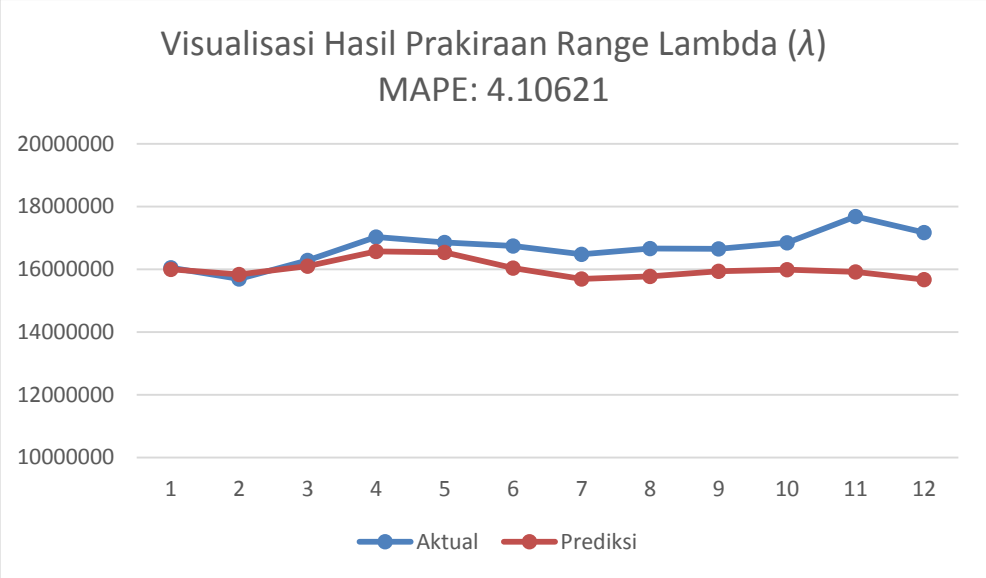
Grafik diatas merupakan visualisasi hasil prakiraan jumlah *popSize* yang terbaik dari 10 variasi dan 10 kali percobaan dengan MAPE sebesar 3.912394, yaitu menggunakan iterasi SVR sebanyak 100000 iterasi.

B.5 Visualisasi Hasil Uji Coba Kombinasi *Crossover Rate* dan *Mutation Rate*



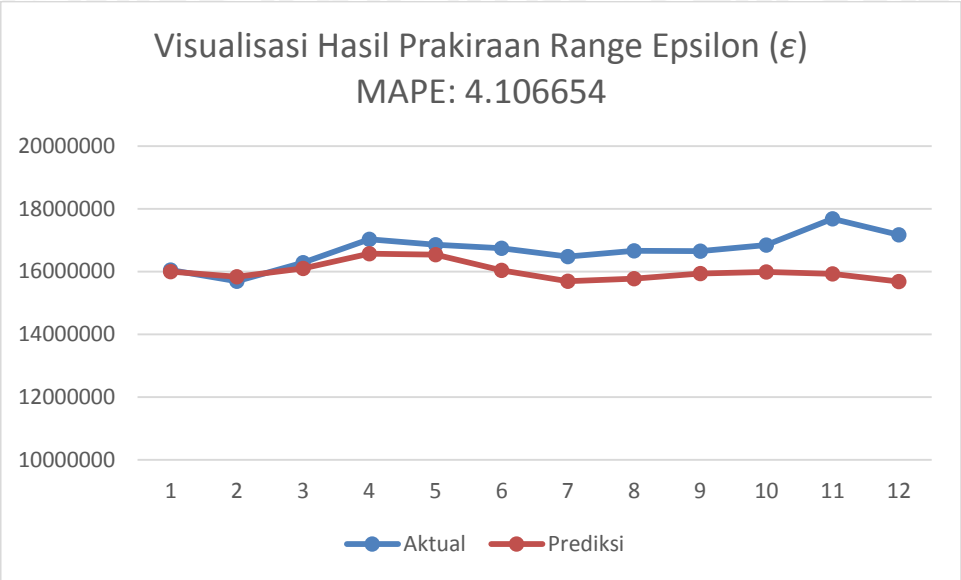
Grafik diatas merupakan visualisasi hasil prakiraan kombinasi *crossover rate* dan *mutation rate* yang terbaik dari 11 variasi dan 10 kali percobaan dengan MAPE sebesar 4.106209, yaitu menggunakan *crossover rate* sebesar 0.1 dan *mutation rate* sebesar 0.9.

B.6 Visualisasi Hasil Uji Coba Range *Lambda* (*λ*)



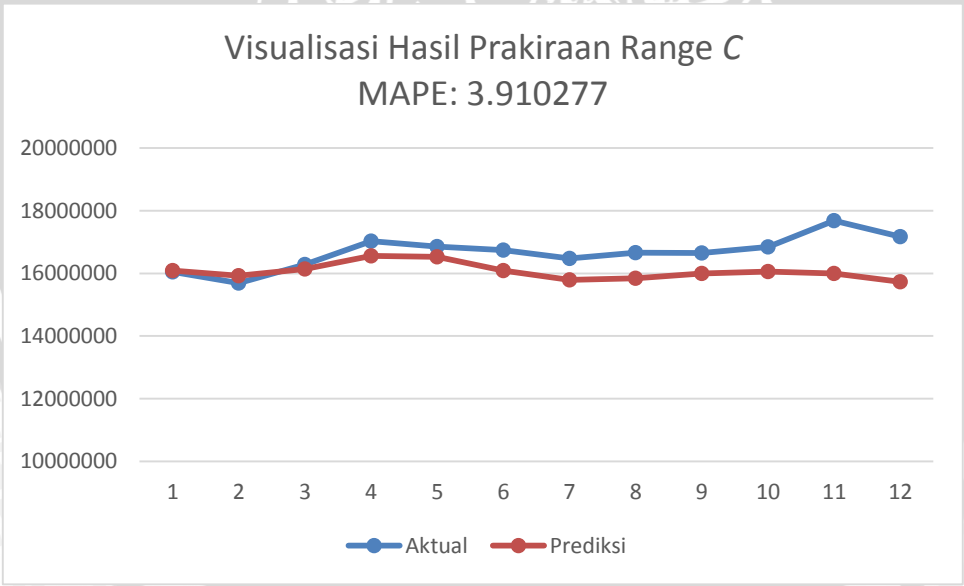
Grafik diatas merupakan visualisasi hasil prakiraan *range λ* yang terbaik dari 9 variasi dan 10 kali percobaan dengan MAPE sebesar 4.10621, yaitu menggunakan batas atas *λ* sebesar 0.01 dan batas atas sebesar 10.

B.7 Visualisasi Hasil Uji Coba Range *Epsilon* (*ε*)



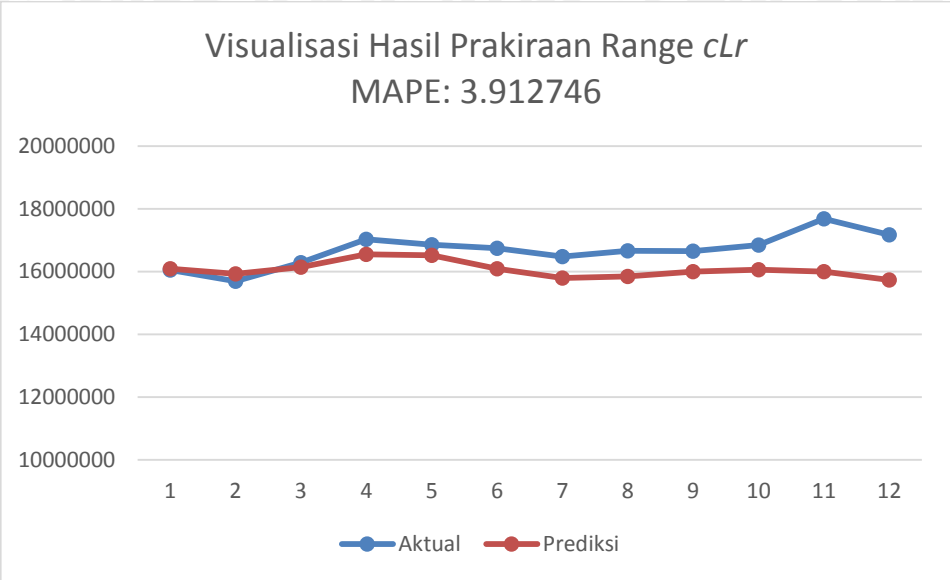
Grafik diatas merupakan visualisasi hasil prakiraan *range ε* yang terbaik dari 9 variasi dan 10 kali percobaan dengan MAPE sebesar 4.106654, yaitu menggunakan batas atas *ε* sebesar 0.00000001 dan batas atas sebesar 0.0001.

B.8 Visualisasi Hasil Uji Coba Range *C*



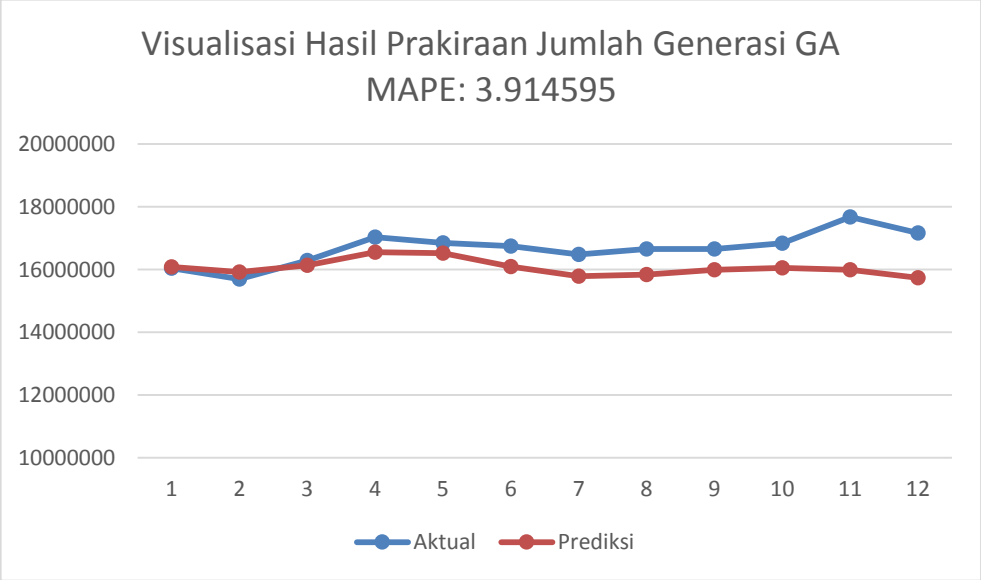
Grafik diatas merupakan visualisasi hasil prakiraan *range C* yang terbaik dari 9 variasi dan 10 kali percobaan dengan MAPE sebesar 3.910277, yaitu menggunakan batas atas *ε* sebesar 0.0001 dan batas atas sebesar 0.1.

B.9 Visualisasi Hasil Uji Coba Range *cLr*



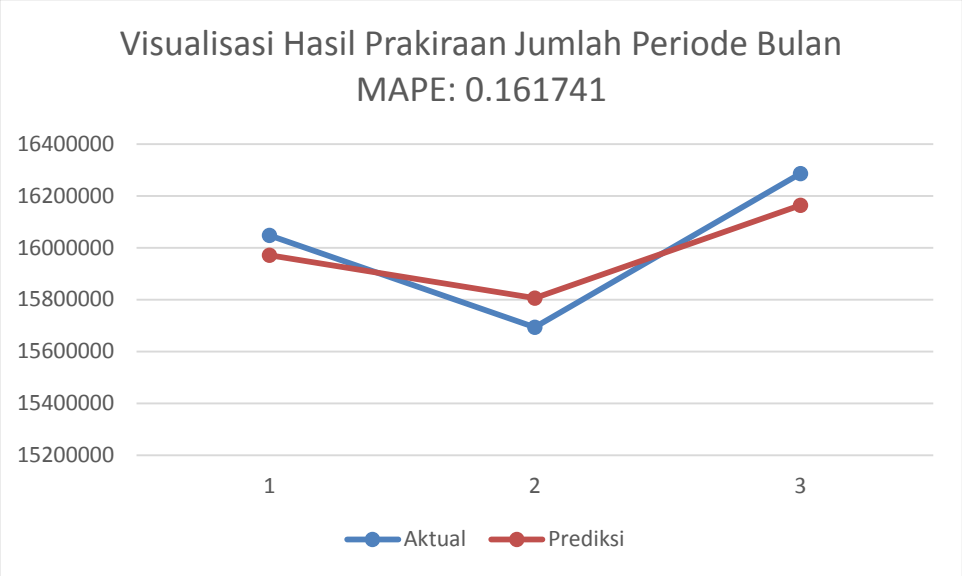
Grafik diatas merupakan visualisasi hasil prakiraan *range cLr* yang terbaik dari 9 variasi dan 10 kali percobaan dengan MAPE sebesar 3.912746, yaitu menggunakan batas atas ϵ sebesar 0.00001 dan batas atas sebesar 0.1.

B.10 Visualisasi Hasil Uji Coba Jumlah Generasi GA



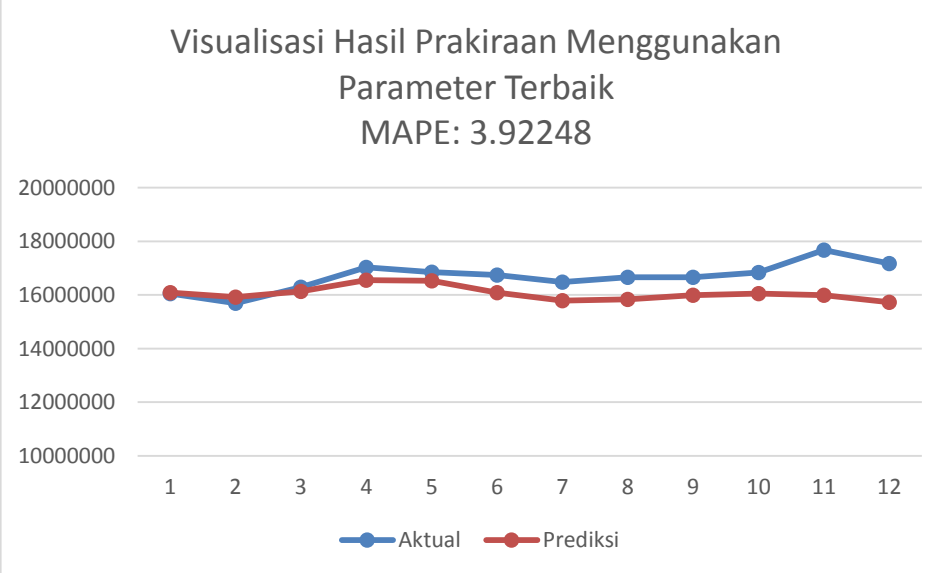
Grafik diatas merupakan visualisasi hasil prakiraan jumlah generasi GA yang terbaik dari 10 variasi dan 10 kali percobaan dengan MAPE sebesar 3.914595, yaitu menggunakan generasi sebesar 1750.

B.11 Visualisasi Hasil Uji Coba Jumlah Periode Prakiraan



Grafik diatas merupakan visualisasi hasil prakiraan jumlah periode bulan yang terbaik dari 10 variasi dan 10 kali percobaan dengan MAPE sebesar 0.161741, yaitu menggunakan periode sebanyak 3 bulan.

B.12 Visualisasi Hasil Uji Coba Parameter Terbaik



Grafik diatas merupakan visualisasi hasil prakiraan menggunakan parameter terbaik yang didapatkan pada pengujian sebelumnya dengan MAPE sebesar 3.92248, yaitu menggunakan periode sebanyak 12 bulan.

LAMPIRAN C WAWANCARA

Berikut dibawah ini adalah hasil wawancara dengan Mbak Werdi Pratiwi selaku pegawai PLN Kota Malang bagian *Junior Functional* Pengelolaan Rekening mengenai sistem yang digunakan PLN Kota Malang saat ini untuk memprakiraan kebutuhan konsumsi energi listrik kedepannya.

Saya	: Di PLN Kota Malang apakah sudah ada sistem untuk memprakirakan kebutuhan energi listrik untuk waktu kedepan?
Pegawai PLN	: Kalau prediksi disini tidak ada sistemnya. Disini prediksi hanya berdasarkan data-data sebelumnya.
Saya	: Apakah sistemnya menggunakan algoritma tertentu?
Pegawai PLN	: Tidak, hanya pakai perhitungan sederhana saja
Saya	: Jadi, perhitungannya memakai hari atau bulan sebelumnya?
Pegawai PLN	: Jadi untuk kWh itu kita ada aplikasi dari P3B. Misal, kita buka aplikasi nya tanggal 20, kita mau memprediksi untuk sisa bulan (10 har kedepan). Nah untuk tanggal 1-20 kan kita sudah punya datanya, nah dari situ kita prediksi 10 hari kedepan. Prediksi nya itu kita rata-rata saja pemakaiannya.
Saya	: Oh, jadi sistemnya hanya menggunakan rumus rata-rata saja?
Pegawai PLN	: Iya
Saya	: Berarti datanya harian ya bukan bulanan?
Pegawai PLN	: Bukan
Saya	: Oh begitu, baik mbak terimakasih
Pegawai PLN	: Iya sama-sama