IMPLEMENTASI METODE *SUPPORT VECTOR MACHINE* (SVM) BERBASIS *ANT COLONY OPTIMIZATION* (ACO) UNTUK KLASIFIKASI TINGKAT RESIKO PENYAKIT STROKE

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:
Herdian Anggara Winata
NIM: 115060800111063

PROGRAM STUDI INFORMATIKA / ILMU KOMPUTER
PROGRAM TEKNOLOGI INFORMASI DAN ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2016

PENGESAHAN

IMPLEMENTASI METODE *SUPPORT VECTOR MACHINE* (SVM) BERBASIS *ANT COLONY OPTIMIZATION* (ACO) UNTUK KLASIFIKASI TINGKAT RESIKO PENYAKIT STROKE

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh :
Herdian Anggara Winata
NIM:115060800111063

Skripsi ini telah diuji dan dinyatakan lulus pada
20 Januari 2016
Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I	Dosen Pembimbing II
Imam Cholissodin, S.Si., M.Kom	Edy Santoso, S.Si., M.Kom
NIK: 850719 16 11 0422	NIP: 19740414 200312 1 004

Mengetahui
Ketua Program Studi Informatika / Ilmu Komputer

Drs. Marji, M.T.
NIP: 19670801 199203 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, 26 Januari 2016

Herdian Anggara Winata

NIM: 115060800111063

KATA PENGANTAR

Puji dan syukur Penulis panjatkan kehadirat Allah SWT, karena hanya dengan rahmat dan karunia-Nya Penulis dapat menyelesaikan skripsi dengan judul **"Implementasi Metode *Support Vector Machine* (SVM) Berbasis *Ant Colony Optimization* (ACO) untuk Klasifikasi Tingkat Resiko Penyakit"** dengan baik. Melalui kesempatan ini, Penulis ingin menyampaikan rasa hormat dan terima kasih yang sebesar-besarnya kepada semua pihak yang telah memberikan bantuan dan dukungan selama pengerjaan skripsi, diantaranya:

1. Imam Cholissodin, S.Si., M.Kom. selaku dosen pembimbing I yang telah banyak memberikan ilmu, bimbingan, arahan, motivasi, serta meluangkan waktunya selama penyusunan skripsi ini.
2. Edy Santoso, S.Si., M.Kom. selaku dosen pembimbing II yang telah banyak memberikan ilmu, bimbingan, arahan, nasihat, serta meluangkan waktunya selama penyusunan skripsi ini.
3. Drs. Marji, M.T. dan Issa Arwani, S.Kom., M.Sc., selaku Ketua dan Sekretaris Program Studi Informatika/Ilmu Komputer Universitas Brawijaya.
4. Ir. Sutrisno, M.T., Ir. Heru Nurwarsito, M.Kom., Himawat Aryadita, S.T., M.Sc., dan Edy Santoso, S.Si., M.Kom., selaku Ketua, Wakil Ketua I, Wakil Ketua II, dan Wakil Ketua III Program Teknologi Informasi dan Ilmu Komputer Universitas Brawijaya.
5. Kedua orang tua Srinoto dan Nanik Widayati yang telah memberi motivasi, kasih sayang serta dukungan moril dan materil. Kakak saya Ika Wahyu Anita yang telah memberikan semangat dan dukungan dari awal sampai akhir pengerjaan skripsi ini.
6. Segenap bapak dan ibu dosen yang telah mendidik dan mengajarkan ilmunya kepada Penulis selama menempuh pendidikan di Program Teknologi Informasi dan Ilmu Komputer Universitas Brawijaya.
7. Staf administrasi Program Studi Informatika/Ilmu Komputer, Program Teknologi Informasi dan Ilmu Komputer.
8. Sahabat saya Fajar Farisuddin, yang telah banyak membantu dalam pengerjaan skripsi ini.
9. Sahabat saya Aulia Reza Farhana, yang telah banyak membantu, memberikan semangat dan dukungan dalam pengerjaan skripsi ini.
10. Keluarga Besar Mahasiswa Informatika/Ilmu Komputer khususnya angkatan 2011, seluruh teman-teman Kelas H terima kasih atas segala bantuan dan dukungannya selama ini.

11. Semua pihak yang tidak dapat penulis sebutkan satu per satu yang terlibat baik secara langsung maupun tidak langsung demi terselesaikannya skripsi ini.

Semoga jasa dan amal baik mendapatkan balasan dari Allah SWT. Dengan segala kerendahan hati, penulis menyadari sepenuhnya bahwa skripsi ini masih jauh dari sempurna karena keterbatasan materi dan pengetahuan yang dimiliki penulis. Akhirnya, semoga skripsi ini dapat bermanfaat dan berguna bagi pembaca terutama mahasiswa FILKOM Universitas Brawijaya.

Malang, 26 Januari 2016

Herdian Anggara Winata

ABSTRAK

Herdian Anggara Winata. 2016. Implementasi Metode *Support Vector Machine* (SVM) berbasis *Ant Colony Optimization* (ACO) untuk Klasifikasi Tingkat Resiko Penyakit Stroke. Program Studi Informatika, Fakultas Ilmu Komputer, Universitas Brawijaya, Malang. Dosen Pembimbing: Imam Cholissodin, S.Si, M.Kom. dan Edy Santoso, S.Si., M.Kom.

Stroke didefinisikan sebagai gangguan suplai darah ke otak yang dapat disebabkan karena adanya penyumbatan atau pendarahan pada pembuluh darah. Karena gangguan tersebut, daerah otak yang terkena stroke tidak dapat berfungsi dengan normal yang mengakibatkan sebagian tubuh penderita dapat mengalami kelumpuhan atau bahkan bisa mengakibatkan kematian. Penyakit stroke sudah dapat dideteksi, akan tetapi untuk proses deteksinya sendiri dibutuhkan waktu yang relatif lama dan biaya yang cukup mahal. Oleh karena itu, dibutuhkan suatu sistem yang dapat mempercepat dan memudahkan pendeteksian awal penyakit stroke. Salah satu teknik yang dapat digunakan yaitu melakukan klasifikasi data dengan metode *Support Vector Machine* (SVM). Akan tetapi dalam metode SVM masih belum ada perhitungan optimum dalam melakukan pemilihan parameter. Untuk mengatasi masalah tersebut digunakan metode *Ant Colony Optimization* untuk melakukan pencarian terhadap nilai parameter *C* dan σ pada SVM sehingga nantinya didapatkan nilai parameter yang optimal. Dari hasil pengujian didapatkan akurasi terbaik sebesar 0,87867 yaitu ketika batas nilai parameter *C* 0,0001 ~ 9,9999, batas nilai parameter σ 10,000 ~ 99,999, nilai parameter λ = 0,2, nilai parameter γ = 0,1, iterasi ACO = 10, jumlah semut = 100, dan panjang *fold* = 3.

Kata kunci: Stroke, Klasifikasi, *Support Vector Machine*, *Ant Colony Optimization*, *K-fold Cross Validation*.

ABSTRACT

Herdian Anggara Winata. 2016. *Implementation of Support Vector Machine (SVM) Method Based on Ant Colony Optimization (ACO) for Stroke Risk Level Classification.* *Informatics Studies Program*, *Faculty of Computer Science*, *University of Brawijaya*, Malang. *Supervisors*: Imam Cholissodin, S.Si, M.Kom. *And* Edy Santoso, S.Si., M.Kom.

Stroke was defined as the interruption of blood supply to the brain that can be caused due to blockage or bleeding in blood vessels. Because of the disorder, the area of the brain affected by stroke can not function normally, which resulted in part of the body the patient may experience a seizure or could even lead to death. Stroke can be detected, but for its own detection process takes a relatively long time and the cost is quite expensive. Therefore, needed a system that can accelerate and facilitate the early detection of stroke. One technique that can be used is to perform classification using Support Vector Machine (SVM). However, in SVM there's no calculation in the selection of the optimum parameters. To overcome these problems Ant Colony Optimization method is used to perform a search on the value of parameters C and σ on SVM to obtain optimal parameter values. From the test result, obtained the best accuracy 0.87867 when the limit value of the parameter C 0.0001 ~ 9.9999, the limit value of parameter σ 10.000 ~ 99.999, value of parameter λ = 0.2, value of parameter γ = 0.1, iteration ACO = 10, the number of ants = 100, and fold length = 3.

Keywords: *Stroke, Classification, Support Vector Machine, Ant Colony Optimization, K-fold Cross Validation.*

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR LAMPIRAN

BAB 1 PENDAHULUAN

Dalam bab ini akan dibahas mengenai beberapa alasan dan rencana selama melakukan penelitian yang akan dilakukan agar dapat mendapatkan hasil yang optimal.

1.1 Latar Belakang

Cerebrovascular Accident (CVA) atau lebih dikenal dengan stroke merupakan penyakit yang dialami oleh otak. Stroke terjadi ketika suplai darah ke otak mengalami gangguan seperti adanya penyumbatan atau pendarahan, yang pada akhirnya dapat mengakibatkan fungsi-fungsi otak akan menghilang secara drastis (Muyderman, 2009). Pada tahun 2010, prevalensi stroke di seluruh dunia sebesar 33 juta. Stroke juga merupakan penyebab kedua kematian global setelah penyakit jantung, dengan angka 11,13% dari total kematian di seluruh dunia (Mozaffarian, 2015). Di Indonesia sendiri stroke merupakan masalah yang butuh penanganan serius. Kementerian Kesehatan Republik Indonesia tahun 2013 menyebutkan, prevalensi stroke di Indonesia mencapai 12,1 per 1000 orang. Pada masyarakat dengan pendidikan rendah prevalensi stroke mencapai 32,8%. Masyarakat kota tercatat memiliki prevalensi stroke lebih tinggi daripada masyarakat desa, yaitu sebesar 12,7% (Kemenkes RI, 2013).

Pendeteksian penyakit stroke sudah dapat dilakukan dengan dua cara, yaitu dengan *Computed Tomography* (CT) *Scan* dan *Magnetic Resonance Imaging* (MRI). Namun biaya pemeriksaan dengan CT *Scan* dan MRI ini cukup mahal, berkisar antara 1 - 4 juta rupiah. Selain itu pemeriksaan dengan CT *Scan* dan MRI ini kebanyakan dilakukan ketika seseorang sudah terkena stroke. Sehingga hal ini dapat menyebabkan keterlambatan dalam penanganan medis yang dapat menyebabkan kondisi pasien semakin buruk. Penyakit stroke sebenarnya juga dapat di deteksi sejak dini, yaitu dengan pengamatan sederhana yang dikenal dengan FAST (*Face*, *Arms Drive*, *Speech*, dan *Three of signs*), namun cara ini sangat kurang dalam implementasinya karena ketidaktahuan masyarakat awam terhadap gejala-gejala yang ada. Oleh karena itu dibutuhkan sistem cerdas yang bisa memudahkan petugas medis ataupun masyarakat awam dalam melakukan identifikasi dini tingkat resiko penyakit stroke.

Pada tahun 1979 Vapnik menciptakan suatu metode yang dikenal dengan nama *Support Vector Machine* (SVM). Pada tahun 1995 metode ini digunakan untuk melakukan klasifikasi dan memecahkan permasalahan regresi (Cortes, 1995). Penelitian sebelumnya tentang SVM yang dilakukan oleh Sahar A. Mokhtar dan Alaa. M. Elsayad pada tahun 2011 dengan membandingkan metode SVM dengan metode *Decision Tree* (DT) dan *Artificial Neural Network* (ANN) untuk memprediksi tingkat keparahan penyakit kanker payudara dengan hasil nilai akurasi masing-masing metode adalah 81,25%, 78,12%, dan 80,56%. Dalam penelitian tersebut dapat dilihat bahwa metode SVM memiliki akurasi paling baik dalam melakukan prediksi dibandingkan dua metode lainnya. Namun demikian, LIU Chun-bo dkk (2008) dalam penelitiannya menyatakan bahwa kinerja dari

SVM model bergantung pada pengaturan nilai parameter yang tepat. Pengaturan parameter SVM yang sesuai dianggap bisa meningkatkan akurasi dan kinerja dari SVM model. Namun untuk SVM sendiri tidak ada perhitungan khusus untuk melakukan pengaturan nilai parameter. Maka dari itu percobaan baru dilakukan dalam melakukan pemilihan parameter. Zhang dkk (2014) kemudian mengusulkan penggunaan *Ant Colony Optimization* (ACO) untuk mencari nilai parameter *C* dan *σ* pada SVM yang nantinya dapat mengurangi *randomness* dan *running time* yang diakibatkan dari pemilihan parameter secara manual (Zhang, 2014). Ant *Colony Optimization* (ACO) sendiri merupakan metode yang diadopsi dari perilaku koloni semut yang dikenal sebagai *Ant System* (Dorigo, 1996). ACO merupakan suatu metode untuk melakukan optimisasi yang diperkenalkan pada awal 90-an oleh Dorigo dan Caro. Penelitian sebelumnya menyebutkan, implementasi metode ACO untuk pemilihan fitur pada kategorisasi dokumen teks dapat menghasilkan akurasi sampai 96% (Yudis, 2009).

Berdasarkan kasus dan beberapa penelitian sebelumnya di atas, maka pada penelitian ini digunakan algoritma *Support Vector Machine* (SVM) yang dioptimalkan dengan metode *Ant Colony Optimization* (ACO). Sehingga judul yang diambil pada skripsi ini adalah "Implementasi metode *Support Vector Machine* (SVM) berbasis *Ant Colony Optimization* (ACO) untuk klasifikasi tingkat resiko penyakit stroke". Sistem ini nantinya diharapkan akan dapat melakukan klasifikasi tingkat resiko stroke secara efektif dan efisien sehingga dapat menekan waktu dan biaya yang dibutuhkan untuk melakukan pendeteksian tingkat resiko penyakit stroke. Sebelum melakukan klasifikasi dengan menggunakan metode SVM, pertama-tama sistem ini akan terlebih dahulu melakukan pencarian nilai parameter SVM dengan menggunakan metode *Ant Colony Optimization*. Setelah didapatkan nilai parameter SVM, setelah itu dilakukan klasifikasi tingkat resiko penyakit stroke yang nantinya didapatkan nilai akurasi dari proses klasifikasi tersebut untuk menentukan mana nilai parameter SVM yang optimal.

1.2 Rumusan Masalah

Berdasarkan uraian latar belakang diatas, maka dapat dirumuskan masalah sebagai berikut:

1. Bagaimana mengimplementasi Metode *Support Vector Machine* berbasis *Ant Colony Optimization* untuk klasifikasi tingkat resiko penyakit stroke.
2. Bagaimana akurasi dari hasil Implementasi Metode *Support Vector Machine* berbasis *Ant Colony Optimization* untuk klasifikasi tingkat resiko penyakit stroke.

1.3 Tujuan

Tujuan penelitian didasarkan pada rumusan masalah yang telah dijabarkan adalah sebagai berikut:

1. Mengimplementasi Metode *Support Vector Machine* berbasis *Ant Colony Optimization* untuk klasifikasi tingkat resiko penyakit stroke.
2. Mendapatkan akurasi dari hasil Implementasi Metode *Support Vector Machine* berbasis *Ant Colony Optimization* untuk klasifikasi tingkat resiko penyakit stroke.

1.4 Manfaat

Penelitian ini memiliki manfaat sebagai berikut:

1. Membantu melakukan pendeteksian dini resiko penyakit stroke.
2. Meminimalkan waktu dalam pendeteksian resiko penyakit stroke.
3. Meminimalkan biaya yang dibutuhkan untuk pendeteksian resiko penyakit stroke.
4. Membantu ahli medis dalam melakukan identifikasi awal untuk penanganan kasus stroke.

1.5 Batasan Masalah

Untuk menghindari penyimpangan maka penulis membatasi permasalahan yang akan dibahas sebagai berikut :

1. Metode utama yang digunakan adalah *Support Vector Machine* dan *Ant Colony Optimization*.
2. Kernel SVM yang digunakan adalah *kernel Gaussian* RBF
3. Parameter SVM yang akan dioptimasi adalah parameter *C* dan σ.
4. Data yang digunakan adalah data rekam medik hasil Laboratorium klinik sejahtera sebanyak 200 buah dengan 5 fitur yaitu umur, kolesterol total, HDL, LDL, dan Trigliserida.

1.6 Sistematika Pembahasan

Sistematika penulisan digunakan untuk memberikan gambaran dan uraian dari penulisan laporan penelitian secara garis besar yang meliputi beberapa bab sebagai berikut:

BAB 1 PENDAHULUAN

Menguraikan mengenai latar belakang tugas akhir, rumusan masalah, tujuan dan manfaat implementasi metode *Support Vector Machine* (SVM) berbasis *Ant Colony Optimization* (ACO) untuk klasifikasi tingkat resiko penyakit stroke.

BAB 2 LANDASAN KEPUSTAKAAN

Dalam bab ini menjelaskan kajian pustaka dan teori-teori yang berkaitan dengan pembuatan skripsi ini. Teori yang dijelaskan di sini meliputi penjelasan tentang stroke, klasifikasi, *Support Vector Machine*, *Ant Colony Optimization*, dan penjelasan teori lain yang berhubungan dengan proses pembuatan skripsi ini.

BAB 3 METODOLOGI PENELITIAN

Pada bab ini membahas mengenai langkah-langkah kerja yang dilakukan dalam penelitian ini dan formulasi algoritma yang digunakan.

BAB 4 PERANCANGAN

Membahas mengenai analisis kebutuhan dan perancangan sistem klasifikasi tingkat resiko penyakit stroke.

BAB 5 IMPLEMENTASI

Menguraikan proses implementasi metode *Support Vector Machine* (SVM) berbasis *Ant Colony Optimization* (ACO) untuk klasifikasi tingkat resiko penyakit stroke.

BAB 6 PENGUJIAN DAN ANALISIS

Membahas tentang hasil pengujian dan analisa pengujian implementasi metode *Support Vector Machine* (SVM) berbasis *Ant Colony Optimization* (ACO) untuk klasifikasi tingkat resiko penyakit stroke.

BAB 7 PENUTUP

Menguraikan kesimpulan dan saran dari seluruh rangkaian penelitian implementasi metode *Support Vector Machine* (SVM) berbasis *Ant Colony Optimization* (ACO) untuk klasifikasi tingkat resiko penyakit stroke.

BAB 2 LANDASAN KEPUSTAKAAN

Bab ini berisi tentang landasan kepustakaan yang meliputi kajian pustaka dan dasar teori yang digunakan untuk menunjang penulisan skripsi mengenai Implementasi Metode *Support Vector Machine* berbasis *Ant Colony Optimization* untuk klasifikasi tingkat resiko penyakit stroke. Beberapa dasar teori yang dimaksud adalah Stroke, Klasifikasi, *Support Vector Machine* (SVM), dan *Ant Colony Optimizaton* (ACO).

2.1 Kajian Pustaka

Berikut kajian pustaka yang menjelaskan beberapa penelitian yang telah dilakukan terkait dengan metode *Support Vector Machine* dan *Ant Colony Optimizatio*n yang ditunjukkan melalui Tabel 2.1. Salah satunya yaitu penelitian yang dilakukan oleh Afrizal dalam implementasi algoritma *Support Vector Machine*.

Afrizal dalam penelitiannya menggunakan algoritma *Support Vector Machine* yang dioptimalkan dengan metode *Simplified Sequential Minimal Optimization* (*Simplified* SMO). *Simplified Sequential Minimal Optimization* (*Simplified* SMO) adalah algoritma sederhana dalam proses pelatihan SVM untuk menyelesaikan permasalahan *Quadratic Problem* (QP) yaitu dengan memilih dua *Langrange Multipliers* untuk dioptimasi bersamaan. Data yang digunakan yaitu sebanyak 200 data yaitu data penderita penyakit stroke dengan parameter-parameter yaitu Umur, *Cholesterol Total*, HDL *Cholesterol*, LDL *Cholesterol*, dan Trigliserida. Dari hasil pengujian didapatkan akurasi terbaik ketika dilakukan pengujian *cross validation* dimana rata-rata akurasi adalah 0,8939 ketika panjang *fold* 5 dan nilai parameter *epsilon* 0,001 dan *cost* 10. Standar deviasi terbaik didapatkan sebesar 0,1005 dengan pengujian *Cross Validation* pada saat panjang *fold* 10 dengan nilai parameter *epsilon* 0,001 dan *cost* 10 (Afrizal, 2015).

Penelitian kedua dilakukan oleh Yudis menggunakan metode *Ant Colony Optimization* untuk melakukan pemilihan fitur dalam kategorisasi dokumen teks. Dalam sistem pemilihan fitur yang dibangun terdapat dua subsistem yang saling berhubungan yaitu subsistem ACO dan subsistem *NN- Classifier*. Untuk subsistem ACO digunakan untuk menghasilkan *feature subset* sedangkan subsistem *NN-Classifier* digunakan untuk mengevaluasi *feature subset* yang dihasilkan oleh subsistem ACO. Pada hasil uji coba dan analisa hasil pengujian, didapatkan bahwa implementasi metode ACO bisa mengurangi *feature space* dalam *feature selection* sampai menjadi kurang dari 10% dari dimensi awal. Dari *feature subset* yang dihasilkan oleh *feature selection* dapat menghasilkan akurasi sampai 96% (Yudis, 2009).

Penelitian ketiga dilakukan oleh Chao Zhang, H-C Mei, dan Hao Yang. Pada penelitian ini bertujuan untuk melakukan optimasi parameter *C* dan *σ* pada *Support Vector Machine* dengan menggunakan algoritma *Ant Colony Optimization*. Dalam implementasinya pemilihan parameter *C* dan *σ* pada

algoritma ini direpresentasikan dalam sistem koordinat dua dimensi. Koordinat *X* didefinisikan sebagai banyaknya digit yang mungkin dari parameter *C* dan *σ*, koordinat *Y* merupakan variasi angka dari 0 – 10. Diberikan batasan untuk banyak nya digit yang mungkin dari parameter *C* dan *σ* yaitu masing-masing sebanyak 5 digit. Langkah pertama yaitu melakukan inisialisai jumlah semut dan jumlah iterasi. Kemudian menghitung nilai probabilitas yang selanjutnya akan digunakan untuk menentukan titik yang akan dipilih oleh semut. Setelah didapat lintasan dari semut kemudian dilakukan kalkulasi terhadap lintasan tersebut. Dari kalkulasi akan didapatkan nilai yang akan digunakan sebagai parameter SVM. Nilai parameter yang telah dikalkulasi selanjutnya akan dihitung nilai akurasi rata-rata dengan *K-fold cross validation* dengan pelatihan SVM. Nilai parameter dari semut yang memiliki akurasi rata-rata terbaik akan digunakan untuk *melakukan update pheromone*. Langkah tersebut akan diulang sampai iterasi mencapai maksimal (Zhang, 2014).

Data yang digunakan adalah *data set wine_SVM*. Dari *data set* tersebut dibagi menjadi 90 data *training* dan 88 data *test*. *Input* parameter ACO yang digunakan yaitu m = 30, N = 500, ρ = 0.7, Q = 100, α = 1, dan β = 1. Dari implementasi yang telah dilakukan didapatkan akurasi terbaik sebesar 95.4545% dan akurasi rata-rata sebesar 86.3636%. Nilai parameter yang optimal yang didapat yaitu untuk parameter C = 18.605 dan untuk parameter σ = 0.6643. Dari penelitian tersebut dapat disimpulkan bahwa implementasi ACO untuk melakukan pemilihan parameter SVM layak dan efektif, terbukti dari akurasi rata-rata yang mencapai lebih dari 85% (Zhang, 2014).

Referensi *paper* terkait secara menyeluruh ditunjukkan pada Tabel 2.1.

Tabel 2.1 Referensi *paper* terkait

No	Judul	Objek	Metode (Proses)	Hasil (*Output*)
1	Implementasi Algoritma SVM (*Support vector Machine*) Untuk Mengetahui Tingkat Resiko Penyakit Stroke	Objek: data rekam medik hasil tes laboratorium yang dilakukan klasifikasi kedalam 3 tingkat resiko stroke *Input*: Data latih dan data uji dengan 5 fitur, nilai parameter SVM, panjang *fold*	Metode yang digunakan adalah SVM yang dioptimalkan dengan metode *Simplified SMO* dengan proses sebagai berikut : 1. Inisialisasi parameter. 2. Pelatihan SVM 3. SMO *4. One-against-all* 5. Hitung akurasi dengan *K-fold cross validation*.	Nilai akurasi.

Tabel 2.1 Referensi *paper* terkait(lanjutan)

No	Judul	Objek	Metode (Proses)	Hasil (*Output*)
2	Implementasi Metode *Ant Colony Optimization* Untuk Pemilihan Fitur Pada Kategorisasi Dokumen Teks	Objek : pemilihan fitur pada kategorisasi dokumen teks dengan dataset dari sumber berita online Kompas.com. *Input* : nilai parameter ACO seperti nilai awal *pheromone*, nilai *heuristic value*, jumlah populasi semut, dan jumlah iterasi	Proses dibagi menjadi dua sistem, pertama subsistem ACO dan kedua subsistem *NN-Classifier* Untuk subsistem ACO: 1. Merepresentasi *feature space dalam graph*. 2. Inisialisasi awal sistem ACO. 3. Pembuatan *feature subset* dan evaluasi *ant*. 4. Menghitung nilai *pheromone* setiap *ant*. 5. Memeriksa jumlah iterasi. 6. Memperbarui *pheromone.* 7. *Generate* populasi *ant* baru. Untuk subsistem *NN-Classifier*: 1. Membuat matriks vektor model. 2. Mengategorisasi dokumen testing.	Kategorisasi dokumen
3	*A Parallel Way to Select the Parameters of SVM Based on the Ant Optimization Algorithm*	Objek : *dataset wine_SVM*. Input: nilai parameter ACO dan panjang *fold*	1. Inisialisasi parameter. 2. Hitung probabilitas. 3. Cari lintasan semut. 4. Hitung akurasi dengan *K-fold cross validation* dengan pelatihan SVM . 5. *Update pheromone*.	Nilai parameter optimal yang memiliki akurasi rata-rata terbaik

Sumber:[(Afrizal, 2015), (Yudis, 2009), (Zhang, 2014)]

2.2 Stroke

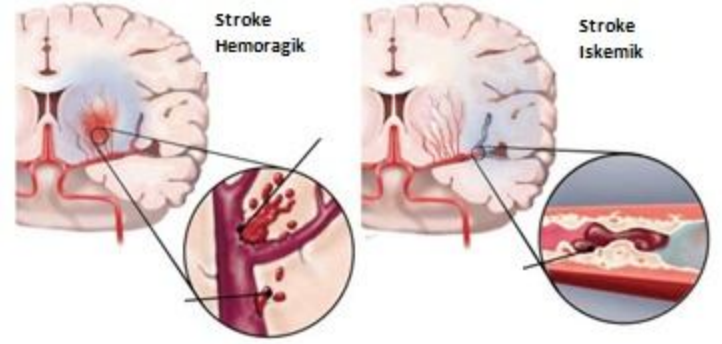
Secara medis stroke didefinisikan sebagai gangguan fungsional otak baik itu gangguan fokal maupun global akut yang berlangsung selama lebih dari 24 jam yang disebabkan oleh terganggunya aliran darah ke otak dan bukan disebabkan oleh gangguan peredaran otak sepintas, tumor otak, stroke sekunder karena trauma ataupun infeksi (WHO MONICA, 1990).

Stroke atau CVA (*Cerebrovascular Accident*) adalah keadaan ketika otak kehilangan fungsi-fungsinya dengan cepat yang disebabkan karena gangguan suplai darah ke otak yang dapat terjadi karena adanya penyumbatan atau karena adanya pendarahan pada pembuluh darah (Muyderman, 2009). Karena gangguan tersebut, maka daerah otak yang terkena stroke tidak dapat berfungsi dengan normal. Gejala-gejala yang dapat diamati ketika seseorang terkena stroke antara lain: tidak mampu menggerakkan anggota tubuh pada salah satu sisi badan (*hemiplegia*), gangguan dalam hal berbicara (*aphasia*), atau gangguan penglihatan berupa kebutaan pada salah satu sisi dari lapang pandang (*visual field*) (Donnan, 2008).

2.2.1 Jenis-jenis Stroke

Secara umum gangguan peredaran darah di otak yang dapat menyebabkan terjadinya stroke dibagi menjadi dua jenis yaitu gangguan karena adanya penyumbatan pembuluh darah otak yang biasa disebut stroke iskemik dan gangguan karena perdarahan yang disebabkan pecahnya pembuluh darah otak yang biasa disebut stroke hemoragik (Yastroki, 2012).

Secara umum stroke iskemik dan stroke hemoragik dapat diilustrasikan seperti Gambar 2.1 dibawah ini.



Gambar 2.1 Stroke Iskemik dan Stroke Hemoragik

Sumber : diadaptasi dari puskesmasbarutengah.com

1. Stroke Iskemik

Dari semua kejadian stroke, sekitar 80% adalah kejadian stroke iskemik. Stroke iskemik disebabkan adanya penyumbatan pembuluh darah otak yang menyebabkan berkurangnya suplai oksigen dan glukosa ke bagian otak yang mengalami penyumbatan (Hacke, 2003). Kekurangan suplai oksigen lama

kelamaan akan mengakibatkan jaringan otak mati (*infark*). Penyumbatan yang terjadi dapat berupa *trombus* (pembekuan darah) atau *embolus*.

a. Stroke Trombotik

Terbentuk *atheroma* (endapan lemak) pada dinding pembuluh darah yang mengakibatkan arteri menjadi tebal dan keras. Dari *atheroma* tersebut dapat memicu terbentuknya pembekuan darah (*trombus*). *Trombus* yang terbentuk di sekitar *atheroma* menyebabkan aliran darah tertutup.

b. Stroke Embolik

Trombus yang terbentuk dari pembuluh darah dibagian tubuh lain di luar otak dapat membuat *emboli* (pecahan lemak) terbawa aliran darah ke otak. *Trombus* juga dapat berasal dari *atrial fibrillation* jantung yang kemudian hanyut menjadi *emboli* yang menyumbat pembuluh darah otak.

2. Stroke Hemoragik

Stroke hemoragik merupakan perdarahan yang terjadi karena pembuluh darah otak yang pecah dan mengakibatkan suplai darah ke otak menjadi terganggu. Stroke hemoragik lebih berbahaya dibandingkan stroke iskemik, karena 1/3 dari kejadiannya berakhir dengan kematian. Menurut lokasi perdarahannya stroke hemoragik dibagi menjadi dua jenis yaitu perdarahan di dalam otak (*intraserebral*) dan perdarahan di antara bagian dalam dan luar lapisan jaringan yang melindungi otak (*subarachnoid*).

a. Hemoragik *intraserebral*

Stroke hemoragik *intraserebral* berhubungan erat dengan hipertensi. Stroke ini terjadi karena adanya perdarahan yang terjadi di dalam otak yang disebabkan oleh pecahnya pembuluh darah.

b. Hemoragik *subarachnoid*

Stroke yang terjadi karena perdarahan pada daerah antara otak dengan rongga tengkorak yang disebabkan pecahnya pembuluh darah. Perdarahan yang terjadi akan sangat membahayakan karena tekanan dalam rongga tengkorak akan meningkat dan dapat menekan jaringan otak yang lain.

2.2.2 Faktor Risiko Stroke

Faktor risiko stroke adalah suatu keadaan atau kondisi kesehatan atau penyakit yang ada pada seseorang yang berisiko terhadap timbulnya serangan stroke. Jika kondisi ini tidak dikendalikan atau diobati maka akan dapat memperburuk dan berakibat terjadinya penyempitan atau pecah pembuluh darah otak. Faktor risiko stroke dibagi menjadi dua yaitu faktor risiko yang dapat diubah/dikendalikan dan faktor risiko yang tidak dapat diubah/dikendalikan (Yastroki, 2012). Beberapa faktor yang paling berpengaruh yang biasanya diperiksa dalam laboratorium untuk deteksi stroke (Soeharto, 2012) :

1. Umur

Kejadian stroke dapat terjadi pada semua umur namun sebagian besar diderita oleh umur di atas 55 tahun, dan setiap penambahan 10 tahun setelah umur 55 tahun terdapat peningkatan resiko stroke sebesar dua kali lipat. Skala pengelompokan ditunjukkan pada Tabel 2.2.

Tabel 2.2 Umur

Range	**Keterangan**
< 35	Muda
35 – 55	Paruh baya
> 55	Tua

Sumber : Soeharto (2004)

2. Total Kolesterol

Total kolesterol adalah kadar keseluruhan kolesterol yang beredar dalam tubuh. Peredaran kolesterol juga melewati pembuluh darah. Skala pengelompokan ditunjukkan pada Tabel 2.3.

Tabel 2.3 Total Kolesterol

Range	**Keterangan**
< 200 mg/dl	Normal
200 – 239 mg/dl	Tinggi
> 239 mg/dl	Sangat tinggi

Sumber : Soeharto (2004)

3. HDL (*High Density Lipoprotein*)

HDL adalah kendaraan yang membawa kolesterol dari berbagai sel-sel jaringan tubuh kembali ke liver. Disebut juga sebagai kolesterol baik karena semakin tinggi HDL akan menyebabkan penurunan dari penumpukan kolesterol, sehingga semakin baik bagi tubuh. Skala pengelompokan ditunjukkan pada Tabel 2.4.

Tabel 2.4 HDL

Range	**Keterangan**
< 35 mg/dl	Terlalu rendah
35 – 60 mg/dl	Menguntungkan
> 60 mg/dl	Sangat menguntungkan

Sumber : Soeharto (2004)

4. LDL (*Low Density Lipoprotein*)

LDL adalah kendaraan yang membawa kolesterol dari liver ke banyak jaringan. Disebut juga sebagai kolesterol jahat karena peranannya membawa kolesterol ke banyak jaringan tubuh sehingga memberikan peluang terjadi penumpukan kolesterol di berbagai jaringan tubuh termasuk pembuluh darah. Skala pengelompokan ditunjukkan pada Tabel 2.5.

Tabel 2.5 LDL

Range	**Keterangan**
< 100 mg/dl	Normal
100 – 199	Batas Tinggi
> 199	Sangat tinggi

Sumber : Soeharto (2004)

5. Trigliserida

Trigliserida adalah sejenis lemak dalam darah yang bermanfaat sebagai sumber energi. Jika konsumsi makanan lebih dari yang diperlukam maka kelebihan kalori akan disimpan sebagai trigliserida dalam jaringan lemak untuk simpanan penggunaan dikemudian waktu jika diperlukan. Skala pengelompokan ditunjukkan pada Tabel 2.6.

Tabel 2.6 Trigliserida

Range	**Keterangan**
< 150 mg/dl	Normal
150 – 199 mg/dl	Batas tinggi
200 – 499 mg/dl	Tinggi
> 500 mg/dl	Sangat tinggi

Sumber : Soeharto (2004)

2.3 Klasifikasi

Klasifikasi adalah mengelompokkan fakta-fakta berdasarkan kriteria tertentu. Klasifikasi dapat dibedakan menjadi dua jenis, yaitu pengelompokan objek menjadi dua kelompok yang disebut klasifikasi sederhana dan klasifikasi kompleks yang mengelompokkan objek menjadi tiga kelompok atau lebih. Menurut Han dan Amber klasifikasi adalah suatu model dalam *data mining* dimana *classifier* dikonstruksi untuk melakukan prediksi kategori atau kelas dari suatu data (Han, 2006).

Proses klasifikasi terdiri dari dua tahapan : pelatihan dan klasifikasi. Pada tahap pelatihan, dilakukan analisa terhadap data training dengan menggunakan algoritma klasifikasi. Pada tahap klasifikasi, dilakukan estimasi akurasi dari aturan-aturan dari klasifikasi dengan menggunakan data tes. Apabila dari

tahapan kedua tersebut didapat akurasi yang sesuai, maka aturan tersebut dapat digunakan pada klasifikasi data yang baru.

2.4 *Support Vector Machine* (SVM)

SVM diperkenalkan oleh Vapnik, Boser, dan Guyon pada tahun 1992. Metode SVM sendiri merupakan metode yang dapat dikatakan masih baru dibandingkan dengan metode yang lain, tetapi memiliki performansi yang lebih baik (Cristianini, 2001). Dalam SVM terdapat *training* set positif dan negatif yang nantinya akan digunakan untuk membuat *hyperplane,* yang akan memisahkan data positif dengan data negatif di ruang *n*-dimensi.

2.4.1 *SVM pada kasus Linearly Separable Data*

Linearly separable data adalah data yang dapat dipisahkan secara linier. Misal $x_i \in \{x_1,...,x_n\}$ adalah *dataset* dan $y_i \in \{+1,-1\}$ adalah label kelas dari data x_i. Pertama yaitu mencari *hyperplane* dengan Persamaan (2.1) dan (2.2) (Vijayakumar, 1999).

$$x_i \cdot w+b \geq 1 \text{ untuk } y_i = +1 \quad (2.1)$$

$$x_i \cdot w+b \leq -1 \text{ untuk } y_i = -1 \quad (2.2)$$

keterangan:

x_i = data ke-*i*
w = vektor yang tegak lurus terhadap *hyperplane*
b = nilai bias (*threshold*)
y_i = kelas data ke-*i*

yang ekuivalen dengan Persamaan (2.3) (Vijayakumar, 1999).

$$y_i(x_i \cdot w+b)-1 \geq 0 \text{ untuk } i = 1,..., n. \quad (2.3)$$

keterangan:

n = jumlah data

Jika $wx_1 + b = +1$ adalah *hyperplane* dari kelas $+1$ atau bisa disebut H_1 dengan jarak *hyperplane* terhadap *hyperplane* asal $d_{H_1} = \frac{(1-b)}{\|w\|}$ dan $wx_2 + b = -1$ merupakan *hyperplane* dari kelas -1 atau bisa disebut H_2 dengan jarak *hyperplane* terhadap *hyperplane* asal $d_{H_2} = \frac{(-1-b)}{\|w\|}$ maka margin antara kedua *hyperplane* tersebut dapat dihitung dengan Persamaan (2.4) (Vijayakumar, 1999).

Margin = $\left| d_{H_1} - d_{H_2} \right| = \frac{2}{\|w\|}$ (2.4)

keterangan:

w = vektor yang tegak lurus terhadap *hyperplane*

Untuk mendapatkan *hyperplane* pemisah kedua kelas, *margin* perlu dilakukan optimasi dengan Persamaan (2.5) dan (2.6) (Vijayakumar, 1999):

Minimize $J_1[w] = \frac{1}{2} \| w \|^2$ (2.5)

dengan $y_i(x_i \cdot w + b) - 1 \geq 0, i = 1, \ldots, l$ (2.6)

keterangan:

x_i = data ke-*i*

y_i = kelas data ke-*i*

b = nilai bias

w = vektor yang tegak lurus terhadap *hyperplane*

2.4.2 SVM pada kasus *Nonlinearly Separable Data*

Untuk mengklasifikasi data yang tidak dapat dipisahkan secara linier, formula SVM harus dilakukan modifikasi. Oleh karena itu kedua bidang pembatas pada Persamaan (2.1) dan (2.2) harus diubah agar lebih fleksibel (Sembiring, 2006). Maka kemudian diperkenalkan variabel *slack* untuk menangani *misclassification* atau kesalahan klasifikasi yang ditunjukkan dengan Persamaan (2.7) (Vijayakumar, 1999).

$y_i(x_i \cdot w + b) - 1 + \xi_i \geq 0$ untuk $i = 1, \ldots, n$ (2.7)

Kemudian memaksimalkan *margin* dan mengurangi jumlah *misclassification* dengan menggunakan parameter *C* yang telah ditetapkan oleh pengguna dengan Persamaan (Vijayakumar, 1999).

Minimize $J_2[w, \xi_i] = (\frac{1}{2} \| w \|^2 + C \sum_{i=1}^{l} \xi_i)$ (2.8)

dengan $y_i(x_i \cdot w + b) - 1 + \xi_i \geq 0,$ (2.9)

$\xi_i \geq 0, \; i = 1, \ldots, n$ (2.10)

keterangan:

x_i = data ke-*i*

y_i = kelas data ke-*i*

b = nilai bias (*threshold*)

ξ_i = variabel *slack*

w = bobot *support vector*

C = parameter yang ditentukan pengguna

n = jumlah data

J_2 = fungsi *minimize*

Persamaan optimasi diatas dapat dituliskan dengan menggunakan *Lagrange multiplier* yang ditunjukkan pada Persamaan (2.11) (Vijayakumar, 1999).

$$Minimize\ L(w,b,\xi_i,h,r) = \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{l}\xi_i - \sum_{i=1}^{l}h_i[y_i(x_i \cdot w + b) - 1 + \xi_i] - \sum_{i=1}^{l}r_i\xi_i \quad (2.11)$$

dimana $h = (h_1, \ldots h_l)$ dan $r = (r_1, \ldots r_l)$ adalah *non-negative Lagrange Multiplier*.

Persamaan (2.11) dapat diubah menjadi *dual problem* yang akan lebih mudah ditangani dengan Persamaan berikut (Vijayakumar, 1999).

$$Maximize \quad L_D(h) = (\sum_{i=1}^{l}h_i - \frac{1}{2}h\ .Dh) \quad (2.12)$$

dengan $$\sum_{i=1}^{l}y_i h_i = 0, \quad (2.13)$$

$$0 \le h_i \le C,\ i = 1, \ldots, n \quad (2.14)$$

dimana $$D_{ij} = y_i y_j x_i \cdot x_j$$

Penyelesaian terhadap *dual problem* ini ditemukan saat menggunakan standard *quadratic programming*. ketika *optimal multiplier* h_i ditemukan, maka klasifikasinya menggunakan fungsi yang ditunjukkan pada Persamaan (2.15) (Vijayakumar, 1999).

$$f(x) = sign\left(\sum_i h_i y_i x_i \cdot x + b\right) \quad (2.15)$$

keterangan:

h_i = optimal *multipliers*

y_i = kelas data ke *i*

x_i = data ke *i*

b = nilai bias

Nilai bias *b* dapat ditemukan dengan persamaan (Boswell, 2002).

$$w \cdot x^+ + b = +1 \quad (2.16)$$

$$w \cdot x^{-} + b = -1 \tag{2.17}$$

Dimana *w* dinyatakan dengan Persamaan (2.18) (Boswell, 2002).

$$w = \sum_{i=1}^{n} \alpha_i y_i x_i \tag{2.18}$$

keterangan:

α_i = *Lagrange Multiplier* data ke-*i*

y_i = kelas data ke-*i*

x_i = data ke-*i*

x^{+} = data kelas positif yang memiliki nilai *Lagrange Multiplier* terbesar

x^{-} = data kelas negatif yang memiliki nilai *Lagrange Multiplier* terbesar

Sehingga persamaan untuk nilai bias *b* didapatkan sebagai berikut (Boswell, 2002).

$$b = -\frac{1}{2}\left(w \cdot x^{+} + w \cdot x^{-}\right) \tag{2.19}$$

Metode lain untuk melakukan klasifikasi data yang tidak dapat dipisah secara linier adalah dengan melakukan transformasi data ke dalam dimensi ruang fitur (*feature space*) sehingga dapat dipisah secara linier pada *feature space* (Sembiring, 2006). Data dilakukan pemetaan dengan fungsi transformasi $x_k \rightarrow \Phi(x_k)$ ke dalam *feature space* sehingga didapatkan bidang pemisah yang dapat memisahkan data sesuai kelas. *Feature space* dalam implementasinya biasanya memiliki dimensi yang lebih tinggi dari *input space*, yang menyebabkan komputasi pada *feature space* dimungkinkan akan sangat besar karena kemungkinan *feature space* memiliki jumlah *feature* tak hingga. Untuk itu digunakan *kernel trick*, dengan fungsi kernel *K* dapat dirumuskan seperti pada persamaan (Sembiring, 2006).

$$K(x_i, x_j) = \Phi(x_i).\Phi(x_j) \tag{2.20}$$

Fungsi transformasi $\Phi(x_k)$ tidak perlu diketahui secara pasti. Sehingga untuk Persamaan (2.15) akan didapatkan fungsi kernel sebagai berikut

$$f(x) = sign\left(\sum_i h_i y_i K(x_i \cdot x) + b\right) \tag{2.21}$$

Syarat sebuah fungsi menjadi fungsi kernel adalah memenuhi *teorema Mercer* yang menyatakan bahwa matriks kernel yang dihasilkan harus bersifat *positive semi-definite*. Beberapa macam fungsi kernel SVM dapat dilihat pada Tabel 2.7.

Tabel 2.7 Berbagai Fungsi Kernel

No	Nama Kernel	Definisi Fungsi
1	*Linear*	$K(x_i, x) = x_i^T \cdot x$
2	*Polynomial of degree up to d*	$K(x,y)=(x\cdot y+c)^d$
3	*Gaussian* RBF	$K(x,y) = \exp\left(\frac{-\|x-y\|^2}{2\sigma^2}\right)$
4	*Sigmoid* (Tangen iperbolik)	$K(x,y) = \tanh(\sigma(x\cdot y)+c)$

Sumber: Diadaptasi dari Sembiring (2006)

2.4.3 *Sequential Learning*

Metode ini dikembangkan untuk klasifikasi, dengan algoritma sebagai berikut (Vijayakumar, 1999):

1. Setelah menghitung kernel dan didapatkan matriks kernel lalu melakukan inisiasi nilai parameter *αi* = 0 dan parameter lain seperti *λ, γ, C, ε,* dan Iterasi Maksimal. Kemudian hitung matriks *Hessian* dengan Persamaan (2.21)

$$D_{ij} = y_i y_j \left(K\left(x_i, x_j\right) + \lambda^2\right) \text{ untuk } i, j = 1, \ldots, n. \quad (2.21)$$

keterangan:

x_i = data ke-*i*

x_j = data ke-*j*

y_i = kelas data ke-*i*

y_j = kelas data ke-*j*

λ = batas teoritis yang akan diturunkan

$K\left(x_i, x_j\right)$ = fungsi kernel

n = jumlah data

2. Tiap *i* = 1 sampai *l*, gunakan persamaan

$$E_i = \sum_{j=1}^{l} \alpha_j D_{ij} \quad (2.22)$$

$$\delta\alpha_i = \min\{\max[\gamma(1-E_i), -\alpha_i], C-\alpha_i\} \quad (2.23)$$

$$\alpha_i = \alpha_i + \delta\alpha_i \quad (2.24)$$

keterangan:

E_i = Error data ke-*i*

α_j = *Lagrange Multiplier* data ke-*j*

D_{ij} = nilai matriks *hessian* dari date ke-*i,j*

γ = parameter kontrol

3. Jika iterasi maksimum telah dicapai atau ketika nilai α mencapai nilai konvergen dimana $\max\left(\left|\delta\alpha_i\right|\right) < \varepsilon$ maka dilanjutkan ke langkah berikutnya. Jika tidak maka kembali ke langkah 2.

4. Kemudian didapat nilai Support Vector $SV = \left(\alpha_i > ThresholdSV\right)$. Nilai $ThresholdSV$ ditentukan setelah melakukan uji coba beberapa kali. Nilai $ThresholdSV$ adalah $ThresholdSV \geq 0$.

2.4.4 *Multi Class* SVM

Pertama kali SVM diperkenalkan, metode *Support Vector Machine* (SVM) hanya digunakan untuk klasifikasi data ke dalam dua kelas. Kemudian *multi class* diperkenalkan untuk menyelesaikan permasalahan klasifikasi data yang memiliki dua kelas atau lebih. Secara umum implementasi *multi class* SVM dapat dijabarkan sebagai berikut

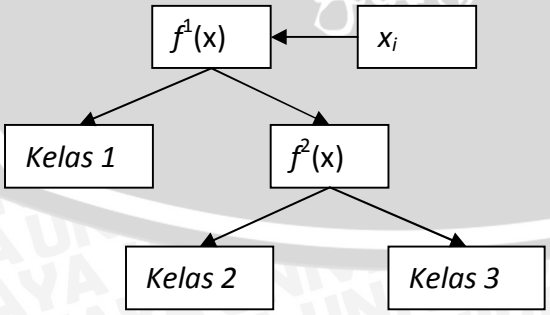
1. Metode *One-Against-All*

Pada metode ini dibangun model SVM *biner*. Tiap model akan dicari solusi permasalahan dengan menggunakan seluruh data. Misal akan melakukan klasifikasi dengan 3 kelas data. Ilustrasi dapat dilihat seperti pada Tabel 2.8 dan Gambar 2.2.

Tabel 2.8 Metode *One-Against-All*

$y_i = 1$	$y_i = -1$	Hipotesis
Kelas 1	Kelas 2 dan Kelas 3	$f^1(x) = (w^1)x + b^1$
Kelas 2	Kelas 3	$f^2(x) = (w^2)x + b^2$

Sumber: Diadaptasi dari Sembiring (2006)



Gambar 2.2 Klasifikasi dengan metode *One-Against-All*

Sumber: Diadaptasi dari Sembiring (2006)

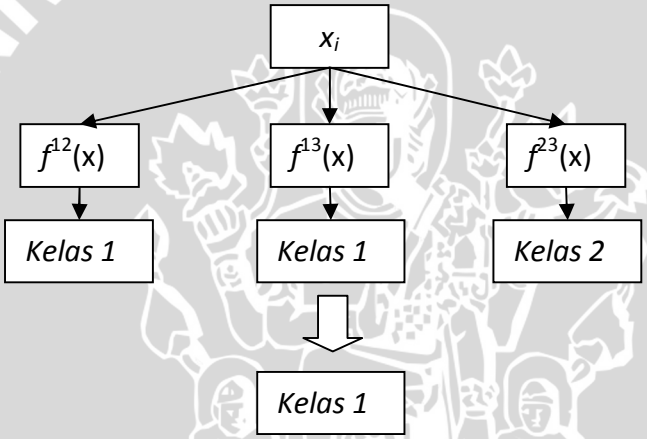
2. Metode *One-Against-One*

Pada metode ini dibangun buah model SVM *biner* dimana *k* adalah jumlah kelas. Tiap model akan dilatih pada data dari dua kelas. Misal akan melakukan klasifikasi dengan 3 kelas data maka akan terbentuk sebanyak 3 buah model SVM *biner*. Ilustrasi untuk kasus ini dapat dilihat seperti pada Tabel 2.9 dan Gambar 2.3.

Tabel 2.9 Metode *One-Against-One* dengan 3 SVM *biner*

$y_i = 1$	$y_i = -1$	Hipotesis
Kelas 1	Kelas 2	$f^{12}(x) = (w^{12})x + b^{12}$
Kelas 1	Kelas 3	$f^{13}(x) = (w^{13})x + b^{13}$
Kelas 2	Kelas 3	$f^{23}(x) = (w^{23})x + b^{23}$

Sumber: Diadaptasi dari Sembiring (2006)



Gambar 2.3 Klasifikasi dengan metode O*ne-Against-One*

Sumber: Diadaptasi dari Sembiring (2006)

Setelah model klasifikasi $\frac{k(k-1)}{2}$ dibangun maka dilakukan pengujian. Jika data *x* dimasukkan ke dalam fungsi dan hasilnya adalah data tersebut masuk ke dalam kelas *i*, maka voting untuk kelas *i* ditambah 1. Kelas dari data *x* akan ditentukan dari voting terbanyak. Jika ada dua kelas atau lebih yang memiliki voting sama, maka akan dipilih kelas dengan indeks terkecil yang akan dipilih menjadi kelas dari data *x*.

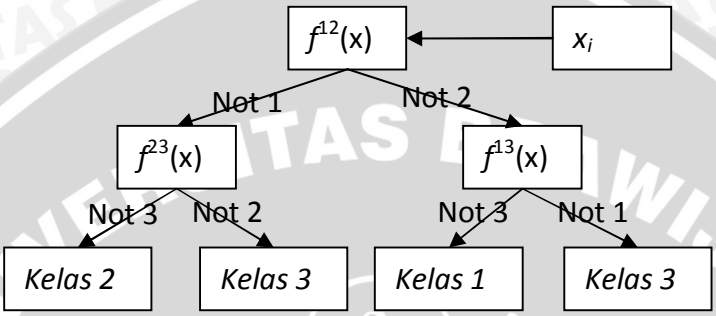
3. Metode DAGSVM (*Directed Acyclic Graph Support Vector Machine*)

Untuk metode ini sama seperti metode *one-against-one*. Namun dalam pengujiannya digunakan *binary directed acyclic graph*. Saat melakukan klasifikasi, hipotesis di klasifikasi mulai dari awal lalu akan bergerak tergantung output dari hipotesis. Misal akan melakukan klasifikasi dengan 3 kelas data maka akan terbentuk sebanyak 3 buah model SVM *biner*. Ilustrasi untuk kasus ini dapat dilihat seperti pada Tabel 2.10 dan Gambar 2.4.

Tabel 2.10 Metode DAGSVM dengan 3 SVM *biner*

$y_i = 1$	$y_i = -1$	Hipotesis
Bukan kelas 2	Bukan kelas 1	$f^{12}(x) = (w^{12})x + b^{12}$
Bukan kelas 3	Bukan kelas 1	$f^{13}(x) = (w^{13})x + b^{13}$
Bukan kelas 3	Bukan kelas 2	$f^{23}(x) = (w^{23})x + b^{23}$

Sumber: Diadaptasi dari Sembiring (2006)



Gambar 2.4 Klasifikasi dengan metode DAGSVM

Sumber: Diadaptasi dari Sembiring (2006)

2.5 *Ant Colony Optimization* (ACO)

Ant Colony Optimization diperkenalkan oleh Dorigo dan Caro pada awal 90-an. Metode ini salah satu dari kelompok *Swarm Intelligence*, yang merupakan suatu algoritma optimisasi yang didasarkan oleh perilaku kerja sama koloni semut yang dapat menemukan jalur terpendek dari sarang menuju sumber makanan. Pada ACO, setiap semut yang telah berjalan melalui jalur tertentu akan meninggalkan informasi berupa *pheromone* (semacam zat kimia). *Pheromone* ini yang nantinya akan menjadi sejenis sinyal bagi semut-semut jalur mana yang akan mereka lewati (Santosa, 2012).

Konsep ACO benar-benar meniru perilaku koloni semut yang secara alami akan mencari rute terpendek saat mencari sumber makanan dan saat kembali ke sarangnya. Misal ada sekumpulan semut yang akan memilih dua jalur untuk menuju ke sumber makanan. Pada saat semut pertama berjalan semut tersebut akan meninggalkan *pheromone* yang akan dapat dideteksi oleh semut dibelakangnya. Pertama kali semut-semut tersebut akan secara *random* memilih jalur mana yang akan dilewati. Namun karena jalur yang pendek akan memungkinkan semut mencapai sumber makanan lebih cepat maka pada rentang waktu yang sama, semut yang melewati jalur yang pendek akan lebih banyak dan akan meninggalkan jumlah *pheromone* lebih banyak daripada semut yang melewati jalur yang lebih panjang. Ditambah lagi adanya *pheromone evaporation* (penguapan *pheromone*) pada jalur yang jarang dilalui sehingga jalur yang lebih panjang tadi lama-kelamaan akan berkurang jumlah *pheromone*nya

sehingga pada akhirnya semut-semut akan terfokus untuk memilih melalui jalur yang lebih pendek.

2.5.1 Karakteristik ACO

Karakteristik ACO adalah sebagai berikut (Yudis, 2009):

1. Menggunakan interaksi agen (*ant*) dimana masing-masing *ant* hanya mampu melakukan tugas sederhana untuk menghasilkan solusi.
2. Menggunakan informasi yang diperoleh dari iterasi sebelumnya berupa *pheromone* untuk menentukan hasil pada iterasi berikutnya.
3. Terdapat mekanisme untuk penambahan *pheromone* pada suatu *node* dan mekanisme untuk pengurangan *pheromone*.

2.5.2 Perilaku Semut

Probabilitas semut *k* pada node *i* akan memilih node *j* yang dituju pada *layer* berikutnya ditunjukkan pada Persamaan (2.25) (Santosa, 2012).

$$p_{i,j} = \begin{cases} \dfrac{\tau_{i,j}^{\alpha}}{\sum_{j \in N_i^{(k)}} \tau_{i,j}^{\alpha}} & \text{, jika } j \in N_i^{(k)} \\ 0 & \text{, untuk yang lain} \end{cases} \quad (2.25)$$

Dimana α menunjukkan derajat kepentingan *pheromone* dan $N_i^{(k)}$ adalah pilihan yang dipunyai semut *k* (*neighborhood*) pada saat ia berada pada node *i*. *Neighborhood* dari semut *k* pada node *i* akan mengandung semua node yang bisa dituju yang tersambung secara langsung ke node *i*, kecuali node yang sudah dituju sebelumnya.

2.5.3 Penambahan dan Penguapan *Pheromone*

Semut *k* saat melewati jalur akan meninggalkan *pheromone*. Jumlah *pheromone* pada jalur *ij* setelah dilewati semut *k* diberikan dengan Persamaan (2.26) (Santosa, 2012).

$$\tau_{i,j} = (1-\rho)\tau_{i,j}, j; \forall(i,j) \in A \quad (2.26)$$

Dimana $\rho \in (0, 1)$ adalah parameter tingkat penguapan dan *A* menyatakan segmen atau jalur yang sudah dilalui oleh semut *k* sebagai bagian dari lintasan dari sarang menuju sumber makanan. Penurunan jumlah *pheromone* memungkinkan semut untuk mengeksploitasi jalur yang berbeda selama proses pencarian. Ini juga akan menghilangkan kemungkinan memilih lintasan yang kurang bagus dan juga membantu membatasi nilai maksimum yang dicapai oleh suatu lintasan *pheromone*. Jumlah *pheromone* yang ditambahkan pada jalur *i – j* oleh semut *k* ditunjukkan pada Persamaan (2.27) (Santosa, 2012).

$$\Delta\tau_{i,j}^{(k)} = \frac{Q}{L_k} \quad (2.27)$$

Dimana Q adalah konstanta dan L_k adalah lintasan terpendek yang dilalui semut k. Nilai Q ditentukan oleh pengguna atau bisa diimplementasikan dengan Persamaan (2.28) (Santosa, 2012).

$$\Delta\tau_{i,j}^{(k)} = \begin{cases} \frac{Q}{L_k} & \text{, jika } (i, j) \in \text{jalur terbaik global} \\ 0 & \text{, untuk yang lain} \end{cases} \quad (2.28)$$

2.5.4 ACO-SVM

Algoritma ACO-SVM merupakan metode gabungan dengan melakukan pemilihan parameter SVM menggunakan ACO. ACO digunakan untuk mencari nilai parameter SVM yang optimal sehingga diharapkan dapat mengurangi kekurangan SVM dalam segi *randomness* dan *running time*. Semut-semut nantinya akan mencari nilai parameter terbaik dengan cara menelusuri titik-titik yang ada. Dengan bantuan proses pelatihan SVM kemudian nilai parameter yang telah didapat dari semut-semut akan dihitung nilai akurasi rata-ratanya menggunakan *K-fold cross validation*. Nilai parameter pada semut dengan akurasi rata-rata terbaik akan digunakan untuk melakukan *update pheromone*. Dengan begitu titik yang telah dilalui oleh semut terbaik akan mengalami peningkatan jumlah *pheromone* sedangkan titik yang tidak dilalui oleh semut terbaik akan mengalami penguapan jumlah *pheromone*. Proses *update pheromone* ini memungkinkan semut pada iterasi selanjutnya untuk memilih titik yang bagus yang nantinya akan dihasilkan nilai akurasi rata-rata yang terbaik (Zhang, 2014).

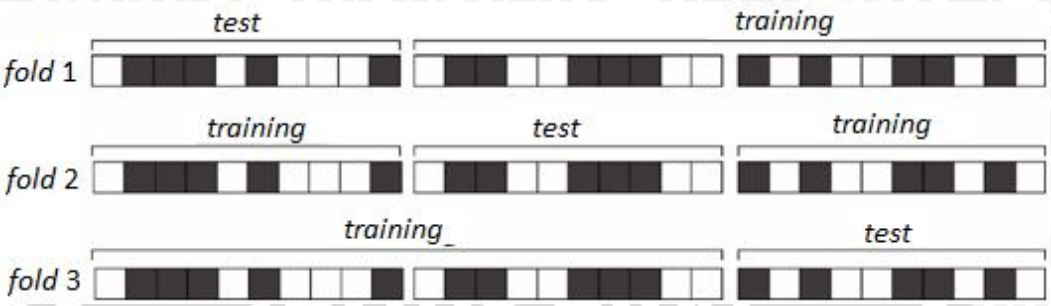
2.6 *K-fold Cross Validation*

K-fold cross validation adalah teknik yang biasa digunakan untuk melakukan pengujian terhadap performa dari suatu model. Misal ada sejumlah *m dataset*, langkah proses *K-fold cross validation* dapat dijabarkan sebagai berikut (Hastie, 2009) :

1. Acak urutan data pada *dataset*.
2. Bagi dataset ke dalam *K fold*.
3. Untuk i = 1, . . ., *K*:
4. Latih model menggunakan semua data yang tidak termasuk *fold i*.
5. Uji model dengan semua data pada *fold i*.
6. Hitung n_i, yaitu jumlah data pada *fold i* yang salah.
7. Hitung *error* dengan persamaan (2.29).

$$E = \frac{\sum_{i=1}^{k} n_i}{m} \tag{2.29}$$

Proses *K-fold cross validation* yang telah dijabarkan di atas dapat diilustrasikan dengan Gambar 2.5.



Gambar 2.5 Proses *K-fold Cross Validation*

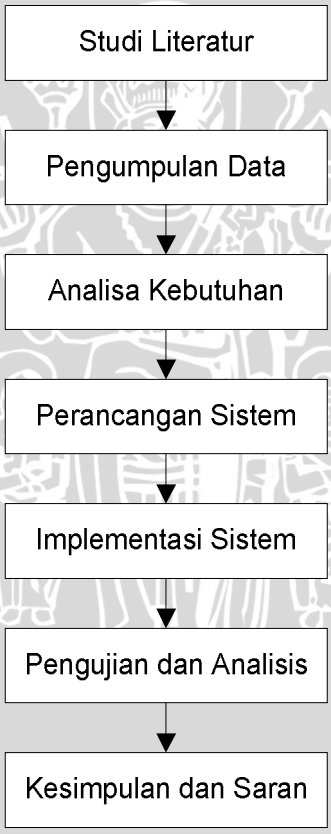
Sumber: Diadaptasi dari Dispel Tutorial 0.8 Documentation (2012)

BAB 3 METODOLOGI PENELITIAN

Pada bab ini dibahas tentang metode penelitian yang akan ditempuh dalam melakukan penelitian tentang implementasi metode SVM berbasis ACO untuk klasifikasi tingkat resiko penyakit stroke, yaitu mengenai tahapan penelitian, analisis kebutuhan sistem, dan formulasi.

3.1 Tahapan Penelitian

Pada bagian tahapan penelitian akan dibahas mengenai langkah-langkah yang akan dilakukan dalam penyusunan sistem implementasi metode SVM berbasis ACO untuk klasifikasi tingkat resiko penyakit stroke. Tahapan tahapan yang akan ditempuh antara lain yaitu mulai dari studi literatur, pengumpulan data, analisa kebutuhan, perancangan, implementasi, dan pengujian dari aplikasi perangkat lunak yang dibuat. Secara umum langkah-langkah penelitian yang akan dilakukan seperti ditunjukkan pada Gambar 3.1 :



Gambar 3.1 Diagram Alir Tahapan Penelitian

3.2 Analisis Kebutuhan Sistem

Analisis kebutuhan sistem merupakan tahapan yang bertujuan untuk mengetahui apa saja yang diperlukan dalam membangun sistem klasifikasi tingkat resiko penyakit stroke, sehingga sistem yang dibuat nantinya akan dapat berjalan dengan baik.

3.2.1 Deskripsi Sistem

Sistem ini merupakan sistem yang akan melakukan optimasi parameter SVM yang nantinya akan digunakan untuk proses klasifikasi. Untuk prosesnya sistem ini membutuhkan data rekam medik pasien yang nantinya akan disimpan di dalam database dan akan digunakan sebagai data uji dan data latih. Sistem ini juga memerlukan beberapa *input* dari pengguna berupa parameter-parameter dari ACO dan SVM. Setelah didapatkan *input* dari pengguna, sistem ini akan melakukan optimasi parameter SVM dengan menggunakan algoritma ACO untuk mendapatkan nilai akurasi terbaik. Hasil dari klasifikasi tingkat resiko penyakit stroke pada sistem ini dibagi dalam 3 kelas status yaitu 1 (normal), 2 (rentan), dan 3 (mengkhawatirkan).

3.2.2 Spesifikasi Kebutuhan Sistem

Secara umum kebutuhan sistem meliputi kebutuhan perangkat keras (*hardware*) dan perangkat lunak (*software*). Kebutuhan perangkat keras yang digunakan yaitu prosesor Intel® Core™ i3 @2.53 GHz (4 CPU), RAM 2 GB, harddisk 300 GB, dan monitor 14". Kebutuhan perangkat lunak yang digunakan yaitu sistem operasi Windows 7 Professional 64-bit, Microsoft Office 2007, Visual Studio 2012, Database MySql, dan bahasa pemrograman yang digunakan adalah C#.

3.2.3 Data Kebutuhan Sistem

Data yang akan digunakan sebagai dataset dalam penelitian ini berupa data rekam medik pasien yang diperoleh dari Laboratorium Klinik Sejahtera. Data ini memiliki lima variabel sebagai inputan terdiri dari umur, total kolesterol, HDL (*High Density Lipoprotein*), LDL (*Low Density Lipoprotein*), dan Trigliserida. Dari variable inputan ini nantinya akan dihasilkan satu variabel keluaran sebagai hasil dari pemrosesan variable inputan yang akan disertakan kedalam *dataset*. Variabel keluaran yang akan dihasilkan nantinya berupa status resiko penyakit stroke yaitu normal, rentan, dan mengkhawatirkan. Tiap variabel masukan dan keluaran yang ada pada *dataset* akan dijelaskan pada Tabel 3.1

Tabel 3.1 Keterangan Variabel

No	Kode Atribut	Keterangan
1	No	Nomor urut data pasien
2	Umur	Umur dari pasien dihitung saat pengecekan
3	Kolesterol Total	Kadar total kolesterol yang beredar di tubuh dalam *mg/dl*
4	HDL	Kadar kolesterol pembawa kolesterol total dari banyak jaringan pada tubuh dalam *mg/dl*
5	LDL	Kadar kolesterol jahat pembawa kolesterol total ke banyak jaringan pada tubuh dalam *mg/dl*

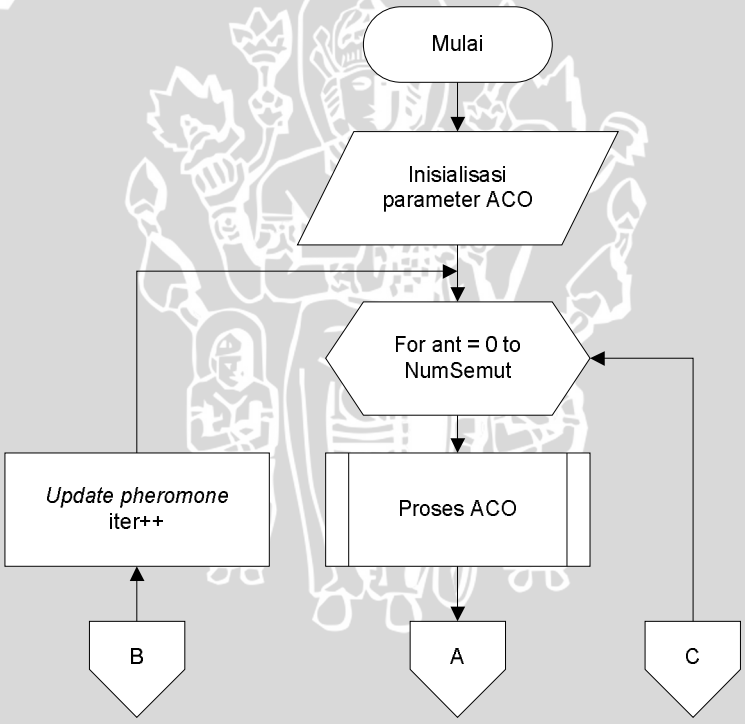
Tabel 3.1 Keterangan Variabel (lanjutan)

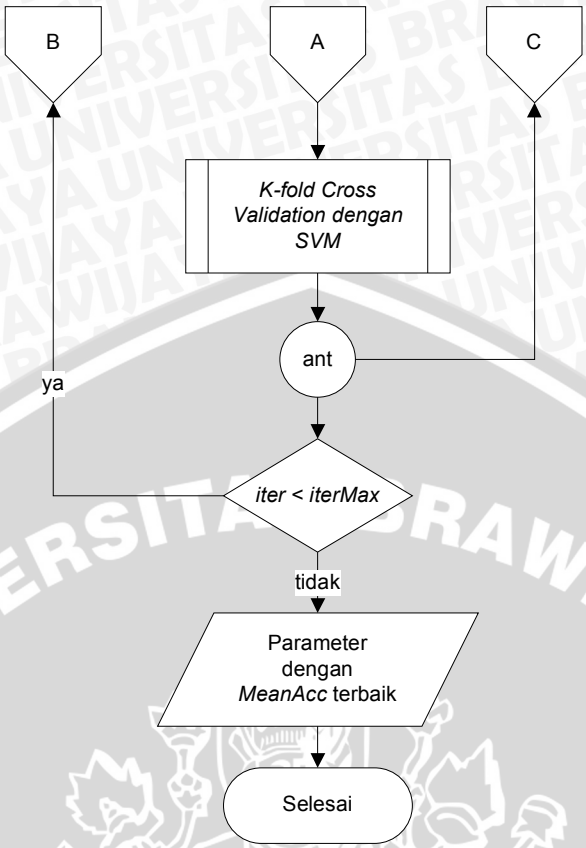
No	Kode Atribut	Keterangan
6	Trigliserida	Trigliserida sejenis lemak hasil penyimpanan kelebihan kalori dalam *mg/dl*
7	Kelas	Tingkat resiko terkena stroke. Dibagi menjadi tiga kelompok yaitu normal, rentan, dan mengkhawatirkan

Sumber : Laboratorium Klinik Sejahtera Probolinggo, 2014

3.3 Formulasi ACO

Metode ACO pada penelitian ini digunakan untuk mengoptimasi parameter dari SVM sebanyak iterasi yang telah ditentukan. Tujuan dari optimasi parameter sendiri adalah untuk mendapatkan nilai parameter yang optimal sehingga nanti ketika parameter SVM yang telah dioptimasi digunakan untuk proses klasifikasi akan didapatkan nilai akurasi yang baik. Proses umum formulasi ACO untuk menyelesaikan permasalahan dapat ditunjukkan pada Gambar 3.2.





Gambar 3.2 Diagram Alir Proses ACO-SVM

Parameter SVM yang dioptimasi adalah parameter *cost* (*C*), dan varian (*σ*). Pada studi kasus ini digunakan sistem koordinat dua dimensi untuk membantu memberi solusi permasalahan, dimana untuk sumbu *X* diibaratkan layer, yang didefinisikan sebagai jumlah digit dari nilai parameter *C* dan *σ* yaitu sebanyak 10 digit, 5 digit pertama untuk parameter *C* dan 5 digit terakhir untuk parameter *σ*. Sumbu *Y* didefinisikan sebagai titik kandidat yang akan dilalui oleh semut, direpresentasikan dengan angka 0 sampai 9.

Proses ACO untuk optimasi parameter dari SVM adalah sebagai berikut :

1. Menentukan jumlah iterasi, jumlah semut, dan parameter ACO

Proses awal yaitu memberikan nilai untuk jumlah iterasi, jumlah semut, dan parameter ACO. Dimisalkan ada sejumlah *m* semut. Tiap semut *k* memiliki *array* satu dimensi yang memiliki *n* elemen, dimana elemen *n* adalah jumlah total dari digit nilai parameter *C* dan *σ*, yaitu sebanyak 10 digit. *Array* satu dimensi nanti akan digunakan untuk menyimpan tiap titik yang dikunjungi semut *k*. Parameter ACO lain yaitu *ρ, Q, c, α, β,* dan *η*. Nilai ρ nantinya digunakan sebagai parameter untuk evaporasi *pheromone*. Nilai *Q* adalah konstan, digunakan untuk melakukan *update pheromone*. Untuk *c* adalah sebagai konstanta, yaitu sebagai nilai untuk pengendali saat dilakukan *update pheromone* agar tidak dihasilkan nilai tak terhingga (*undefined*). Parameter *α* adalah parameter pengendali intensitas jejak semut. Parameter *β* sebagai parameter pengendali visibilitas. Parameter *η*

sebagai visibilitas antara titik a dan titik b. Misal jumlah iterasi = 2, jumlah semut = 5, ρ = 0,7, Q = 1, c = 0,01, α = 1, β = 1, dan η = 1.

2. Menentukan nilai *pheromone* awal dari tiap titik

Pheromone awal (τ_0) dari tiap titik ditentukan oleh pengguna. Misal untuk studi kasus ini ditentukan nilai *pheromone* awal dari setiap titik $\tau_0(x = 0 \sim 9, y = 0 \sim 9) = 1$. Tabel *pheromone* awal yang terbentuk ditunjukkan pada Tabel 3.2

Tabel 3.2 Tabel *Pheromone* Awal

$\tau_0(x,y)$	0	1	2	3	4	5	6	7	8	9
0	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1
2	1	1	1	1	1	1	1	1	1	1
3	1	1	1	1	1	1	1	1	1	1
4	1	1	1	1	1	1	1	1	1	1
5	1	1	1	1	1	1	1	1	1	1
6	1	1	1	1	1	1	1	1	1	1
7	1	1	1	1	1	1	1	1	1	1
8	1	1	1	1	1	1	1	1	1	1
9	1	1	1	1	1	1	1	1	1	1

3. Menghitung nilai probabilitas tiap titik

Berdasarkan nilai *pheromone* awal yang sudah ditentukan, kemudian dihitung nilai probabilitas setiap titik untuk dilalui oleh tiap semut k dengan Persamaan (3.1).

$$P(x,y) = \frac{\tau^{\alpha}(x,y)\eta^{\beta}(x,y)}{\sum_{j=0}^{9}\tau^{\alpha}(x,y)\eta^{\beta}(x,y)} \tag{3.1}$$

Dengan menggunakan nilai parameter yang telah diinisialisasi pada langkah sebelumnya, maka untuk perhitungan nilai probabilitas x = 0 dan y = 0, $P(0,0)$ didapat:

$$P(0,0) = \frac{1^1 * 1^1}{\sum_{j=0}^{9}\left(1^1 * 1^1\right) + \left(1^1 * 1^1\right) + \ldots + \left(1^1 * 1^1\right)} = \frac{1}{10} = 0{,}1$$

Contoh hasil perhitungan nilai probabilitas tiap titik ditunjukkan pada tabel 3.3. Untuk iterasi pertama nilai probabilitas tiap titik bernilai sama. Artinya tiap titik memiliki probabilitas yang sama untuk dipilih oleh semut.

Tabel 3.3 Tabel Nilai Probabilitas

P(x,y)	0	1	2	*3*	*4*	5	6	7	8	9
1	0,1	0,1	0,1	0,1	0,1	0,1	0,1	0,1	0,1	0,1
2	0,1	0,1	0,1	0,1	0,1	0,1	0,1	0,1	0,1	0,1
3	0,1	0,1	0,1	0,1	0,1	0,1	0,1	0,1	0,1	0,1
4	0,1	0,1	0,1	0,1	0,1	0,1	0,1	0,1	0,1	0,1
5	0,1	0,1	0,1	0,1	0,1	0,1	0,1	0,1	0,1	0,1
6	0,1	0,1	0,1	0,1	0,1	0,1	0,1	0,1	0,1	0,1
7	0,1	0,1	0,1	0,1	0,1	0,1	0,1	0,1	0,1	0,1
8	0,1	0,1	0,1	0,1	0,1	0,1	0,1	0,1	0,1	0,1
9	0,1	0,1	0,1	0,1	0,1	0,1	0,1	0,1	0,1	0,1

4. Memilih titik yang akan dilalui tiap semut dengan *Roulette Wheel*

Setelah dihitung nilai probabilitas, kemudian tiap semut *k* akan memilih titik yang akan dilalui. Pemilihan titik dilakukan dengan bantuan *Roulette Wheel*. Pertama dari nilai probabilitas akan dihitung nilai probabilitas kumulatif. Setelah didapatkan nilai probabilitas kumulatif kemudian dihitung *range* dari tiap titik. Nilai *range* ditunjukkan pada Tabel 3.4.

Tabel 3.4 Perhitungan *Roulette Wheel*

y	probabilitas	probabilitas kumulatif	*range*
0	0,1	0,1	0,0 - 0,1
1	0,1	0,2	0,1 - 0,2
2	0,1	0,3	0,2 - 0,3
3	0,1	0,4	0,3 - 0,4
4	0,1	0,5	0,4 - 0,5
5	0,1	0,6	0,5 - 0,6
6	0,1	0,7	0,6 - 0,7
7	0,1	0,8	0,7 - 0,8
8	0,1	0,9	0,8 - 0,9
9	0,1	1	0,9 - 1

Kemudian setelah diketahui *range* dari tiap titik, dibangkitkan bilangan random antara 0 – 1 pada tiap semut *k*. Misal nilai bilangan *random* masing-masing semut *k* ditunjukkan pada Tabel 3.5.

Tabel 3.5 Tabel Bilangan *Random* semua Semut

Semut	Bilangan *random*
1	0,277858
2	0,733929
3	0,119857
4	0,333094
5	0,282218

Langkah selanjutnya yaitu mengecek bilangan *random* dengan *range* dari tiap titik. Misal untuk semut 1 dengan nilai bilangan *random* 0,27785836 ketika dilakukan pengecekan dengan *range* maka akan terpilih titik 2 sebagai kandidat terpilih. Maka semut 1 akan memilih titik 2 untuk dilalui. Titik terpilih untuk masing-masing semut saat *x*=1 ditunjukkan pada Tabel 3.6 dan saat *x*=10 pada Tabel 3.7.

Tabel 3.6 Tabel *Path* Semut pada *x* = 1

Semut	bilangan *random*	poin yang dipilih	*Path*
1	0,277858	2	[2]
2	0,733929	7	[7]
3	0,119857	1	[1]
4	0,333094	3	[3]
5	0,282218	2	[2]

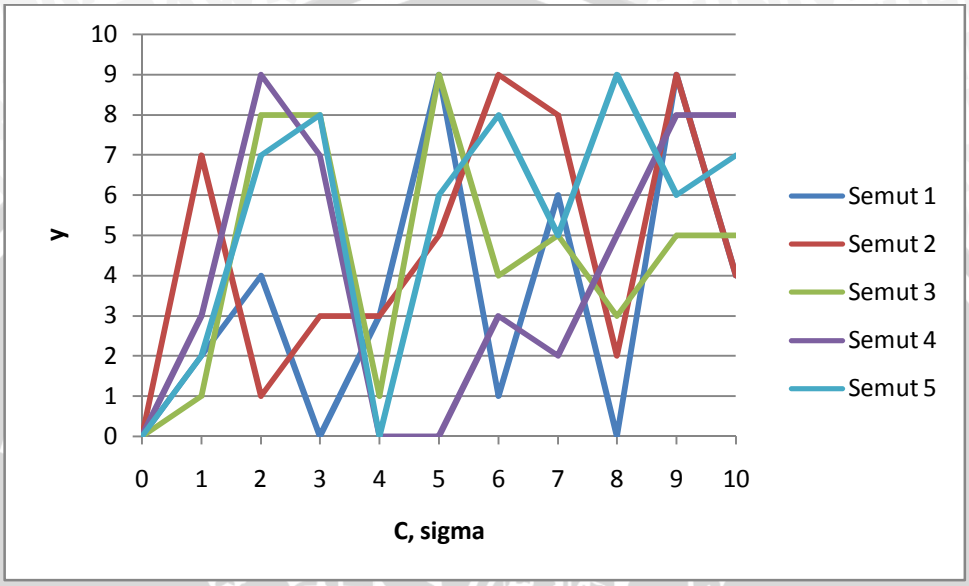
Tabel 3.7 Tabel *Path* Semut pada *x* = 10

Semut	bilangan *random*	poin yang dipilih	*Path*
1	0,417232	4	[2 4 0 3 9 1 6 0 9 4]
2	0,473618	4	[7 1 3 3 5 9 8 2 9 4]
3	0,545405	5	[1 8 8 1 9 4 5 3 5 5]
4	0,896417	8	[3 9 7 0 0 3 2 5 8 8]

Tabel 3.7 Tabel *Path* Semut pada *x* = 10

Semut	bilangan *random*	poin yang dipilih	*Path*
5	0,707881	7	[2 7 8 0 6 8 5 9 6 7]

Path semua semut yang telah ditunjukkan pada Tabel 3.7 dapat disajikan dalam bentuk gambar grafik yang ditunjukkan pada gambar 3.3.



Gambar 3.3 Grafik *path* semua Semut

5. Melakukan kalkulasi jalur yang telah dilalui semut

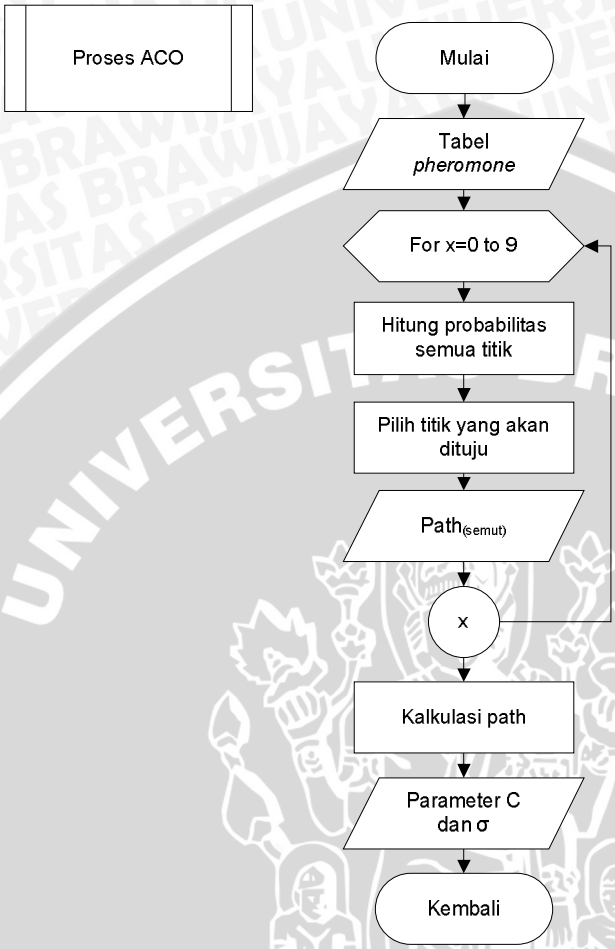
Setelah semua semut selesai memilih semua jalur, yaitu ketika *x* = 10 semut akan menyimpan titik dipilih ke dalam *array*, maka kemudian dilakukan kalkulasi terhadap jalur yang tadi sudah dilalui semut yang nantinya akan dijadikan nilai parameter *C* dan *σ*. Untuk 5 digit pertama akan menjadi nilai parameter *C* dan 5 digit terakhir sebagai nilai parameter *σ*. Tabel 3.8 menunjukkan jalur dari masing-masing semut yang telah diperbarui.

Tabel 3.8 Pembagian Nilai Parameter

Semut	*C*	*σ*
1	[2 4 0 3 9]	[1 6 0 9 4]
2	[7 1 3 3 5]	[9 8 2 9 4]
3	[1 8 8 1 9]	[4 5 3 5 5]
4	[3 9 7 0 0]	[3 2 5 8 8]
5	[2 7 8 0 6]	[8 5 9 6 7]

Kemudian ditentukan batasan untuk masing-masing parameter, untuk studi kasus ini ditentukan batasan untuk nilai parameter *C* sebesar 100,00 ~ 999,99

dan untuk nilai parameter σ sebesar 0,0001 ~ 9,9999. Proses inisialisasi tabel *pheromone* sampai perhitungan kalkulasi jalur lebih jelasnya dapat dilihat pada Gambar 3.4.



Gambar 3.4 Diagram Alir Proses ACO

Hasil dari kalkulasi akhir ditunjukkan pada Tabel 3.9.

Tabel 3.9 Nilai Parameter Setelah Kalkulasi

Semut	*C*	σ
1	240,39	1,6094
2	713,35	9,8294
3	188,19	4,5355
4	397	3,2588
5	278,06	8,5967

6. Menghitung akurasi dengan *K-fold cross validation*

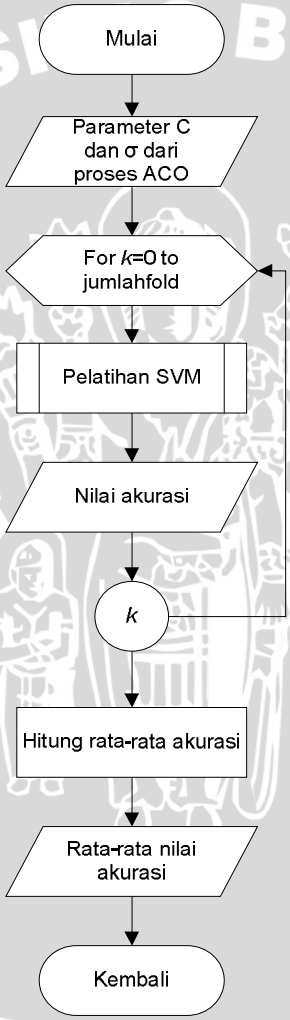
Setelah didapat nilai parameter *C* dan σ, kemudian menentukan banyak nya *K-fold* yang akan digunakan untuk menghitung akurasi. Pada studi kasus ini

digunakan banyak *fold* sebanyak 3 *fold*. Dengan menggunakan pelatihan SVM, masing-masing *fold* dihitung nilai akurasi (Acc_{foldi}) dan akurasi rata-rata nya ($MeanAcc_{ant}$) dengan Persamaan (3.2) dan (3.3).

$$Acc_{foldi} = \frac{Benar}{Benar + Salah} \tag{3.2}$$

$$MeanAcc_{ant} = \frac{\sum_{i=1}^{K} Acc_i}{jumlahfold} \tag{3.3}$$

Proses perhitungan akurasi rata-rata dengan *K-fold cross validation* dapat diilustrasikan pada Gambar 3.5.



Gambar 3.5 Diagram Alir Proses *K-fold Cross Validation*

Hasil dari penghitungan akurasi dengan *k-fold cross validation* ditunjukkan pada Tabel 3.10.

Tabel 3.10 Hasil Perhitungan Akurasi dan Akurasi Rata-rata

Semut	*C*	*σ*	*fold* 1	*fold* 2	*fold* 3	*MeanAcc*
1	240,39	1,6094	0,5	0,66666667	0,5	0,55555556
2	713,35	9,8294	0,66666667	0,66666667	0,5	0,61111111
3	188,19	4,5355	0,5	0,5	0,33333333	0,44444444
4	397	3,2588	0,5	0,5	0,33333333	0,44444444
5	278,06	8,5967	0,5	0,66666667	0,33333333	0,5

7. *Update* nilai *pheromone*

Update nilai *pheromone* dilakukan dengan mengambil nilai akurasi terbaik dari semua semut. Misal dari 5 semut diketahui bahwa semut 2 memiliki nilai akurasi rata-rata terbaik dari 4 semut lainnya yaitu sebesar 0,61111111 dengan nilai parameter *C* = 713,35 dan nilai parameter σ = 9,8294. Tiap titik yang dilalui oleh semut 2 akan mengalami penambahan *pheromone* sedangkan untuk titik yang tidak dilalui oleh semut 2 akan mengalami penguapan *pheromone*. Rumus untuk *update pheromone* ditunjukkan pada Persamaan (3.4).

$$\tau(x,y) = \rho\tau(x,y) + \Delta\tau(x,y) \tag{3.4}$$

$$\Delta\tau(x,y) = \begin{cases} \dfrac{Q}{(1-Acc_i)+c} & \text{, semut terbaik melalui titik (x, y)} \\ 0 & \text{, untuk yang lain} \end{cases} \tag{3.5}$$

Dengan menggunakan nilai *Q* = 1 dan *c* = 0,01 yang telah diinisialisasi pada langkah awal, maka perhitungan nilai *Δτ* untuk *x* = 0 dan *y* = 7 sesuai dengan Persamaan (3.5):

$$\Delta\tau_{(0,7)} = \frac{Q}{(1-Acc_2)+c} = \frac{1}{(1-0{,}61111111)+0{,}01} = \frac{1}{3{,}9888889} = 2{,}50696378$$

Kemudian perhitungan nilai *pheromone* untuk *x* = 0 dan *y* = 7 dengan Persamaan (3.4):

$$\begin{aligned} \tau_{(0,7)} &= \rho\tau_{(0,7)} + \Delta\tau_{(0,7)} \\ &= (0{,}7 * 1) + 2{,}50696378 \\ &= 3{,}20696378 \end{aligned}$$

Dapat dilihat bahwa untuk titik *y* = 7 saat *x* = 0, nilai *pheromone* yang dihasilkan setelah dilakukan *update pheromone* bertambah. Ini dikarenakan titik *y* = 7 saat *x* = 0 dilalui oleh semut terbaik. Untuk perhitungan *update pheromone* titik yang lain, misal perhitungan nilai *Δτ* untuk *x* = 0 dan *y* = 4 sesuai dengan Persamaan (3.5):

$$\Delta\tau_{(0,4)} = 0$$

Kemudian perhitungan nilai *pheromone* untuk *x* = 0 dan *y* = 7 dengan Persamaan (3.4):

$$\begin{aligned}\tau_{(0,4)} &= \rho\tau_{(0,4)} + \Delta\tau_{(0,4)} \\ &= (0{,}7 * 1) + 0 \\ &= 0{,}7\end{aligned}$$

Hasil perhitungan *update pheromone* berupa tabel *pheromone* setelah dilakukan pembaruan ditunjukkan pada Tabel 3.11.

Tabel 3.11 Tabel *Pheromone* setelah di *Update*

τ(x,y)	0	1	2	*3*	*4*	5	6	7	8	9
0	0,7	0,7	0,7	0,7	0,7	0,7	0,7	0,7	0,7	0,7
1	0,7	3,21	0,7	0,7	0,7	0,7	0,7	0,7	0,7	0,7
2	0,7	0,7	0,7	0,7	0,7	0,7	0,7	3,21	0,7	0,7
3	0,7	0,7	3,21	3,21	0,7	0,7	0,7	0,7	0,7	0,7
4	0,7	0,7	0,7	0,7	0,7	0,7	0,7	0,7	0,7	3,21
5	0,7	0,7	0,7	0,7	3,21	0,7	0,7	0,7	0,7	0,7
6	0,7	0,7	0,7	0,7	0,7	0,7	0,7	0,7	0,7	0,7
7	3,21	0,7	0,7	0,7	0,7	0,7	0,7	0,7	0,7	0,7
8	0,7	0,7	0,7	0,7	0,7	0,7	3,21	0,7	0,7	0,7
9	0,7	0,7	0,7	0,7	0,7	3,21	0,7	0,7	3,21	0,7

Ketika iterasi < *iterasiMax*, maka akan kembali ke langkah 3 yaitu menghitung nilai probabilitas menggunakan *pheromone* yang telah diperbarui. Ketika iterasi telah mencapai maksimal maka algoritma ACO berhenti. Jalur terbaik terakhir yang memiliki akurasi terbaik akan dijadikan sebagai hasil akhir optimasi nilai parameter *C* dan *σ*.

3.4 Formulasi SVM

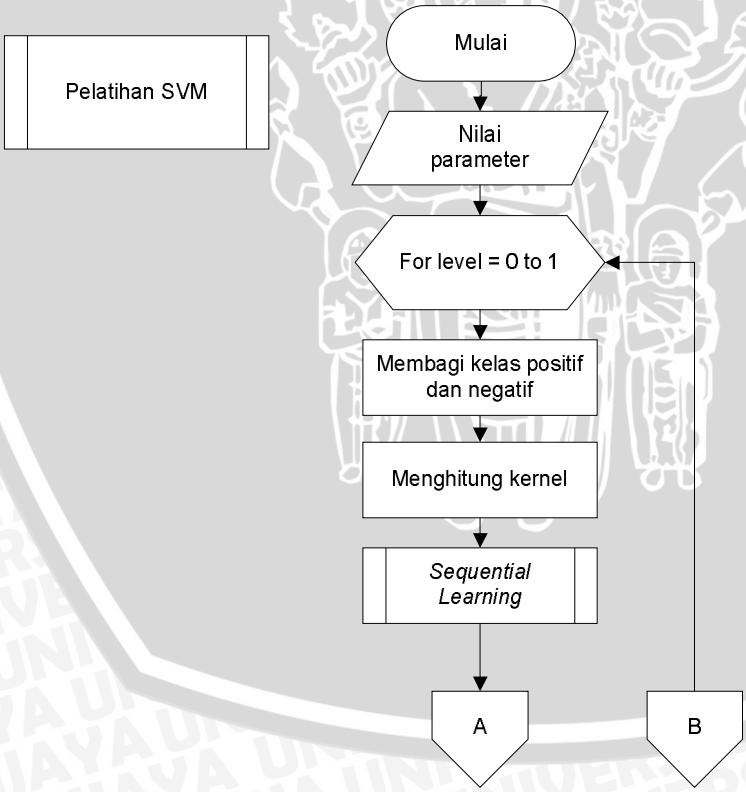
Terdapat sebuah studi kasus yaitu melakukan klasifikasi tingkat resiko penyakit stroke dengan menggunakan SVM. Klasifikasi dilakukan dengan menggunakan kernel RBF dan dengan metode *multi class One-Against-All* untuk mendapatkan status resiko. Status resiko stroke yang dihasilkan sistem akan dibandingkan dengan hasil yang sebenarnya yang nanti digunakan untuk menghitung akurasi dari klasifikasi SVM.

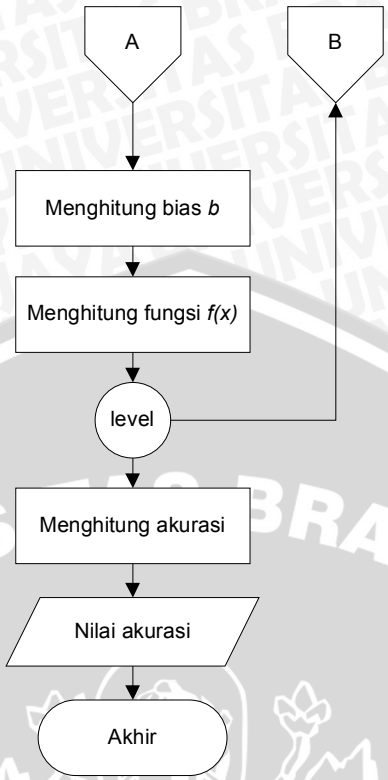
Proses klasifikasi dengan SVM adalah sebagai berikut :

1. Inisialisasi nilai parameter.

2. Menentukan kelas positif dan negatif dari data latih.
3. Menghitung kernel RBF.
4. *Sequential learning.*
    a. Menghitung matriks *Hessian*
    b. Iterasi *alpha*:
        - Menghitung nilai *error* E_i
        - Menghitung nilai $\delta\alpha_i$
        - Memperbarui nilai α_i
5. Menghitung nilai bias *b*.
6. Menghitung nilai fungsi *f(x).*
7. Kembali ke langkah 2 untuk perhitungan level 2 *One-Against-All.*
8. Mencari nilai akurasi.

Proses klasifikasi dengan SVM diatas dapat diilustrasikan dengan diagram alir yang ditunjukkan pada Gambar 3.6.





Gambar 3.6 Diagram Alir Proses Pelatihan SVM

Pada studi kasus ini digunakan 18 data pasien untuk dijadikan *dataset* yang akan ditunjukkan pada Tabel 3.12 yang diambil dari *dataset* pasien dari tiap-tiap kelas.

Tabel 3.12 *Dataset*

No	Umur	Kolestrol Total	HDL	LDL	Trigliserida	Kelas
1	44	174	36,5	123,3	71	1
2	60	169	36,9	110,1	110	1
3	23	196	40,3	137,1	93	1
4	41	213	47,4	143,2	112	1
5	40	253	46,5	179,3	136	1
6	48	197	30,5	120,5	79	1
119	55	255	37,8	184,6	163	2
120	42	191	40,1	117,1	169	2
121	42	242	42,5	162,5	185	2
122	25	210	40,1	134,1	179	2
123	35	197	39,6	119,6	189	2

Tabel 3.12 *Dataset*

No	Umur	Kolestrol Total	HDL	LDL	Trigliserida	Kelas
124	52	197	30,6	92,6	169	2
156	53	245	45,7	56,5	214	3
157	44	291	46,7	180,5	319	3
158	53	223	39,4	132,2	252	3
159	26	266	40,2	181,2	223	3
160	67	276	51,4	188,2	182	3
161	79	340	31,2	208,9	162	3

Seperti ditunjukkan pada tabel, nilai dari kolom F1 merupakan nilai dari fitur umur, kolom F2 menunjukkan nilai fitur kolesterol total, kolom F3 menunjukkan nilai fitur HDL, kolom F4 menunjukkan nilai fitur LDL, kolom F5 menunjukkan nilai fitur trigliserida, dan nilai kolom Kelas menunjukkan status resiko stroke. Untuk studi kasus ini dataset kemudian akan dibagi menjadi 3 *fold* dengan jumlah data yang sama banyaknya. Sebelum melakukan pembagian data terlebih dahulu dilakukan pengacakan. Pembagian data kedalam 3 *fold* ditunjukkan pada Tabel 3.13.

Tabel 3.13 Pembagian *Dataset* kedalam 3 *fold*

No	F1	F2	F3	F4	F5	Kelas	fold
1	44	174	36,5	123,3	71	1	
2	60	169	36,9	110,1	110	1	
119	55	255	37,8	184,6	163	2	fold 1
120	42	191	40,1	117,1	169	2	
156	53	245	45,7	56,5	214	3	
157	44	291	46,7	180,5	319	3	
3	23	196	40,3	137,1	93	1	
4	41	213	47,4	143,2	112	1	
121	42	242	42,5	162,5	185	2	
122	25	210	40,1	134,1	179	2	fold 2
158	53	223	39,4	132,2	252	3	
159	26	266	40,2	181,2	223	3	
5	40	253	46,5	179,3	136	1	fold 3

Tabel 3.13 Pembagian *Dataset* kedalam 3 *fold*

No	F1	F2	F3	F4	F5	Kelas	fold
6	48	197	30,5	120,5	79	1	
123	35	197	39,6	119,6	189	2	
124	52	157	30,6	92,6	169	2	
160	67	276	51,4	188,2	182	3	
161	79	340	31,2	208,9	162	3	

Untuk studi kasus ini data uji yang digunakan adalah *fold* 3 yang berisi 6 data, dan 12 data yang lain selain *fold* 3 akan menjadi data latih.

Penjelasan dari diagram alir diatas akan dijabarkan sebagai berikut :

1. Inisialisasi nilai parameter

Inisialisasi nilai parameter SVM yang akan digunakan ditentukan oleh pengguna. Nilai parameter yang diset harus lebih dari nol. Untuk kasus ini parameter yang dilakukan inisialisasi adalah parameter *augmenting factor* (λ), dan *gamma* (γ), untuk nilai parameter *cost* (C) dan varian (σ) pada studi kasus ini digunakan nilai parameter yang didapat dari hasil pencarian ACO dari semut 2 dengan nilai C = 713,35 dan nilai σ = 9,8294. Misal parameter yang telah di inisialisasi ditunjukkan pada Tabel 3.14.

Tabel 3.14 Tabel Inisialisasi Parameter

λ	γ	C	σ
0,2	0,1	713,35	9,8294

2. Menentukan kelas positif dan negatif dari data latih

Pada studi kasus ini , terdapat 3 kelas yang akan digunakan sebagai acuan untuk pengklasifikasian data, untuk itu digunakan metode *One-Against-All* untuk membantu memberi solusi permasalahan. Dalam proses *One-Against-All*, data latih yang digunakan perlu dilakukan pembagian antara kelas positif dan negatif. Untuk *level* 1 data latih sebanyak 12 data dengan kelas 1 (normal) adalah sebagai kelas positif sedangkan data dengan kelas 2 (rentan)dan 3 (mengkhawatirkan) adalah sebagai kelas negatif. Penentuan kelas positif dan negatif untuk *level* 1 ditunjukkan pada Tabel 3.15.

Tabel 3.15 Tabel Penentuan Kelas Positif dan Negatif

Data Latih Nomor	Kelas	y
1	1	1
2	1	1

Tabel 3.15 Tabel Penentuan Kelas Positif dan Negatif

Data Latih Nomor	Kelas	*y*
119	2	-1
120	2	-1
156	3	-1
157	3	-1
3	1	1
4	1	1
121	2	-1
122	2	-1
158	3	-1
159	3	-1

3. Menghitung kernel RBF

Pada tahap ini akan dilakukan perhitungan *kernel gaussian* RBF yang dilakukan dengan menghitung semua data latih yang telah disediakan dengan Persamaan (3.6).

$$K(x,y)=\exp\left(-\frac{\|x-y\|^2}{2\sigma^2}\right) \tag{3.6}$$

Variabel *x* dan *y* merupakan 2 data *training* yang akan dicari nilai *dot product* menggunakan fungsi kernel yang telah ditentukan. Nilai parameter varian menggunakan nilai yang telah didapatkan pada saat proses ACO. Untuk *One-Against-All* level 1, data latih yang digunakan sebanyak 12 data latih, maka *kernel gaussian* RBF yang dihasilkan berupa matriks 12 x 12. Untuk perhitungan baris pertama kolom pertama, K(1,1):

$$K(1,1)=\exp\left(\frac{-\|x-y\|^2}{2\sigma^2}\right)$$

$$=\exp\left(-\frac{\begin{array}{l}(44-44)^2+(174-174)^2+(36{,}5-36{,}5)^2+(123{,}3-123{,}3)^2\\+(71-71)^2\end{array}}{2*(9{,}8294)^2}\right)$$

$$=1$$

Sehingga diperoleh matriks kernel seperti yang ditunjukkan dalam Tabel 3.16.

Tabel 3.16 Tabel matriks kernel

	1	2	3	4	5	6	7	8	9	10	11	12
1	1	3,6144E-05	3,2367E-43	4,3746E-23	2,0416E-68	2,6073E-177	2,3594E-04	4,2287E-09	7,1696E-44	5,8848E-31	3,9294E-80	5,7189E-80
2	3,6144E-05	1	3,4042E-36	1,6878E-10	9,2954E-45	2,7847E-144	9,3497E-08	1,3132E-08	2,5768E-32	2,8237E-19	8,0492E-54	1,4815E-64
3	3,2367E-43	3,4042E-36	1	1,2062E-20	7,9358E-44	8,0214E-59	5,9877E-27	4,8964E-15	1,0121E-03	1,2797E-13	5,1314E-27	5,1046E-11
4	4,3746E-23	1,6878E-10	1,2062E-20	1	1,9908E-20	6,5356E-83	1,7856E-15	9,0580E-11	8,5797E-12	4,6222E-03	2,6916E-19	9,8657E-30
5	2,0416E-68	9,2954E-45	7,9358E-44	1,9908E-20	1	5,2925E-65	1,0213E-55	1,2336E-43	3,4811E-28	1,3402E-21	4,9925E-18	1,4843E-38
6	2,6073E-177	2,7847E-144	8,0214E-59	6,5356E-83	5,2925E-65	1	4,0393E-141	7,4933E-114	2,9563E-47	2,8513E-65	9,4083E-27	1,1431E-23
7	2,3594E-04	9,3497E-08	5,9877E-27	1,7856E-15	1,0213E-55	4,0393E-141	1	4,1116E-03	8,8959E-27	8,0844E-18	2,9101E-61	4,1041E-54
8	4,2287E-09	1,3132E-08	4,8964E-15	9,0580E-11	1,2336E-43	7,4933E-114	4,1116E-03	1	1,7359E-15	1,0221E-11	9,6552E-46	1,3418E-38
9	7,1696E-44	2,5768E-32	1,0121E-03	8,5797E-12	3,4811E-28	2,9563E-47	8,8959E-27	1,7359E-15	1	1,3881E-05	5,5291E-14	1,2214E-06
10	5,8848E-31	2,8237E-19	1,2797E-13	4,6222E-03	1,3402E-21	2,8513E-65	8,0844E-18	1,0221E-11	1,3881E-05	1	7,4468E-15	4,0974E-17
11	3,9294E-80	8,0492E-54	5,1314E-27	2,6916E-19	4,9925E-18	9,4083E-27	2,9101E-61	9,6552E-46	5,5291E-14	7,4468E-15	1	8,2812E-14
12	5,7189E-80	1,4815E-64	5,1046E-11	9,8657E-30	1,4843E-38	1,1431E-23	4,1041E-54	1,3418E-38	1,2214E-06	4,0974E-17	8,2812E-14	1

4. Sequential learning

Pada proses pembelajaran algoritma *sequential* sesuai penelitian Vijayakumar, langkah pertama yang dilakukan adalah melakukan perhitungan matriks *Hessian* menggunakan nilai λ yang telah ditentukan sebelumnya, dengan rumus yang ditunjukkan pada Persamaan (2.21).

Untuk perhitungan kolom pertama kolom pertama:

$$D_{1,1} = y_1 y_1 \left(K(x_1, x_1) + \lambda^2 \right)$$
$$= 1 * 1 \left(1 + 0{,}2^2 \right)$$
$$= 1{,}04$$

Demikian seterusnya sehingga didapatkan tabel matriks *Hessian* yang ditunjukkan pada Tabel 3.17.

Tabel 3.17 tabel matriks *Hessian*

	1	2	3	4	5	6	7	8	9	10	11	12
1	1,04	0,04	-0,04	-0,04	-0,04	-0,04	0,04	0,04	-0,04	-0,04	-0,04	-0,04
2	0,04	1,04	-0,04	-0,04	-0,04	-0,04	0,04	0,04	-0,04	-0,04	-0,04	-0,04
3	-0,04	-0,04	1,04	0,04	0,04	0,04	-0,04	-0,04	0,04	0,04	0,04	0,04
4	-0,04	-0,04	0,04	1,04	0,04	0,04	-0,04	-0,04	0,04	0,04	0,04	0,04
5	-0,04	-0,04	0,04	0,04	1,04	0,04	-0,04	-0,04	0,04	0,04	0,04	0,04
6	-0,04	-0,04	0,04	0,04	0,04	1,04	-0,04	-0,04	0,04	0,04	0,04	0,04
7	0,04	0,04	-0,04	-0,04	-0,04	-0,04	1,04	0,04	-0,04	-0,04	-0,04	-0,04
8	0,04	0,04	-0,04	-0,04	-0,04	-0,04	0,04	1,04	-0,04	-0,04	-0,04	-0,04
9	-0,04	-0,04	0,04	0,04	0,04	0,04	-0,04	-0,04	1,04	0,04	0,04	0,04
10	-0,04	-0,04	0,04	0,04	0,04	0,04	-0,04	-0,04	0,04	1,04	0,04	0,04
11	-0,04	-0,04	0,04	0,04	0,04	0,04	-0,04	-0,04	0,04	0,04	1,04	0,04
12	-0,04	-0,04	0,04	0,04	0,04	0,04	-0,04	-0,04	0,04	0,04	0,04	1,04

Langkah kedua, inisialisasi nilai α awal yaitu sama dengan 0. Lalu dilakukan iterasi menghitung nilai E_i, $\delta\alpha_i$, dan memperbarui α_i hingga iterasi yang diinginkan, untuk mendapatkan α terbaru dari tiap data latih. Iterasi dalam studi kasus ini menggunakan 5 kali proses pembaharuan α. Perhitungan nilai E_i dapat dinyatakan menggunakan Persamaan (2.22). Untuk perhitungan nilai E ke 1 pada iterasi 1:

$$E_1 = \sum_{j=1}^{12} \alpha_j D_{1j}$$
$$= (0 * 1{,}04) + (0 * 0{,}04) + (0 * -0{,}04) + (0 * -0{,}04) + (0 * -0{,}04) + (0 * -0{,}04)$$
$$+ (0 * 0{,}04) + (0 * 0{,}04) + (0 * -0{,}04) + (0 * -0{,}04) + (0 * -0{,}04) + (0 * -0{,}04)$$
$$= 0$$

Tabel hasil perhitungan nilai *E* ditunjukkan pada Tabel 3.18.

Tabel 3.18 Tabel nilai *E*

E	iterasi 1	iterasi 2	iterasi 3	iterasi 4	iterasi 5
1	0	0,084027209	0,162018413	0,234226536	0,300931167
2	0	0,084003625	0,161975595	0,234168255	0,300860678
3	0	0,116101205	0,218211527	0,308082314	0,387237635
4	0	0,116462215	0,218855365	0,308943704	0,388262276
5	0	0,116	0,218030988	0,307840708	0,386950164
6	0	0,116	0,218030988	0,307840708	0,386950164
7	0	0,084434758	0,162758364	0,235233742	0,302149377
8	0	0,084411157	0,162715495	0,235175364	0,302078739
9	0	0,116102715	0,218214221	0,308085918	0,387241923
10	0	0,116463603	0,218857842	0,308947019	0,388266222
11	0	0,116	0,218030988	0,307840708	0,386950164
12	0	0,116000122	0,218031205	0,307840999	0,386950511

Setelah diketahui keseluruhan E_i, maka dapat dicari $\delta\alpha_i$ dengan nilai *γ* dan *C* yang sudah ditentukan, dengan rumus yang telah ditunjukkan pada Persamaan (2.23). Untuk perhitungan nilai $\delta\alpha_i$ ke 1 pada iterasi ke 1:

$$\delta\alpha_1 = \min\{\max[\gamma(1 - E_1), -\alpha_1], C - \alpha_1\}$$
$$= \min\{\max[0{,}1(1 - 0), -0], 713{,}35 - 0\}$$
$$= \min\{0{,}1,\ 713{,}35\}$$
$$= 0{,}1$$

Demikian seterusnya sehingga didapatkan sehingga didapatkan tabel nilai *δαi* yang ditunjukkan pada tabel 3.19.

Tabel 3.19 Tabel nilai *δαi*

δαi	iterasi 1	iterasi 2	iterasi 3	iterasi 4	iterasi 5
1	0,1	0,091597279	0,083798159	0,076577346	0,069906883
2	0,1	0,091599637	0,08380244	0,076583175	0,069913932
3	0,1	0,088389879	0,078178847	0,069191769	0,061276237
4	0,1	0,088353778	0,078114463	0,06910563	0,061173772
5	0,1	0,0884	0,078196901	0,069215929	0,061304984
6	0,1	0,0884	0,078196901	0,069215929	0,061304984
7	0,1	0,091556524	0,083724164	0,076476626	0,069785062
8	0,1	0,091558884	0,083728451	0,076482464	0,069792126

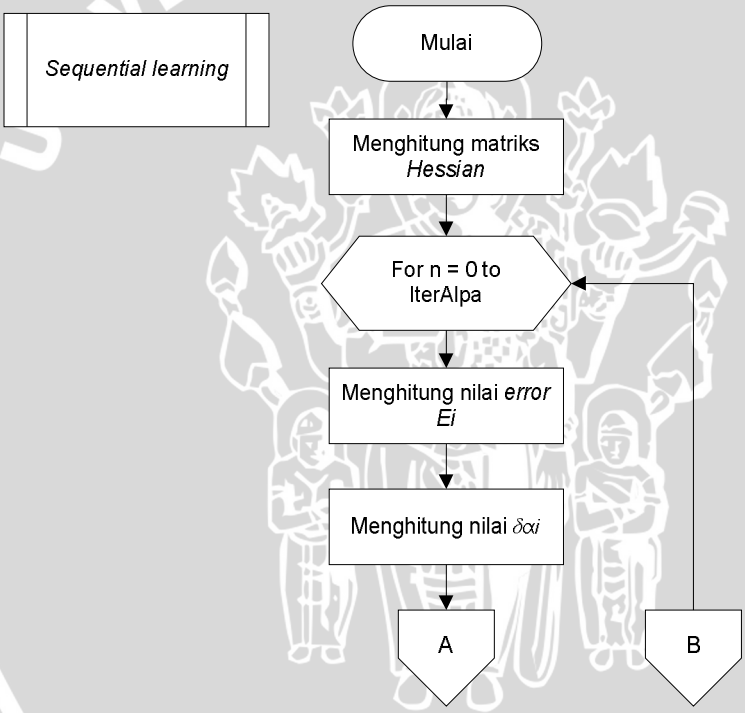
Tabel 3.19 Tabel nilai *δαi*

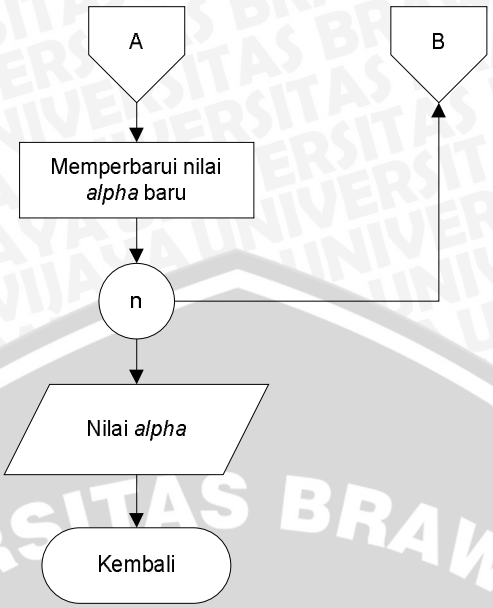
δαi	iterasi 1	iterasi 2	iterasi 3	iterasi 4	iterasi 5
9	0,1	0,088389728	0,078178578	0,069191408	0,061275808
10	0,1	0,08835364	0,078114216	0,069105298	0,061173378
11	0,1	0,0884	0,078196901	0,069215929	0,061304984
12	0,1	0,088399988	0,078196879	0,0692159	0,061304949

Sedangkan untuk memperbaharui nilai α dihitung dengan menggunakan Persamaan (2.24). Untuk perhitungan nilai α ke 1 pada iterasi 1:

$$\alpha_1 = \alpha_1 + \delta\alpha_1$$
$$= 0 + 0{,}1 = 0{,}1$$

Proses *sequential learning* yang telah dijabarkan diatas dapat diilustrasikan pada Gambar 3.7.





Gambar 3.7 Diagram Alir Proses *Sequential Learning*

Berikut Tabel 3.20 menunjukkan pembaruan α dengan iterasi sebanyak 5 pada proses *multi class* level 1 menggunakan nilai λ = 0,2 untuk perhitungan matriks *Hessian*, serta nilai γ = 0,1 dan nilai C = 713,35 untuk proses pembaruan α.

Tabel 3.20 Tabel Pembaruan *Alpha*

αi	iterasi 1	iterasi 2	iterasi 3	iterasi 4	iterasi 5
1	0,1	0,191597279	0,275395438	0,351972784	0,421879667
2	0,1	0,191599637	0,275402078	0,351985252	0,421899185
3	0,1	0,188389879	0,266568727	0,335760495	0,397036732
4	0,1	0,188353778	0,266468242	0,335573872	0,396747644
5	0,1	0,1884	0,266596901	0,33581283	0,397117814
6	0,1	0,1884	0,266596901	0,33581283	0,397117814
7	0,1	0,191556524	0,275280688	0,351757314	0,421542376
8	0,1	0,191558884	0,275287335	0,351769798	0,421561925
9	0,1	0,188389728	0,266568306	0,335759715	0,397035522
10	0,1	0,18835364	0,266467855	0,335573154	0,396746531
11	0,1	0,1884	0,266596901	0,33581283	0,397117814
12	0,1	0,188399988	0,266596867	0,335812767	0,397117716

5. Menghitung nilai bias *b*

Setelah dilakukan iterasi pembaruan nilai α sebanyak 5 kali, kemudian dicari nilai α terbesar dari masing-masing kelas positif dan negatif untuk digunakan saat penghitungan nilai bias. Pada studi kasus ini diketahui nilai α terbesar pada

kelas positif adalah nilai α ke 2 dan nilai α terbesar pada kelas negatif adalah nilai α ke 5. Menghitung nilai bias dilakukan dengan Persamaan (3.7).

$$b = -\frac{1}{2}\sum_{i=1}^{n}\alpha_i y_i\left(K\left(x_i, x^+\right) + K\left(x_i, x^-\right)\right) \tag{3.7}$$

Dimana $K\left(x_i, x^+\right)$ adalah nilai nilai kernel dari data kelas positif ke-*i* yang memiliki nilai α terbesar pada kelas positif dan $K\left(x_i, x^-\right)$ adalah nilai nilai kernel dari data kelas positif ke-*i* yang memiliki nilai α terbesar pada kelas negatif. Untuk perhitungan nilai bias pada level 1:

$$b = -\frac{1}{2}\sum_{i=1}^{12}\alpha_i y_i\left(K\left(x_i, x^+\right) + K\left(x_i, x^-\right)\right)$$

$$= -\frac{1}{2}\begin{pmatrix} (0{,}422*1*(3{,}614E-05+2{,}041E-68))+(0{,}422*1*(1+9{,}2954\text{E-}45)) \\ +(0{,}397*-1*(1{,}481E-64+1{,}484E-38))+(0{,}397*\text{-}1*(1{,}6878\text{E-}10+1{,}9908\text{E-}20)) \\ +(0{,}397*\text{-}1*(9{,}2954\text{E-}45+1))+(0{,}397*\text{-}1*(2{,}7847\text{E-}144+5{,}2925\text{E-}65)) \\ +(0{,}421*1*(9{,}3497\text{E-}08+1{,}0213\text{E-}55))+(0{,}421*1*(1{,}3132\text{E-}08+1{,}2336\text{E-}43)) \\ +(0{,}397*\text{-}1*(2{,}5768\text{E-}32+3{,}4811\text{E-}28))+(0{,}397*\text{-}1*(2{,}8237\text{E-}19+1{,}3402\text{E-}21)) \\ +(0{,}397*\text{-}1*(8{,}0492\text{E-}54+4{,}9925\text{E-}18))+(0{,}397*\text{-}1*(1{,}4815\text{E-}64+1{,}4843\text{E-}38)) \end{pmatrix}$$

$$= -0{,}01239833$$

6. Menghitung nilai fungsi *f(x)*

Setelah didapat nilai bias *b* kemudian mencari *hyperplane* dengan Persamaan (3.8) berikut.

$$f(\mathrm{x}) = \mathrm{sign}\left(\sum_{\mathrm{i}=1}^{\mathrm{n}}\alpha_i y_i K\left(x, x_i\right) + b\right) \tag{3.8}$$

Dengan menggunakan Persamaan (3.8) akan dihasilkan nilai yang nantinya digunakan untuk menentukan status resiko stroke dari data uji yang digunakan. Data uji yang digunakan dalam studi kasus ini ditunjukkan pada Tabel 3.21.

Tabel 3.21 Tabel Data Uji yang Digunakan

No	Umur	Kolestrol Total	HDL	LDL	Trigliserida	Kelas
5	40	253	46,5	179,3	136	1
6	48	197	30,5	120,5	79	1
123	35	197	39,6	119,6	189	2
124	52	157	30,6	92,6	169	2
160	67	276	51,4	188,2	182	3
161	79	340	31,2	208,9	162	3

Pertama menghitung nilai kernel dari data uji terhadap masing-masing data latih yang digunakan dengan menggunakan Persamaan (3.6). Contoh perhitungan nilai kernel untuk data uji ke 1 terhadap lata latih ke 1:

$$K(1,1) = \exp\left(\frac{-\| x-y \|^2}{2\sigma^2}\right)$$

$$= \exp\left(-\frac{\begin{array}{l}(40-44)^2+(253-174)^2+(46{,}5-36{,}5)^2+(179{,}3-123{,}3)^2 \\ +(136-71)^2\end{array}}{2*(9{,}8294)^2}\right)$$

$$= 1{,}47487\text{E-}31$$

Hasil perhitungan kernel data uji terhadap masing-masing data latih ditunjukkan pada Tabel 3.22.

Tabel 3.22 Nilai Kernel Data Uji

K(x,y)	UJI 1	UJI 2	UJI 3	UJI 4	UJI 5	UJI 6
1	1,47487E-31	0,034098877	1,91612E-33	2,6615E-25	5,90323E-63	1,4847E-100
2	5,67142E-30	2,62615E-05	3,86705E-18	8,54004E-10	2,11701E-52	2,40036E-95
3	0,004108149	1,30785E-33	3,29688E-20	1,4944E-41	0,002685569	1,09636E-19
4	1,30855E-20	2,51359E-19	0,078587275	4,21925E-05	2,13152E-30	6,27989E-73
5	8,11143E-49	1,20167E-56	4,53484E-17	3,98541E-26	1,10276E-44	2,80039E-81
6	2,80988E-79	9,2913E-159	3,36786E-67	9,6265E-110	8,69151E-45	1,25644E-66
7	6,36595E-17	0,002076731	1,8695E-22	1,11618E-23	2,0736E-43	7,49394E-77
8	1,50106E-08	1,16669E-05	4,26359E-16	9,74973E-22	9,12913E-27	3,93288E-56
9	4,49275E-07	6,70989E-35	1,4044E-09	4,59291E-29	2,06296E-06	1,05548E-31
10	3,15681E-14	2,15469E-25	0,049839303	5,62321E-13	2,29116E-21	1,15095E-58
11	1,80234E-38	4,71297E-70	2,98713E-12	1,06191E-29	7,2724E-26	1,37059E-64
12	1,17759E-18	1,30392E-67	9,78097E-23	2,36835E-53	6,71568E-09	1,28808E-29

Kemudian menghitung nilai fungsi *f(x)* dengan Persamaan (3.8). Contoh perhitungan untuk nilai fungsi *f(x)* pada data uji ke 1:

$$f(x_5) = sign\left(\sum_{i=1}^{12} \alpha_i y_i K(x_i, x_j) + b\right)$$

$$= \left(\begin{array}{l} \left(\begin{array}{l} (0{,}4227 * 1 * 1{,}47487\text{E-}31) + (0{,}422 * 1 * 5{,}67142\text{E-}30) \\ + (0{,}397 * -1 * 0{,}004108149) + (0{,}397 * -1 * 1{,}30855\text{E-}20) \\ + (0{,}397 * -1 * 8{,}11143\text{E-}49) + (0{,}397 * -1 * 2{,}80988\text{E-}79) \\ + (0{,}421 * 1 * 6{,}36595\text{E-}17) + (0{,}421 * 1 * 1{,}50106\text{E-}08) \\ + (0{,}397 * -1 * 4{,}49275\text{E-}07) + (0{,}397 * -1 * 3{,}15681\text{E-}14) \\ + (0{,}397 * -1 * 1{,}80234\text{E-}38) + (0{,}397 * -1 * 1{,}17759\text{E-}18) \end{array} \right) \\ + - 0{,}01239833 \end{array} \right)$$

$$= sign(-0{,}01402959)$$

$$= -1$$

Hasil dari perhitungan nilai fungsi *f(x)* level 1 dari 6 data uji ditunjukkan pada Tabel 3.23.

Tabel 3.23 Tabel Hasil Perhitungan Nilai Fungsi *f(x) Level* 1

No Data Uji	$\Sigma\alpha_i y_i K(x,x_i)+b$	*f(x)*
5	-0,01402959	-1
6	0,002878719	1
123	-0,063351219	-1
124	-0,012415071	-1
160	-0,013465423	-1
161	-0,012398332	-1

7. Kembali ke langkah 2 untuk perhitungan level 2 O*ne-Against-All*

Setelah didapat nilai fungsi *f(x)* untuk perhitungan level 1 maka selanjutnya akan dilakukan perhitungan untuk level 2. Untuk level 2 data dengan kelas 1 tidak lagi digunakan sehingga data latih yang digunakan untuk pelatihan SVM sebanyak 8 data, dimana data dengan kelas 2 (rentan) sebagai kelas positif dan data dengan kelas 3 (mengkhawatirkan) sebagai kelas negatif. Tabel 3.24 menunjukkan pembagian kelas positif dan negatif untuk level 2.

Tabel 3.24 Tabel Pembagian Kelas Positif dan Negatif *Level* 2

Data Latih Nomor	Kelas	*y*
119	2	1
120	2	1
156	3	-1
157	3	-1
121	2	1
122	2	1

Tabel 3.24 Tabel Pembagian Kelas Positif dan Negatif *Level* 2

Data Latih Nomor	Kelas	*y*
158	3	-1
159	3	-1

Kemudian menghitung kernel RBF dengan 8 data latih yang akan menghasilkan matriks kernel 8x8, dilanjutkan dengan melakukan *sequential learning*, menghitung bias, dan menghitung nilai fungsi *f(x)* untuk level 2.

Untuk hasil dari fungsi *f(x)* level 2 pada 6 data uji yang sama ditunjukkan pada Tabel 3.25.

Tabel 3.25 Tabel Hasil Perhitungan Nilai Fungsi *f(x) Level* 2

No Data Uji	$\Sigma\alpha_i y_i K(x,x_i) + b$	*f(x)*
5	0,001512222	1
6	-0,000169967	-1
123	0,052374101	1
124	-0,000152704	-1
160	0,000930431	1
161	-0,000169967	-1

Setelah diketahui hasil dari dua fungsi, kemudian hasil nya digabungkan untuk dilakukan prediksi kelas dari status resiko penyakit stroke yaitu 1 (normal), 2 (rentan), dan 3 (mengkhawatirkan). Hasil dari penggabungan dua fungsi ditunjukkan pada Tabel 3.26.

Tabel 3.26 Tabel Hasil Penggabungan Dua Fungsi

No Data Uji	*f(x)* level 1	*f(x)* level 2	Hasil
5	-1	1	2 (rentan)
6	1	-1	1 (normal)
123	-1	1	2 (rentan)
124	-1	-1	3 (mengkhawatirkan)
160	-1	1	2 (rentan)
161	-1	-1	3 (mengkhawatirkan)

8. Mencari nilai akurasi

Setelah didapatkan hasil dari klasifikasi kemudian dihitung nilai akurasi dari sistem. Mencari nilai akurasi dilakukan dengan melakukan perbandingan antara

hasil prediksi data uji dari sistem dengan hasil prediksi dokter. Hasil dari perhitungan nilai akurasi ditunjukkan pada Tabel 3.27.

Tabel 3.27 Tabel Perbandingan Hasil Sistem dengan Hasil Aktual

No Data Uji	Hasil Sistem	Hasil Dokter
5	2 (rentan)	1 (normal)
6	1 (normal)	1 (normal)
123	2 (rentan)	2 (rentan)
124	3 (mengkawatirkan)	2 (rentan)
160	2 (rentan)	3 (mengkhawatirkan)
161	3 (mengkhawatirkan)	3 (mengkhawatirkan)

Dari tabel diatas dapat dihitung nilai akurasi dari data uji dari *fold* 3 yang digunakan dengan menggunakan Persamaan (3.2). Perhitungan nilai akurasi untuk fold 3:

$$Acc_{fold3} = \frac{Benar}{Benar + Salah}$$
$$= \frac{3}{3+3}$$
$$= \frac{3}{6}$$
$$Acc_{fold3} = 0{,}5$$

Nilai akurasi ini nantinya akan digunakan untuk menghitung akurasi rata-rata yang pada akhirnya akan digunakan oleh ACO untuk dijadikan acuan untuk melakukan *update pheromone*. Pada studi kasus ini untuk fold 1 didapatkan nilai akurasi sebesar 0,66667, fold 2 sebesar 0,66667, dan fold 3 sebesar 0,5. Untuk menghitung akurasi rata-rata digunakan Persamaan (3.3). Pada studi kasus ini digunakan parameter yang didapat dari hasil pencarian semut 2, maka perhitungan akurasi rata-rata untuk semut 2:

$$MeanAcc_{semut2} = \frac{\sum_{i=1}^{K} Acc_i}{jumlahfold}$$
$$= \frac{0{,}66667 + 0{,}66667 + 0{,}5}{3}$$
$$= 0{,}6111111$$

BAB 4 PERANCANGAN

Pada bab ini akan dibahas mengenai perancangan basis data, perancangan antarmuka, dan perancangan skenario pengujian.

4.1 Perancangan Basis Data

Basis data yang digunakan memiliki sebuah tabel yang digunakan yaitu tabel dataset. Tabel dataset memiliki beberapa kolom yaitu kolom No, Umur, KolesterolTotal, HDL, LDL, Trigliserida, dan Kelas. Desain tabel *dataset* untuk menyimpan data dapat dilihat pada Tabel 4.1.

Tabel 4.1 Tabel *Dataset*

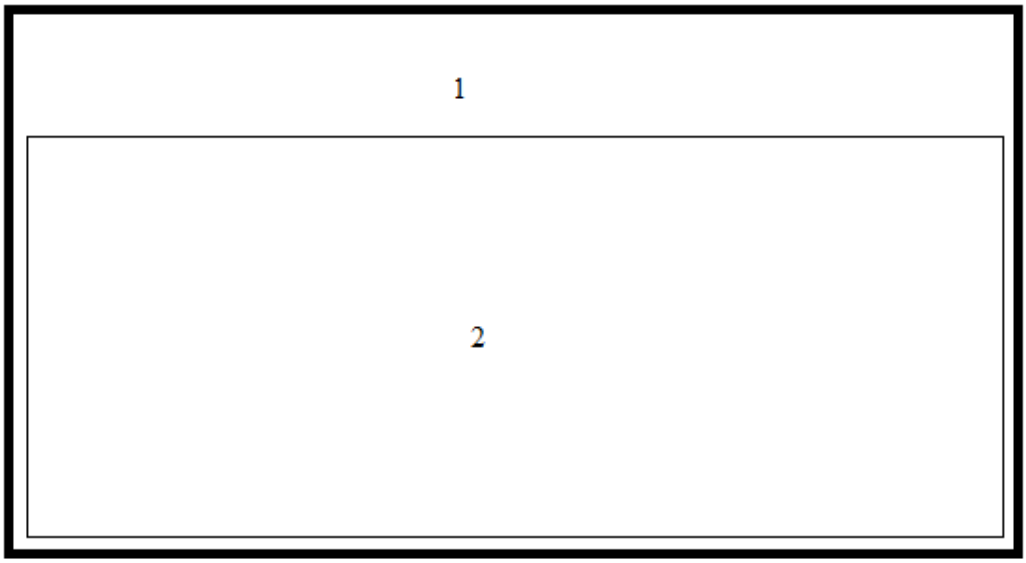
No	Kolom	Tipe
1	No	*int*(11)
2	Umur	*int*(11)
3	KolesterolTotal	*int*(11)
4	HDL	*float*
5	LDL	*float*
6	Trigliserida	*int*(11)
7	Kelompok	*int*(20)

4.2 Perancangan Antarmuka

Pada perancangan antarmuka sistem ini terdiri dari 4 halaman tab. Halaman tab pertama adalah halaman yang berisi daftar data pada tabel dataset yang akan digunakan dalam sistem. Halaman tab kedua adalah halaman masukan parameter algoritma *Ant Colony Optimization* (ACO) dan parameter algoritma *Support Vector Machine* (SVM). Halaman tab ketiga adalah hasil dari perhitungan algoritma ACO dan hasil akurasi. Halaman tab keempat berisi hasil perhitungan SVM dan hasil klasifikasi dari data uji.

4.2.1 Perancangan Antarmuka Halaman Data

Rancangan halaman data adalah halaman yang berisi keseluruhan data yang digunakan pada sistem ini. Rancangan halaman data ditampilkan pada Gambar 4.1.



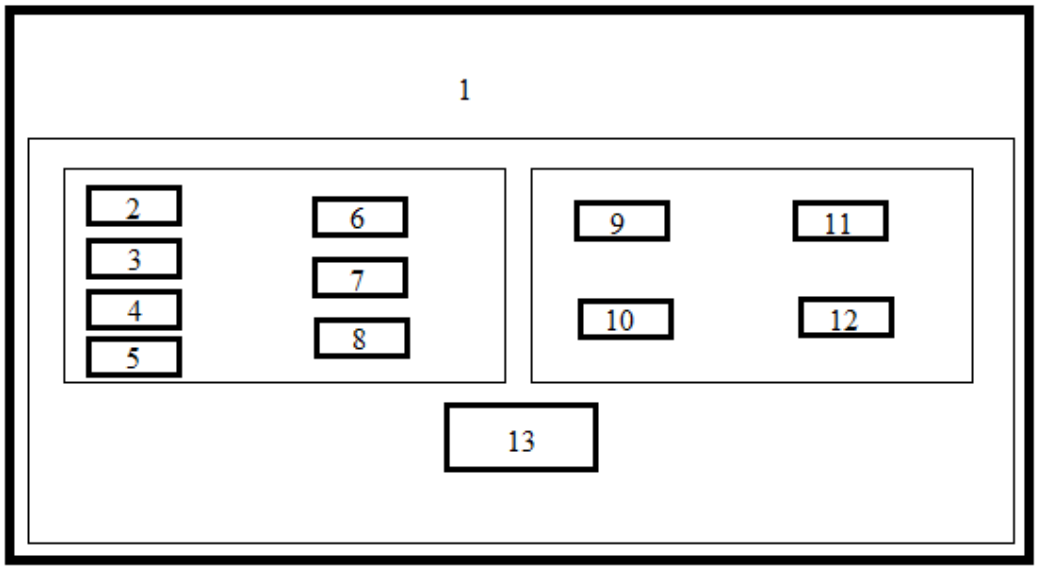
Gambar 4.1 Halaman Data

Keterangan :

1. Logo.
2. Tabel data yang diambil dari *database*.

4.2.2 Perancangan Antarmuka Halaman Parameter

Rancangan halaman parameter adalah halaman yang berisi *form* untuk memasukkan nilai-nilai parameter algoritma. Rancangan halaman parameter dan hasil akurasi ditampilkan pada Gambar 4.2.



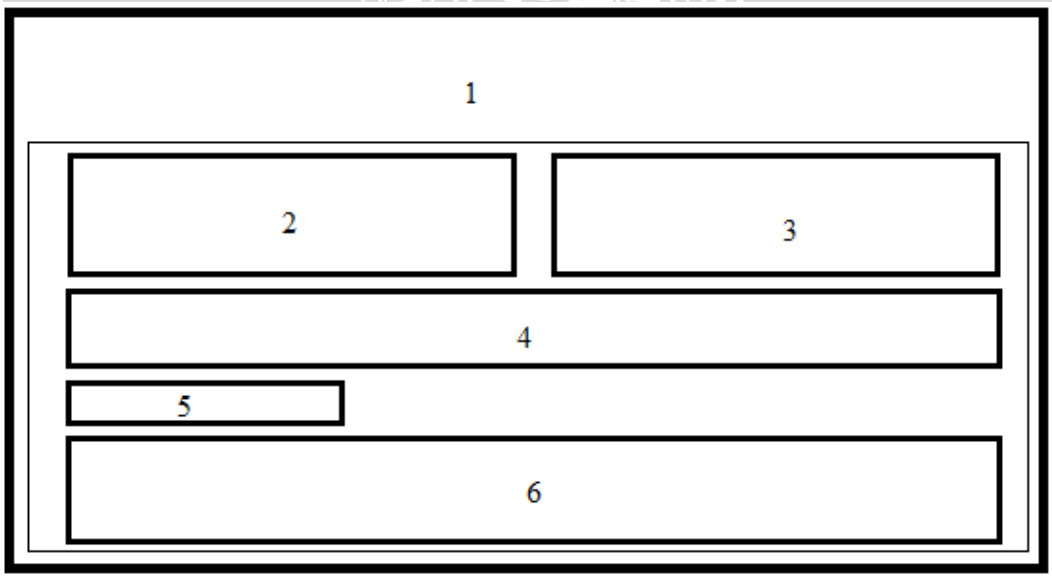
Gambar 4.2 Halaman Parameter dan Hasil Akurasi

Keterangan :

1. Judul.
2. *Textbox* untuk nilai iterasi maksimal ACO.
3. *Textbox* untuk nilai jumlah semut.
4. *Textbox* untuk nilai *pheromone* awal.
5. *Textbox* untuk nilai *K* untuk *K-fold cross validation*.
6. *Textbox* untuk nilai parameter α.
7. *Textbox* untuk nilai parameter β.
8. *Textbox* untuk nilai parameter η.
9. *Textbox* untuk nilai *alpha* ke 0.
10. *Textbox* untuk nilai parameter λ.
11. *Textbox* untuk nilai parameter γ.
12. *Textbox* untuk jumlah iterasi pelatihan *alpha*.
13. *Button* untuk melakukan proses perhitungan.

4.2.3 Perancangan Antarmuka Halaman Hasil Perhitungan ACO

Rancangan halaman hasil perhitungan ACO adalah halaman yang menampilkan hasil dari semua semut dalam menentukan nilai parameter SVM dan hasil akurasi rata-rata *K-fold cross validation*. Halaman ini juga menampilkan tabel *pheromone* yang telah diperbarui. Rancangan halaman hasil perhitungan ACO ditampilkan pada Gambar 4.3.



Gambar 4.3 Halaman Tabel Proses SVM

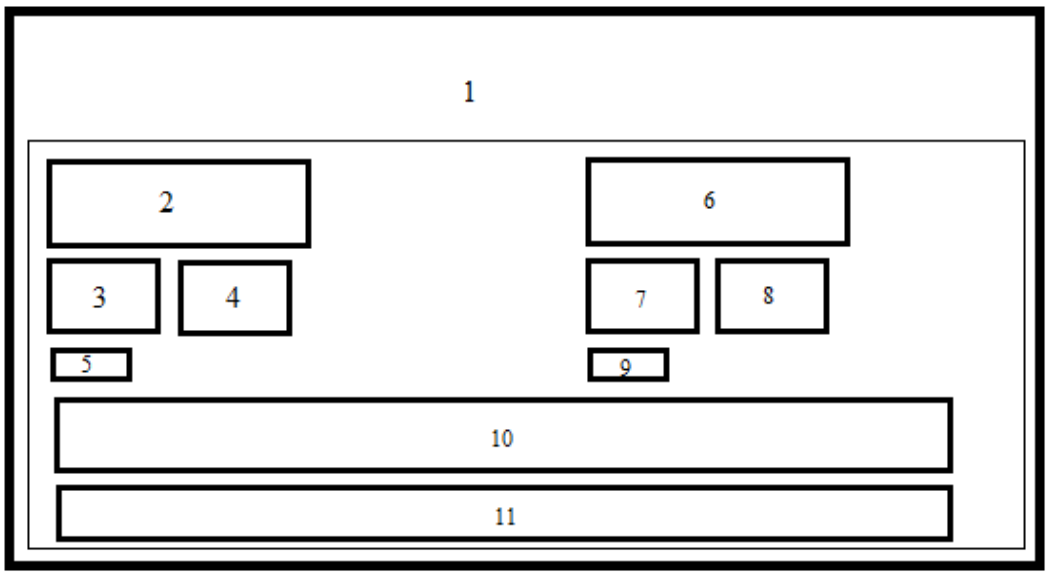
Keterangan :

1. Judul.

2. *Datagridview* untuk jalur yang telah dilalui semua semut.
3. *Datagridview* untuk menampilkan jalur dari semua semut yang telah dibagi.
4. *Datagridview* untuk menampilkan hasil akurasi dan akurasi rata-rata dari *K-fold cross validation*.
5. *Datagridview* untuk menampilkan semut dengan nilai parameternya yang memiliki akurasi rata-rata terbaik.
6. *Datagridview* untuk menampilkan tabel *pheromone* yang telah diperbarui.

4.2.4 Perancangan Antarmuka Halaman Hasil Klasifikasi SVM

Rancangan halaman hasil klasifikasi SVM adalah halaman yang berisi *datagridview* dan *textbox* yang menampilkan hasil perhitungan untuk proses *One-Against All* level 1 pada sisi kiri. Sedangkan sisi kanan adalah perhitungan untuk proses level 2. Juga ditampilkan hasil klasifikasi sistem. Rancangan halaman hasil klasifikasi dan *form* klasifikasi ditampilkan pada Gambar 4.4.



Gambar 4.4 Halaman Tabel Proses SVM

Keterangan :

1. Judul.
2. *Datagridview* untuk menampilkan data latih level 1.
3. *Datagridview* untuk menampilkan nilai *error Ei* level 1.
4. *Datagridview* untuk menampilkan nilai *alpha* level 1.
5. *Textbox* untuk menampilkan nilai bias *b* level 1.
6. *Datagridview* untuk menampilkan data latih level 2.
7. *Datagridview* untuk menampilkan nilai *error Ei* level 1.
8. *Datagridview* untuk menampilkan nilai *alpha* level 1.

9. *Textbox* untuk menampilkan nilai bias *b* level 2.
10. *Datagridview* untuk menampilkan data uji.
11. *Datagridview* untuk menampilkan hasil perhitungan fungsi *f(x).*

4.3 Perancangan Skenario Pengujian

Masih belum adanya cara untuk menentukan parameter terbaik dari metode SVM-ACO sehingga diharuskan melakukan untuk mendapatkan nilai parameter yang optimal. Uji coba tersebut antara lain:

1. Pengujian untuk menentukan batas nilai parameter yang optimal.
2. Pengujian untuk menentukan nilai parameter SVM yang optimal.
3. Pengujian untuk menentukan jumlah semut yang optimal.
4. Pengujian untuk menentukan jumlah iterasi ACO yang optimal.
5. Pengujian untuk menentukan panjang *fold* yang optimal.

4.3.1 Perancangan Skenario Pengujian Batas Nilai Parameter

Pengujian batas nilai parameter yaitu uji coba yang dilakukan untuk mengetahui berapa batasan yang diperlukan untuk menghasilkan nilai parameter yang optimal. Batasan ini nantinya akan mempengaruhi nilai parameter SVM yang akan digunakan.

1. Batas nilai parameter *C*

Rancangan pengujian batas nilai parameter *C* dapat dilihat pada Tabel 4.2.

Tabel 4.2 Rancangan Pengujian Batas Nilai Parameter *C*

Batas nilai parameter *C*	Percobaan Ke-*i*										Akurasi Rata-rata
	1	2	3	4	5	6	7	8	9	10	
0,0000 – 9,9999											
10,000 – 99,999											
100,00 – 999,99											
1000,0 – 9999,9											
10000 - 99999											

2. Batas nilai parameter *σ*

Rancangan pengujian batas nilai parameter *σ* dapat dilihat pada Tabel 4.3.

Tabel 4.3 Rancangan Pengujian Batas Nilai Parameter *σ*

Batas nilai parameter *σ*	Percobaan Ke-*i*										Akurasi Rata-rata
	1	2	3	4	5	6	7	8	9	10	
0,0000 – 9,9999											
10,000 – 99,999											
100,00 – 999,99											

Tabel 4.3 Rancangan Pengujian Batas Nilai Parameter σ

Batas nilai parameter σ	Percobaan Ke-*i*										Akurasi Rata-rata
	1	2	3	4	5	6	7	8	9	10	
1000,0 – 9999,9											
10000 - 99999											

4.3.2 Perancangan Skenario Pengujian Nilai Parameter SVM

Pengujian nilai parameter SVM yaitu uji coba yang digunakan untuk mengetahui nilai parameter λ dan γ untuk mendapatkan akurasi rata-rata terbaik.

1. Nilai parameter λ

Rancangan pengujian nilai parameter λ dapat dilihat pada Tabel 4.4.

Tabel 4.4 Rancangan Pengujian Nilai Parameter λ

Nilai λ	Percobaan Ke-*i*										Akurasi Rata-rata
	1	2	3	4	5	6	7	8	9	10	
0,1											
0,2											
0,3											
0,4											
0,5											
0,6											
0,7											
0,8											
0,9											
1											

2. Nilai parameter γ

Rancangan pengujian nilai parameter γ dapat dilihat pada Tabel 4.5.

Tabel 4.5 Rancangan Pengujian Nilai Parameter γ

Nilai γ	Percobaan Ke-*i*										Akurasi Rata-rata
	1	2	3	4	5	6	7	8	9	10	
0,1											
0,2											
0,3											
0,4											
0,5											
0,6											
0,7											
0,8											
0,9											

Tabel 4.5 Rancangan Pengujian Nilai Parameter *γ*

Nilai *γ*	Percobaan Ke-*i*										Akurasi Rata-rata
	1	2	3	4	5	6	7	8	9	10	
1											

4.3.3 Perancangan Skenario Pengujian Jumlah Semut

Pengujian jumlah semut yaitu uji coba yang digunakan untuk mengetahui banyaknya jumlah semut yang diperlukan untuk mendapatkan nilai parameter SVM yang optimal. Rancangan pengujian dapat dilihat pada Tabel 4.6.

Tabel 4.6 Rancangan Pengujian Jumlah Semut

Jumlah Semut	Percobaan Ke-*i*										Akurasi Rata-rata
	1	2	3	4	5	6	7	8	9	10	
5											
10											
15											
20											
30											
40											
50											
70											
80											
100											

4.3.4 Perancangan Skenario Pengujian Jumlah Iterasi ACO

Uji coba banyaknya jumlah iterasi ACO yaitu uji coba yang digunakan untuk mengetahui jumlah iterasi yang dibutuhkan untuk mendapatkan nilai parameter yang optimal. Rancangan pengujian jumlah iterasi ACO dapat dilihat pada Tabel 4.7.

Tabel 4.7 Rancangan Pengujian Jumlah Iterasi ACO

Jumlah Iterasi	Percobaan Ke-*i*										Akurasi Rata-rata
	1	2	3	4	5	6	7	8	9	10	
5											
10											
15											
20											
25											

Tabel 4.7 Rancangan Pengujian Jumlah Iterasi ACO

Jumlah Iterasi	Percobaan Ke-*i*										Akurasi Rata-rata
	1	2	3	4	5	6	7	8	9	10	
40											
60											
80											
100											
200											

4.3.5 Perancangan Skenario Pengujian Panjang *fold*

Uji coba nilai banyaknya panjang *fold* yaitu uji coba yang digunakan untuk mengetahui jumlah *K* pada *K-fold cross validation* yang optimal agar nantinya dihasilkan hasil optimasi yang juga optimal. Rancangan pengujian dapat dilihat pada Tabel 4.8.

Tabel 4.8 Rancangan Pengujian Panjang *fold*

Panjang *fold*	Percobaan Ke-*i*										Akurasi Rata-rata
	1	2	3	4	5	6	7	8	9	10	
2											
3											
4											
5											
6											
7											
8											
9											
10											

BAB 5 IMPLEMENTASI

Bab ini akan membahas mengenai implementasi perangkat lunak berdasarkan hasil analisis kebutuhan dan perancangan yang dibuat. Hal-hal yang akan dibahas meliputi batasan-batasan implementasi, implementasi algoritma, dan implementasi antarmuka.

5.1 Batasan Implementasi

Beberapa batasan yang digunakan dalam mengimplementasikan metode SVM berbasis ACO untuk klasifikasi tingkat resiko penyakit stroke adalah sebagai berikut :

1. Sistem dibangun berdasarkan ruang lingkup *Desktop Application* dengan menggunakan bahasa pemrograman *C#*.
2. Data-data yang digunakan dalam sistem disimpan dalam *Database Management System* (DBMS) MySQL.
3. Metode yang digunakan dalam menyelesaikan masalah adalah Metode *Support Vector Machine* (SVM) dan *Ant Colony Optimization* (ACO).
4. Keluaran akhir yang diterima pengguna berdasarkan hasil akhir perhitungan dengan metode SVM berbasis ACO yang dilakukan sistem adalah nilai parameter yang optimal dan akurasi rata-rata terbaik dari nilai parameter.

5.2 Implementasi Algoritma

Sistem ini mempunyai 9 proses utama. Berikut akan dijelaskan implementasi masing-masing proses yang dikerjakan.

5.2.1 Implementasi Algoritma Perhitungan Kernel

Proses ini merupakan proses awal pada perhitungan menggukan metode SVM yang melibatkan *dot product* antar 2 data *training*. Fungsi kernel yang digunakan adalah kernel *RBF*. Keluaran yang dihasilkan berupa matriks kernel dengan indeks *n* x *n*, dan *n* merupakan banyaknya data *training*. Gambar 5.1 menunjukkan implementasi perhitungan matriks kernel.

```
1   for (level = 0; level < 2; level++)
2   {
3       lev = (level * -junor[numfold]) + (Data.Length -
4   banyaktes[numfold]);
5   for (i = 0; i < lev; i++)
6   {
7   for (j = 0; j < lev; j++)
8   {
9       jarak[i, j] = 0;
10      for (f = 1; f < 6; f++)
11
12  {
13          jarak[i, j] = jarak[i, j] +
14  Math.Pow(TrainData[level][numfold][i][f] -
15  TrainData[level][numfold][j][f], 2);
16  }
            kernel[level, numfold, i, j] = Math.Exp(jarak[i, j] / (-
```

```
2 * Math.Pow(res1[ant], 2)));
}
}
}
```

Gambar 5.1 Implementasi Algoritma Perhitungan Kernel

Penjelasan algoritma pada Gambar 5.1 adalah sebagai berikut:

1. Menghitung nilai jarak *euclidean* pada baris 12 dan 13.
2. Perhitungan nilai kernel pada baris 16 dan 17.

5.2.2 Implementasi Algoritma Perhitungan Matriks Hessian

Pada proses ini dilakukan merupakan langkah awal pada proses *Sequential Training* SVM. Perhitungan matriks *hessian* ini menggunakan data input berupa nilai dari hasil perhitungan menggunakan fungsi kernel dan nilai dari konstanta *lambda* (λ). Keluaran yang dihasilkan berupa matriks dengan indeks *n* x *n*, dimana *n* merupakan banyaknya data *training*. Gambar 5.2 menunjukkan implementasi perhitungan matriks *hessian*.

```
for (m = 0; m < 2; m++)
                        {
                            lev = (m * -junor[numfold]) +
(Data.Length - banyaktes[numfold]);
                            //MATRIKS D
                            for (i = 0; i < lev; i++)
                            {
                                for (j = 0; j < lev; j++)
                                {
                                    D[m, numfold, i, j] =
uy[m, numfold, i, m] * uy[m, numfold, j, m] * (kernel[m, numfold,
i, j] + Math.Pow(lambda, 2));
                                }
                            }
}
```

Gambar 5.2 Implementasi Algoritma Perhitungan Matriks *Hessian*

Penjelasan algoritma pada Gambar 5.2 adalah sebagai berikut:

1. Menghitung nilai matriks *hessian* menggunakan rumus yang telah ditentukan ditunjukkan pada baris ke-10 sampai baris ke-12.

5.2.3 Implementasi Algoritma Perhitungan Iterasi Pelatihan *α*

Dalam proses ini terdapat 3 tahap perhitungan yang diulang sejumlah *iterAlpa* (dimana *iterAlpa* adalah jumlah iterasi maksimum). Proses pertama pada iterasi ini yaitu melakukan perhitungan untuk mencari nilai *E* untuk tiap data *training*. Proses kedua yaitu menghitung nilai $\delta\alpha_i$ dan proses terakhir adalah mencari nilai α.

a. Implementasi Algoritma Perhitungan Nilai *Ei*

Pada proses ini perhitungan nilai *Ei* dilakukan dengan menjumlahkan hasil perkalian dari matriks hessian dengan *alpha* ke-j. Gambar 5.3 menunjukkan implementasi perhitungan nilai *Ei.*

```
1   for (m = 0; m < 2; m++)
2                       {
3                           lev = (m * -junor[numfold]) +
4   (Data.Length - banyaktes[numfold]);
5   for (n = 0; n < IterAlpa; n++)
6                               {
7                                   //E
8                                   for (i = 0; i < lev; i++)
9                                   {
10                                      E[ant, m, numfold, i] =
11  0;
12                                      for (j = 0; j < lev; j++)
13                                      {
14                                          E[ant, m, numfold, i]
15  = E[ant, m, numfold, i] + alphao[ant, m, numfold, j] * D[m,
16  numfold, i, j];
17                                      }
18                                  }
19  }
```

Gambar 5.3 Implementasi Algoritma Perhitungan Nilai *Ei*

Penjelasan algoritma pada Gambar 5.3 adalah sebagai berikut:

1. Inisialisasi awal nilai *Ei* = 0 pada baris 10.
2. Menghitung nilai *error* yang ditunjukkan pada baris 13 dan baris 14.

b. Implementasi Algoritma Perhitungan Nilai Delta Alpha

Proses perhitungan untuk mencari nilai *delta alpha* yaitu melakukan perkalian antara *gamma* dengan hasil dari 1 dikurangkan dengan nilai *Ei* ke-*i* dan dibandingkan dengan nilai *alpha* ke-*i* untuk dicari nilai paling maksimum. Hasil nilai maksimum dibandingkan dengan hasil dari variabel *C* dikurangkan dengan nilai *alpha* ke-*i* dan diambil nilai terkecil dari perbandingan tersebut. Gambar 5.4 menunjukkan implementasi perhitungan *delta alpha*.

```
1   for (m = 0; m < 2; m++)
2                       {
3                           lev = (m * -junor[numfold]) +
4   (Data.Length - banyaktes[numfold]);
5   for (n = 0; n < IterAlpa; n++)
6                               {
7
8   //Delta Alpha
9                                   for (i = 0; i < lev; i++)
10                                  {
11                                      DA[m, i] =
12  Math.Min(Math.Max(gamma * (1 - E[ant, m, numfold, i]), -
13  alphao[ant, m, numfold, i]), res[ant] - alphao[ant, m, numfold,
14  i]);
15                                  }
16  }
```

Gambar 5.4 Implementasi Algoritma Perhitungan Nilai *Delta Alpha*

Penjelasan algoritma pada Gambar 5.4 adalah sebagai berikut:

1. Menghitung nilai *delta alpha* ditunjukkan pada baris 11 sampai baris 13.

c. Implementasi Algoritma Perhitungan Nilai *Alpha* Baru

Perhitungan untuk menentukan nilai *alpha* baru yaitu dengan menambahkan nilai dari *delta alpha* ke-*i* dengan nilai *alpha* ke-*i* sebelumnya. Gambar 5.5 menunjukkan implementasi perhitungan nilai *alpha* baru.

```
1  for (m = 0; m < 2; m++)
2                          {
3                              lev = (m * -junor[numfold]) +
4  (Data.Length - banyaktes[numfold]);
5  for (n = 0; n < IterAlpa; n++)
6                                  {
7  for (i = 0; i < lev; i++)
8                                  {
9                                      alphao[ant, m, numfold, i] =
10 0;
11                                 }
12 for (i = 0; i < lev; i++)
13                                 {
14                                     alphao[ant, m, numfold,
15 i] = alphao[ant, m, numfold, i] + DA[m, i];
16                                 }
17 }
   }
```

Gambar 5.5 Implementasi Algoritma Perhitungan Nilai *Alpha* Baru

Penjelasan algoritma pada Gambar 5.5 adalah sebagai berikut:

1. Inisialisasi nilai *alpha* awal = 0 pada baris 9.
2. Menghitung nilai *alpha* baru ditunjukkan pada baris 13 dan 14.

5.2.4 Implementasi Algoritma Perhitungan Nilai Bias *b*

Dalam proses perhitungan untuk mencari nilai bias membutuhkan total jumlah bobot yang di-*dot product* dengan data pada kelas positif dan total jumlah bobot yang di-*dot product* dengan data pada kelas negatif. Nilai bobot merupakan hasil dari perkalian dari *alpha*, nilai kernel dari data dengan kelas positif/negatif, dan kelas dari data negatif atau positif seperti yang telah dijlaskan sebelumnya. Kemudian bobot yang telah di-*dot product* dengan data pada kelas positif dan data pada kelas negatif dijumlahkan dan hasilnya dikalikan dengan -0.5. Gambar 5.6 menunjukkan implementasi perhitungan nilai bias *b.*

```
1  for (m = 0; m < 2; m++)
2                          {
3                              lev = (m * -junor[numfold]) +
4  (Data.Length - banyaktes[numfold]);
5  //Pencarian Alpha Terbesar
6                                  mpositif = 0; mnegatif = 0; mn =
7  0; mp = 0;
8                                  for (i = 0; i < lev; i++)
9                                  {
10                                     if (uy[m, numfold, i, m] ==
11 1)
12                                     {
13                                         if (mpositif <=
   alphao[ant, m, numfold, i])
                                           {
                                               mpositif =
```

```
14  alphao[ant, m, numfold, i];
15                                          mp = i;
16                                      }
17                                  }
18                                  if (uy[m, numfold, i, m] == -
19  1)
20                                  {
21                                      if (mnegatif <=
22  alphao[ant, m, numfold, i])
23                                      {
24                                          mnegatif =
25  alphao[ant, m, numfold, i];
26                                          mn = i;
27                                      }
28                                  }
29                                  //b
30                                  sum[ant, numfold, m] = 0.0;
31                                  for (i = 0; i < lev; i++)
32                                  {
33                                      sum[ant, numfold, m] =
34  sum[ant, numfold, m] + alphao[ant, m, numfold, i] * uy[m,
35  numfold, i, m] * (kernel[m, numfold, mp, i] + kernel[m, numfold,
36  mn, i]);
37                                  }
                                    b[ant, numfold, m] = -0.5 *
    sum[ant, numfold, m];
    }
```

Gambar 5.6 Implementasi Algoritma Perhitungan Nilai Bias

Penjelasan algoritma pada Gambar 5.6 adalah sebagai berikut:

1. Mencari nilai *alpha* terbesar kelas positif dan negatif ditunjukkan pada baris 8 - 27.
2. Menghitung nilai bias ditunjukkan pada baris 35.

5.2.5 Implementasi Algoritma Perhitungan Nilai Fungsi *f(x)*

Pada proses ini perhitungan dilakukan terhadap data *testing* dengan cara menjumlahkan nilai dari total bobot data *testing* dengan nilai bias yang dihasilkan dari perhitungan sebelumnya. Hasil akhir adalah nilai yang dihasilkan oleh fungsi linier *f*(x), jika bernilai positif maka masuk kelas +1, jika bernilai negatif maka masuk kelas -1. Gambar 5.7 menunjukkan implementasi perhitungan nilai *f(x).*

```
1   //KERNEL UJI
2                           for (level = 0; level < 2; level++)
3                           {
4                               lev = (level * -junor[numfold]) +
5   (Data.Length - banyaktes[numfold]);
6                               for (i = 0; i < banyaktes[numfold];
7   i++)
8                               {
9                                   for (j = 0; j < lev; j++)
10                                  {
11                                      jarak[i, j] = 0;
12                                      for (f = 1; f < 6; f++)
```

```
                                    {
                                        jarak[i, j] = jarak[i, j]
+ Math.Pow(TesData[numfold][i][f] -
TrainData[level][numfold][j][f], 2);
                                    }
                                    kerneluji[level, numfold, i,
j] = Math.Exp(jarak[i, j] / (-2 * Math.Pow(res1[ant], 2)));
                                }
                            }}

//Fungsi
                            for (m = 0; m < 2; m++)
                            {
                                lev = (m * -junor[numfold]) +
(Data.Length - banyaktes[numfold]);
                                for (i = 0; i <
banyaktes[numfold]; i++)
                                {
                                    F[m, numfold, i] = 0;
                                }
                                for (i = 0; i < 6; i++)
                                {
                                        for (j = 0; j < lev; j++)
                                        {
                                            F[m, numfold, i] =
F[m, numfold, i] + (alphao[ant, m, numfold, j] * uy[m, numfold,
j, m] * kerneluji[m, numfold, i, j]);
                                        }
                                }
                            }
                            for (m = 0; m < 2; m++)
                            {
                                for (i = 0; i <
banyaktes[numfold]; i++)
                                {
                                    fx[ant, m, numfold, i] = F[m,
numfold, i] + b[ant, numfold, m];
                                    tanda[ant, m, numfold, i] =
Math.Sign(F[m, numfold, i] + b[ant, numfold, m]);
                                }}

//Mencari status hasil
                        for (i = 0; i < banyaktes[numfold]; i++)
                        {
                            if (tanda[ant, 0, numfold, i] == 1)
                            {
                                hasilSVM[i] = "1";
                            }
                            else if(tanda[ant, 1, numfold, i]==1)
                            {
                                hasilSVM[i] = "2";
                            }
                            else
                            {
                                hasilSVM[i] = "3";
                            }}
```

Gambar 5.7 Implementasi Algoritma Perhitungan Nilai Fungsi *f(x)*

Penjelasan algoritma pada Gambar 5.9 adalah sebagai berikut:

1. Menghitung jumlah bobot data *testing* ditunjukkan pada baris 1 sampai baris 20.
2. Menghitung nilai *f(x)* ditunjukkan pada baris 36 – baris 49.
3. Mengecek nilai *f(x),* jika pada level 1 data bernilai 1 maka masuk kelas positif (kelas 1 / normal), selain itu masuk kelas negatif . Jika pada level 2 data bernilai 1 maka masuk kelas positif (kelas 2 / rentan) . langkah ini ditunjukkan pada baris 54 – baris 66.

5.2.6 Implementasi Algoritma Perhitungan Probabilitas

Pada proses ini akan perhitungan probabilitas tiap titik pada semua titik yang kemudian dari hasil probabilitas tersebut akan dicari nilai range dari masing-masing titik untuk nantinya digunakan oleh semut sebagai acuan titik mana yang akan dilalui. Cara perhitungan probabilitas yaitu dengan mengalikan nilai *pheromone* yang telah dipangkat dengan parameter *alpha* pada ACO dengan nilai parameter *phi* yang telah dipangkatkan dengan parameter *beta* yang disimpan di variabel *a*. Kemudia hasil perkalian dibagi dengan jumlah seluruh *a*. Gambar 5.8 menunjukkan implementasi perhitungan probabilitas dan *roulette wheel.*

```
1   for(i=0; i<x; i++)
2                   {
3                       a = 0;
4                       for (j = 0; j < y; j++)
5                       {
6                           a = a + (Math.Pow(pheromone[i][j], alpha)
7   * Math.Pow(phi, beta));
8                       }
9                       for(j=0; j<y; j++)
10                      {
11                          p[i, j] = (Math.Pow(pheromone[i][j],
12  alpha) * Math.Pow(phi, beta))/a;
13                          if (j == 0)
14                          {
15                              kum[i, j] = p[i, j];
16                              rangemin[i, j] = 0;
17                              rangemax[i, j] = kum[i, j];
18                          }
19                          else
20                          {
21                              kum[i, j] = kum[i, j - 1] + p[i, j];
22                              rangemin[i, j] = kum[i, j - 1];
23                              rangemax[i, j] = kum[i, j];
24                          }
                        }
```

Gambar 5.8 Implementasi Algoritma Perhitungan Probabilitas dan *Roulette Wheel*

Penjelasan algoritma pada Gambar 5.8 adalah sebagai berikut:

1. Menghitung probabilitas ditunjukkan pada baris 3 – baris 12.
2. Proses *Roulette Wheel* ditunjukkan pada baris 13 – 23.

5.2.7 Implementasi Algoritma Pemilihan Titik oleh Semut

Pada proses ini semut akan memilih titik mana yang akan dipilih. Setelah didapat semua jalur maka kemudian akan dilakukan kalkulasi dengan menerapkan batas nilai parameter yang telah ditetapkan. Gambar 5.9 menunjukkan implementasi algoritma pemilihan titik oleh semut.

```
1   //semut memilih titik yang akan dituju
2                       for (ant = 0; ant < NumSemut; ant++)
3                       {
4                           angra = bilran.Next(1, 1000000);
5                           fillang[time, i, ant] = angra / 1000000;
6                           if (fillang[time, i, ant] >= rangemin[i,
7   0] && fillang[time, i, ant] <= rangemax[i, 0])
8                           {
9                               c = 0;
10                          }
11                          else if (fillang[time, i, ant] >=
12  rangemin[i, 1] && fillang[time, i, ant] <= rangemax[i, 1])
13                              {
14                                  c = 1;
15                              }
16                          else if (fillang[time, i, ant] >=
17  rangemin[i, 2] && fillang[time, i, ant] <= rangemax[i, 2])
18                                  {
19                                      c = 2;
20                                  }
21                          else if (fillang[time, i, ant] >=
22  rangemin[i, 3] && fillang[time, i, ant] <= rangemax[i, 3])
23                                      {
24                                          c = 3;
25                                      }
26                          else if (fillang[time, i, ant] >=
27  rangemin[i, 4] && fillang[time, i, ant] <= rangemax[i, 4])
28                                          {
29                                              c = 4;
30                                          }
31                          else if (fillang[time, i, ant] >=
32  rangemin[i, 5] && fillang[time, i, ant] <= rangemax[i, 5])
33                                              {
34                                                  c = 5;
35                                              }
36                          else if (fillang[time, i, ant] >=
37  rangemin[i, 6] && fillang[time, i, ant] <= rangemax[i, 6])
38                                                  {
39                                                      c = 6;
40                                                  }
41                          else if (fillang[time, i, ant] >=
42  rangemin[i, 7] && fillang[time, i, ant] <= rangemax[i, 7])
43                                                      {
44                                                          c = 7;
45                                                      }
46                          else if (fillang[time, i, ant] >=
47  rangemin[i, 8] && fillang[time, i, ant] <= rangemax[i, 8])
48                                                          {
49                                                              c = 8;
                                                            }
                                                            else
                                                            {
```

```
50                                                          c = 9;
51                                                      }
52                          path2[ant, i] = c;
53                      }
54  //mulai kalkulasi
55                  for (ant = 0; ant < NumSemut; ant++)
56                  {
57                      for (i = 0; i < x; i++)
58                      {
59                          if (i == 0)
60                          {
61                              path[ant, i] =
62  Convert.ToString(path2[ant, i]);
63                          }
64                          else
65                          {
66                              path[ant, i] =
67  string.Concat(Convert.ToString(path[ant, i - 1]),
68  Convert.ToString(path2[ant, i]));
69                          }
70                      }
71                          string[] hasil = new string[NumSemut];
72                          string[] hasil1 = new string[NumSemut];
73                          Double[] conres = new Double[NumSemut];
74                          Double[] conres1 = new Double[NumSemut];
75
76                          hasil[ant] = path[ant, 9].Substring(0,
77  5);
78                          hasil1[ant] = path[ant, 9].Substring(5);
79
80                          conres[ant] =
81  Convert.ToDouble(hasil[ant]);
82                          conres1[ant] =
83  Convert.ToDouble(hasil1[ant]);
84
85                          res[ant] = conres[ant] / 10000;//100
                            res1[ant] = conres1[ant] / 100;
                    }
```

Gambar 5.9 Implementasi Algoritma Pemilihan Titik oleh Semut

Penjelasan algoritma pada Gambar 5.9 adalah sebagai berikut:

1. Menentukan batas nilai parameter yang akan digunakan ditunjukkan pada baris 83 dan 84.
2. Algoritma semut memilih titik yang akan dipilih ditunjukkan pada baris 2 sampai baris 52.
3. Kalkulasi jalur yang telah didapat semut ditunjukkan pada baris 54 – baris 85.

5.2.8 Implementasi Algoritma Perhitungan Update Pheromone

Setelah didapat nilai parameter kemudian dilakukan perhitungan akurasi dengan *k-fold cross validation*. Setelah itu didapat akurasi terbaik yang akan digunakan untuk melakukan *update pheromone*. *Update pheromone* dilakukan dengan melakukan perkalian antara parameter evaporasi (*rho*) dengan nilai *pheromone* sebelumnya kemudian ditambah dengan hasil pembagian antara parameter *Q* dengan hasil dari 1 dikurangi nilai akurasi rata-rata terbaik

ditambah konstanta *c* (0.01). Perhitungan itu berlaku untuk titik yang memiliki akurasi rata-rata terbaik. Untuk titik yang lain perhitungan nya dilakukan dengan hanya mengalikan parameter evaporasi dengan nilai *pheromone* sebelumnya. Pada titik dengan akurasi terbaik jumlah *pheromone* yang dimiliki akan meningkat sedangkan titik selain itu akan mengalami penurunan jumlah *pheromone*. Gambar 5.10 menunjukkan implementasi perhitungan *update pheromone*.

```
1   for (i = 0; i < x; i++)
2                   {
3                       for (j = 0; j < y; j++)
4                       {
5                           if (j == indeks[i])
6                           {
7                               pheromone[i][j] = (rho *
8   pheromone[i][j]) + (Q / (1 - max + 0.01));
9                           }
10                          else
11                          {
12                              pheromone[i][j] = rho *
13  pheromone[i][j];
14                          }
15                      }
                    }
```

Gambar 5.10 Implementasi Algoritma Perhitungan *Update Pheromone*

Penjelasan algoritma pada Gambar 5.10 adalah sebagai berikut:

1. Perhitungan *update pheromone* untuk titik dengan akurasi terbaik ditunjukkan pada baris 7 dan 8.
2. Perhitungan *update pheromone* untuk titik lainnya ditunjukkan pada baris 11 dan baris 12.

5.2.9 Implementasi Algoritma *K-fold Cross Validation*.

Proses pertama yaitu melakukan pengacakan pada data yang digunakan. Kemudian dilakukan perhitungan *interval* untuk masing-masing *fold* dengan cara membagi jumlah data dengan jumlah *fold*. Untuk kelebihan data akan dimasukkan kedalam *interval fold* terakhir. Kemudian setelah itu dilakukan pembagian mana yang digunakan sebagai data latih dan data uji. Perhitungan akurasi dilakukan setelah proses klasifikasi dengan SVM selesai dimana hasil klasifikasi dari sistem dibandingkan dengan hasil aktual dari data. Perhitungan akurasi dilakukan untuk masing-masing *fold* dimana nanti di akhir perhitungan akan dilakukan perhitungan akurasi rata-rata.

```
static void RandomData(double[][] Data)
        {
            Random rndm = new Random(0);
            for (int z = 0; z < Data.Length; z++)
            {
                int r = rndm.Next(z, Data.Length);
                Double[] tmp = Data[r];
                Data[r] = Data[z];
                Data[z] = tmp;
            }
        }

static int[][] Interval(int jumdata, int nofold)
        {
            //
            int interval = jumdata / nofold;
            int[][] result = new int[nofold][];

            for (int i = 0; i < result.Length; ++i)
                result[i] = new int[2];

            for (int t = 0; t < nofold; ++t)
            {
                int awal = t * interval;
                int akhir = (t + 1) * interval - 1;
                result[t][0] = awal;
                result[t][1] = akhir;
            }
            result[nofold - 1][1] = result[nofold - 1][1] +
jumdata % nofold;
            return result;
        }

static double[][][] DataTes(double[][] Data, int fold)
        {
            // TEST data
            int[][] AwalAkhirTest = Interval(Data.Length, fold);
int[] noTest = new int[fold];
            for (int k = 0; k < fold; ++k)
            {
                noTest[k] = AwalAkhirTest[k][1] -
AwalAkhirTest[k][0] + 1;
            }

            double[][][] result = new double[fold][][];
            for (int i = 0; i < fold; ++i)
            {
                result[i] = new double[noTest[i]][];
                for (int zx = 0; zx < noTest[i]; zx++)
                {
                    result[i][zx] = new double[7];
                }
            }
}

static double[][][][] DataTrain(double[][] Data, int fold)
        {
            int[][] AwalAkhirTest = Interval(Data.Length, fold);
int[] noTrain = new int[fold];
            for (int k = 0; k < fold; ++k)
```

```
            {
                noTrain[k] = Data.Length - (AwalAkhirTest[k][1] -
AwalAkhirTest[k][0] + 1); //
            }
            double[][][][] result = new double[2][][][];
            for (int lvl = 0; lvl < 2; lvl++)
            {
                result[lvl] = new double[fold][][];
                for (int ik = 0; ik < fold; ik++)
                {
                    result[lvl][ik] = new
double[noTrain[ik]][];
                    for (int ic = 0; ic < noTrain[ik]; ic++)
                    {
                        result[lvl][ik][ic] = new double[7];
                    }
                }
            }

            for (int ik = 0; ik < fold; ik++)
            {
                int i = 0; //
                int ia = 0; //
                while (i < noTrain[ik])
                {
                    if (ia < AwalAkhirTest[ik][0] || ia >
AwalAkhirTest[ik][1]) //
                    {
                        for (int f = 0; f < 7; f++)
                        {
                            result[0][ik][i][f] =
Data[ia][f];
                        }
                        ++i;
                    }
                    ++ia;
                }
            }

            for (int ik = 0; ik < fold; ik++)
            {
                int junor = 0;
                for (int ab = 0; ab < Data.Length -
(AwalAkhirTest[ik][1] - AwalAkhirTest[ik][0] + 1); ab++)
                {
                    if (result[0][ik][ab][6] == 1)
                    {
                        junor++;
                    }
                }

                int il = 0;
                int ig = 0;
                while (ig < Data.Length -
(AwalAkhirTest[ik][1] - AwalAkhirTest[ik][0] + 1) - junor)
                {

                    if (result[0][ik][il][6] != 1)
                    {
                        for (int f = 0; f < 7; f++)
```

```
116                                 {
117                                     result[1][ik][ig][f] =
118 result[0][ik][il][f];
119                                 }
200                                 ig++;
201                             }
202                             il++;
203                         }
204                     }
205             return result;
206         }
207
208 //akurasi
209                             benar = 0;
300                             salah = 0;
301
302                             for (i = 0; i < banyaktes[numfold]; i++)
303                             {
304                                 if (hasilSVM[i] ==
305 Convert.ToString(TesData[numfold][i][6]))
306                                 {
307                                     benar++;
308                                 }
309                                 else
400                                 {
401                                     salah++;
402                                 }
403                             }
404                             //Akurasi
405                             BenarSalah[numfold, 0] = salah;
406                             BenarSalah[numfold, 1] = benar;
407                             acc[ant, numfold] = BenarSalah[numfold,
408 1] / (BenarSalah[numfold, 0] + BenarSalah[numfold, 1]);
409                             akurasi = akurasi + acc[ant, numfold];
410                         }
411                         //mean
412                         Mean[ant] = akurasi / fold;
413
414 max = Mean.Max();
415                     for (ant = 0; ant < NumSemut; ant++)
416                     {
417                             if(max == Mean[ant])
418                             {
419                                 SemutMax = ant;
420                                 for(i = 0; i < x; i++)
421                                 {
422                                 indeks[i] = path2[ant, i];
423                                 }
424                             }
                    }
```

Gambar 5.11 Implementasi Algoritma *K-fold Cross Validation*

Penjelasan algoritma pada Gambar 5.11 adalah sebagai berikut:

1. Pengacakan data ditunjukkan pada baris 1 – baris 9.
2. Perhitungan pembagian *interval* ditunjukkan pada baris 12 – baris 31.
3. Pembagian data uji ditunjukkan pada baris 33 – baris 52.

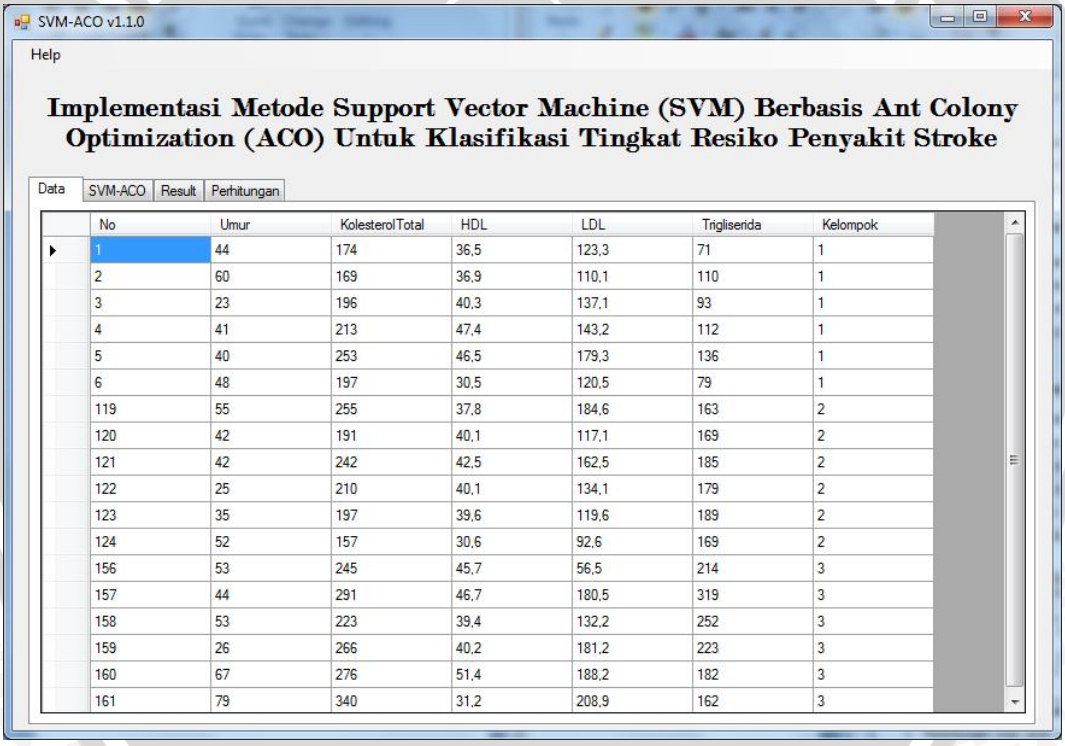
4. Pembagian data latih ditunjukkan pada baris 54 - 204.
5. Menghitung akurasi ditunjukkan pada baris 208 – baris 423.

5.3 Implementasi Antar Muka

Antarmuka sistem digunakan oleh pengguna untuk dapat berinteraksi dengan sistem. Antarmuka dari sistem ini terdiri atas empat bagian, yaitu: antarmuka halaman data, antarmuka halaman parameter, antarmuka halaman hasil perhitungan ACO, dan antarmuka halaman hasil klasifikasi SVM.

5.3.1 Implementasi Antarmuka Halaman Data

Halaman data merupakan halaman yang disediakan untuk menampilkan seluruh data yang akan digunakan dalam sistem untuk melakukan perhitungan. Dalam antarmuka ini hanya menyediakan proses *read* dari database sehingga pengguna hanya dapat melihat isi seluruh data yang ada di *database* dan tidak dapat melakukan manajemen data seperti *create*, *update*, dan *delete*. Implementasi antarmuka halaman data terdapat pada Gambar 5.12.

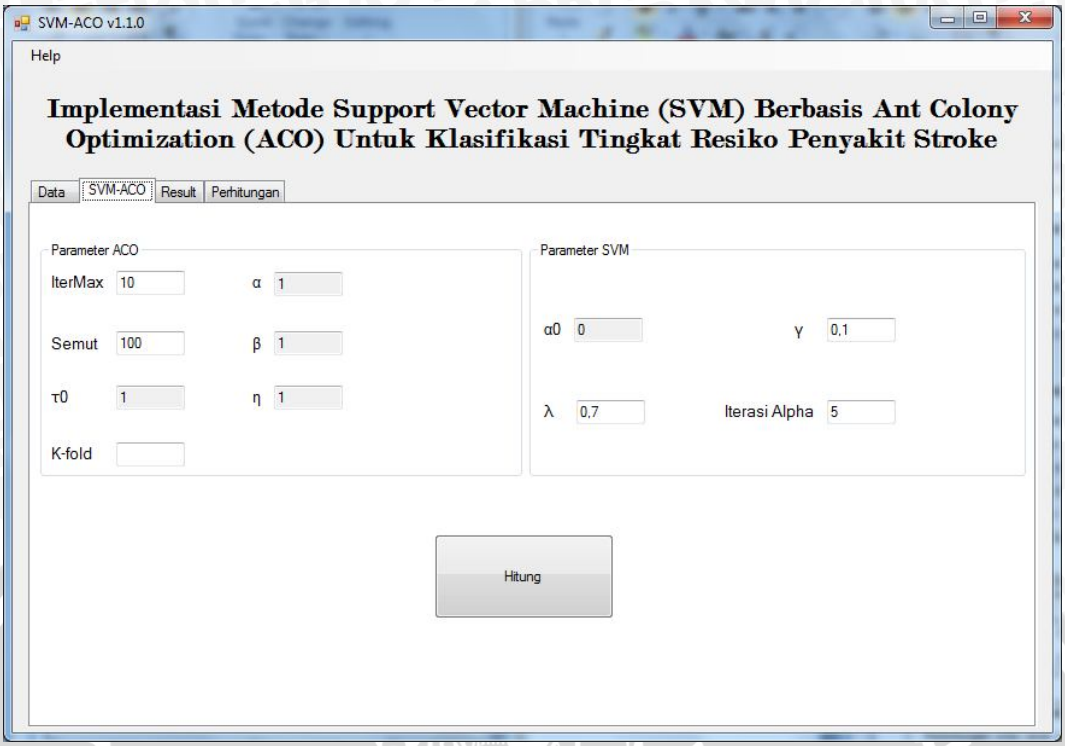


No	Umur	KolesterolTotal	HDL	LDL	Trigliserida	Kelompok
1	44	174	36,5	123,3	71	1
2	60	169	36,9	110,1	110	1
3	23	196	40,3	137,1	93	1
4	41	213	47,4	143,2	112	1
5	40	253	46,5	179,3	136	1
6	48	197	30,5	120,5	79	1
119	55	255	37,8	184,6	163	2
120	42	191	40,1	117,1	169	2
121	42	242	42,5	162,5	185	2
122	25	210	40,1	134,1	179	2
123	35	197	39,6	119,6	189	2
124	52	157	30,6	92,6	169	2
156	53	245	45,7	56,5	214	3
157	44	291	46,7	180,5	319	3
158	53	223	39,4	132,2	252	3
159	26	266	40,2	181,2	223	3
160	67	276	51,4	188,2	182	3
161	79	340	31,2	208,9	162	3

Gambar 5.12 Implementasi Antarmuka Halaman Data

5.3.2 Implementasi Antarmuka Halaman Parameter

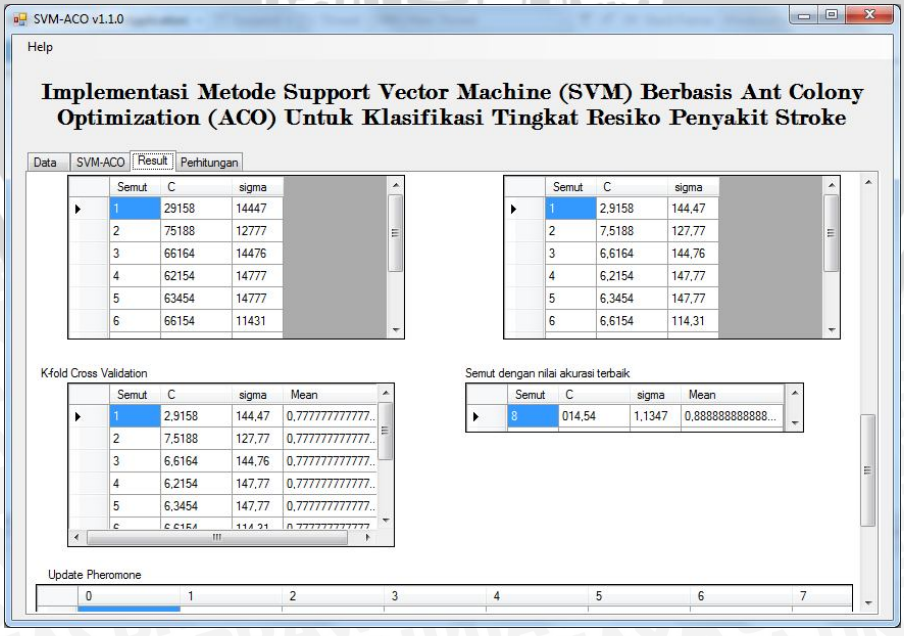
Halaman parameter merupakan halaman yang diakses untuk memberikan masukan pada sistem berupa nilai parameter yang akan digunakan sistem untuk melakukan perhitungan nantinya. Tombol hitung digunakan setelah pengguna memasukkan nilai semua parameter untuk nantinya sistem akan memulai perhitungan. Implementasi antarmuka halaman parameter terdapat pada Gambar 5.13.



Gambar 5.13 Implementasi Antarmuka Halaman Parameter

5.3.3 Implementasi Antarmuka Halaman Hasil Perhitungan ACO

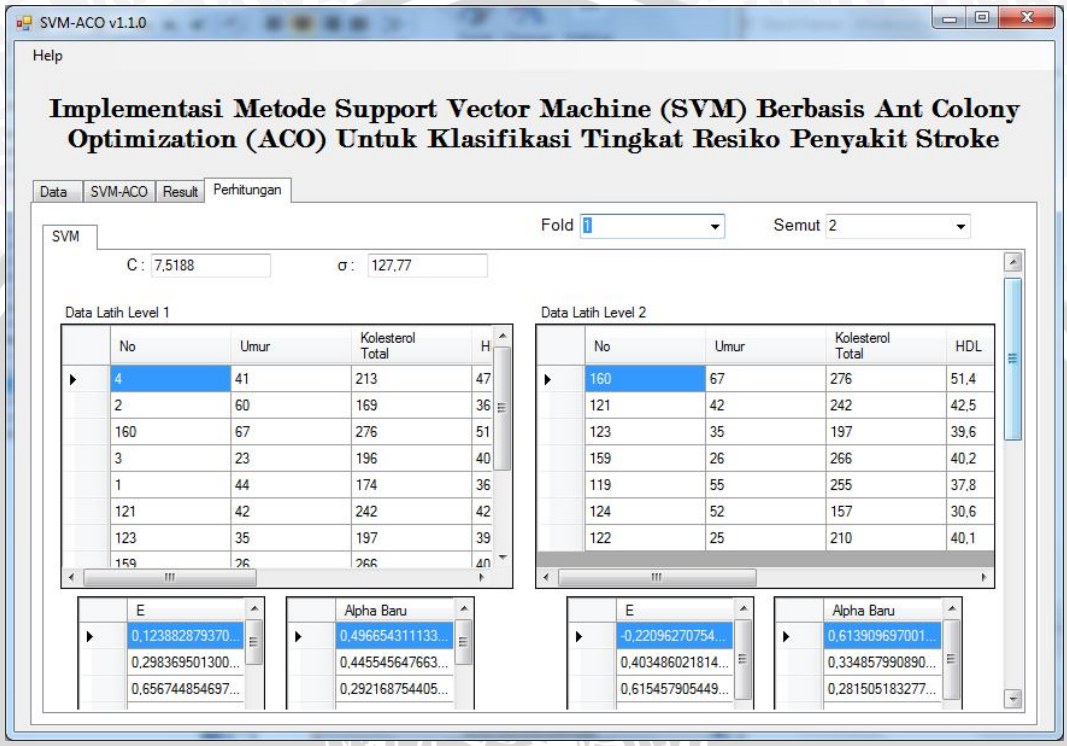
Halaman hasil perhitungan ACO merupakan halaman yang dapat digunakan pengguna untuk melihat hasil dari perhitungan algoritma ACO. Pengguna dapat melihat jalur yang sudah dilalui semua semut, hasil dari proses *K-fold cross validation*, melihat semut terbaik dengan nilai parameter terbaik, dan melihat tabel *pheromone* yang telah di *update*. Implementasi antarmuka halaman hasil perhitungan ACO terdapat pada Gambar 5.14.



Gambar 5.14 Implementasi Antarmuka Halaman Hasil Klasifikasi ACO

5.3.4 Implementasi Antarmuka Halaman Hasil Klasifikasi SVM

Halaman hasil klasifikasi SVM merupakan halaman yang dapat digunakan pengguna untuk melihat hasil perhitungan metode SVM. Pengguna dapat memilih hasil perhitungan saat *fold* keberapa dan semut keberapa yang akan ditampilkan dengan memilih *fold* dan semut pada *combobox* yang telah disediakan. Implementasi antarmuka halaman hasil klasifikasi SVM terdapat pada Gambar 5.15.



Gambar 5.15 Implementasi Antarmuka Halaman Utama Hasil Klasifikasi SVM

BAB 6 PENGUJIAN DAN ANALISIS

Pada bab ini membahas mengenai proses pengujian sistem implementasi metode SVM berbasis ACO untuk klasifikasi tingkat resiko penyakit stroke. Proses pengujian dilakukan untuk mengukur tingkat akurasi dari sistem. Pengujian akurasi dilakukan dengan membandingkan hasil dari keluaran sistem dan hasil dari laboratorium.

6.1 Pengujian Batas Nilai Parameter *C*

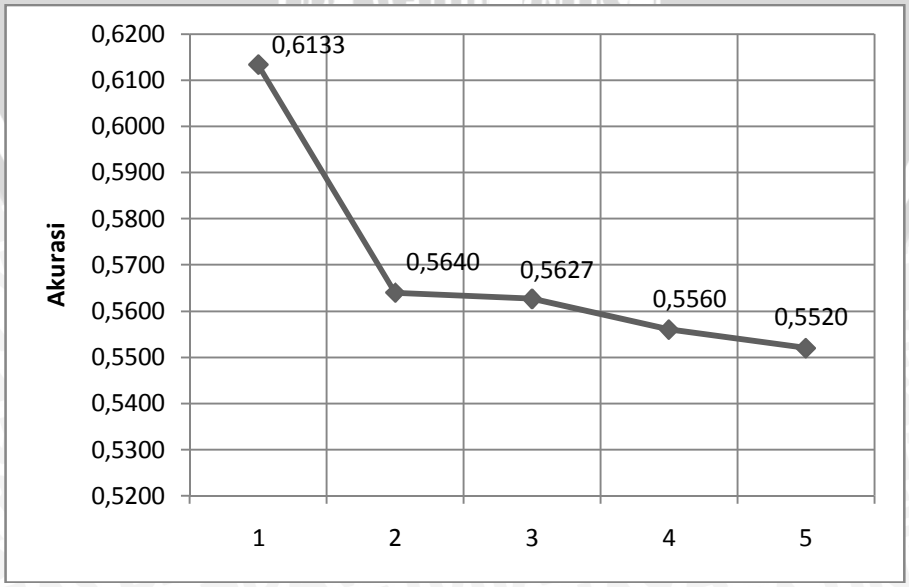
Pada sub bab ini akan menjelaskan tentang skenario pengujian batas nilai parameter *C* terhadap akurasi rata-rata yang dihasilkan, hasil skenario pengujian pertama, dan análisis hasil skenario pengujian pertama.

6.1.1 Skenario Pengujian 1

Pengujian batas nilai parameter *C* yaitu uji coba yang dilakukan untuk mengetahui berapa batasan yang diperlukan untuk menghasilkan nilai parameter *C* yang optimal. Pada pengujian batas nilai parameter *C* batas yang akan diuji yaitu 0,0001 ~ 9,9999, 10,000 ~ 99,999, 100,00 ~ 999,99, 1000,0 ~ 9999,9, dan 10000 ~ 99999. Pada pengujian pertama untuk batas nilai parameter *C* menggunakan parameter tetap yaitu batas parameter σ = 0,0001 ~ 9,9999, jumlah iterasi ACO = 5, jumlah semut =5, nilai parameter λ = 0,2, nilai parameter γ = 0,1, dan panjang *fold* = 3.

6.1.2 Hasil Skenario Pengujian 1

Berdasarkan grafik yang ditunjukkan pada Gambar 6.1, didapat akurasi rata-rata terbaik sebesar 0,6133 pada percobaan pertama yaitu ketika batas nilai parameter *C* = 0,0001 ~ 9,9999. Dengan demikian maka untuk batas nilai parameter *C* = 0,0001 ~ 9,9999 akan digunakan untuk pengujian kedua.



Gambar 6.1 Grafik Hasil Pengujian Batas Nilai Parameter *C*

6.1.3 Analisis Hasil Skenario Pengujian 1

Pada pengujian yang telah dilakukan dapat dilakukan analisa yaitu ketika nilai *C* semakin besar akan mengakibatkan *penalty* terhadap kesalahan yang terjadi akan menjadi semakin besar, sehingga proses *training* menjadi lebih ketat (Vijayakumar, 1999). Ketika nilai *C* terlalu besar, *hyperplane* akan terbentuk sangat dekat dengan data, sehingga *margin* akan menjadi sangat kecil. Apabila data uji jauh berbeda jauh dengan data latih maka akan sangat mungkin terjadi *overfitting*, sehingga nilai akurasi akan menurun. *Overfitting* adalah data yang dipisahkan menghasilkan pembentukan *hyperplane* yang terlalu dekat dengan salah satu kelas.

6.2 Pengujian Batas Nilai Parameter *σ*

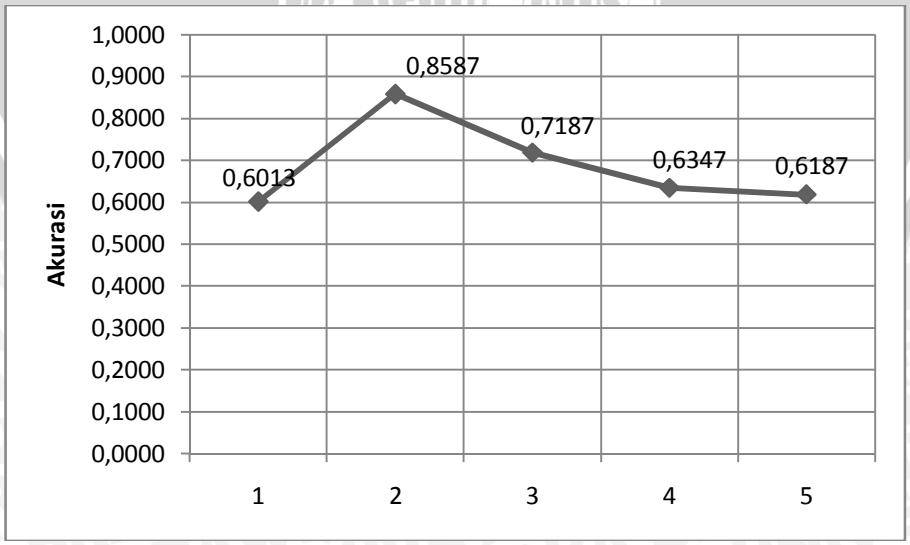
Pada sub bab ini akan menjelaskan tentang skenario pengujian batas nilai parameter *σ* terhadap nilai akurasi rata-rata yang dihasilkan, hasil skenario pengujian kedua, dan análisis hasil skenario pengujian kedua.

6.2.1 Skenario Pengujian 2

Pengujian batas nilai parameter *σ* yaitu uji coba yang dilakukan untuk mengetahui berapa batasan yang diperlukan untuk menghasilkan nilai parameter *σ* yang optimal. Pada pengujian batas nilai parameter *σ* batas yang akan diuji yaitu 0,0001 ~ 9,9999, 10,000 ~ 99,999, 100,00 ~ 999,99, 1000,0 ~ 9999,9, dan 10000 ~ 99999. Pada pengujian kedua untuk batas nilai parameter *σ* menggunakan parameter tetap yaitu batas parameter *C* = 0,0001 ~ 9,9999, jumlah iterasi ACO = 5, jumlah semut =5, nilai parameter *λ* = 0,2, nilai parameter *γ* = 0,1, dan panjang *fold* = 3.

6.2.2 Hasil Skenario Pengujian 2

Pengujian dilakukan sebanyak 10 kali. Hasil pengujian pada Gambar 6.2.



Gambar 6.2 Grafik Hasil Pengujian Batas Nilai Parameter *σ*

Berdasarkan grafik pada Gambar 6.2, akurasi terbaik didapat saat percobaan ke 2 yaitu sebesar 0,8587 yaitu ketika batas nilai parameter σ = 10,000 ~ 99,999. Dengan demikian maka untuk batas nilai parameter σ = 10,000 ~ 99,999 akan digunakan pada pengujian ketiga.

6.2.3 Analisis Hasil Skenario Pengujian 2

Dari hasil pengujian didapatkan bahwa jika nilai σ terlalu kecil maka proses pelatihan SVM akan cenderung *overfitting* dan jika nilai σ terlalu besar maka menyebabkan SVM tidak fleksibel dalam menghitung fungsi yang kompleks (Vijayakumar, 1999). Sehingga tidak dapat dijadikan acuan bahwa nilai parameter σ yang semakin kecil atau semakin besar akan menghasilkan solusi yang optimal.

6.3 Pengujian Nilai Parameter λ

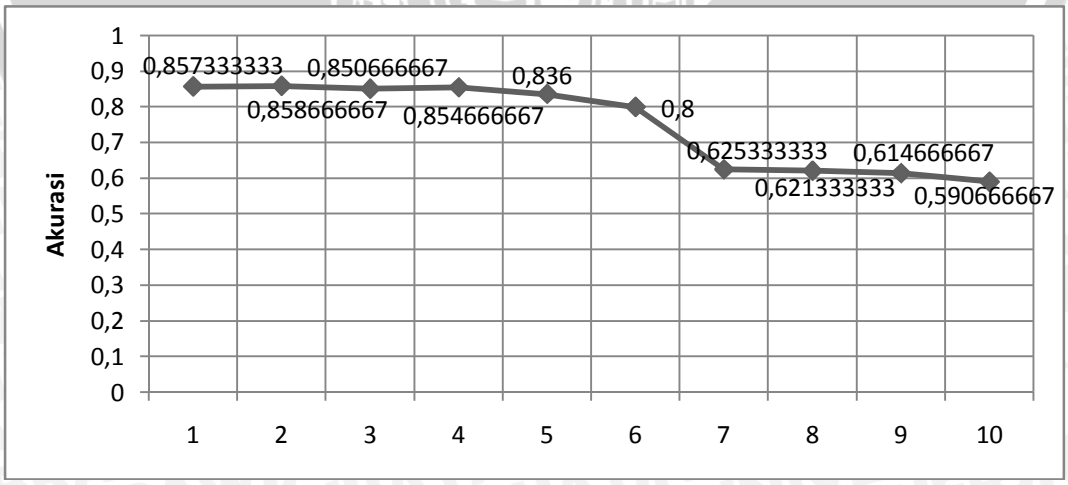
Pada sub bab ini akan menjelaskan tentang skenario pengujian tingkat akurasi terhadap pengaruh nilai parameter λ, hasil skenario pengujian ketiga, dan análisis hasil skenario pengujian ketiga.

6.3.1 Skenario Pengujian 3

Pengujian nilai parameter λ yaitu uji coba yang dilakukan untuk mengetahui berapa nilai parameter λ yang optimal yang diperlukan untuk menghasilkan akurasi rata-rata terbaik. Pada pengujian ini nilai parameter λ yang akan diuji yaitu 0,1, 0,2, 0,3, 0,4, 0,5, 0,6, 0,7, 0,8, 0,9, dan 1. Pada pengujian ketiga untuk nilai parameter λ menggunakan parameter tetap yaitu batas parameter C = 0,0001 ~ 9,9999, batas parameter σ = 10,000 ~ 99,999, jumlah iterasi ACO = 5, jumlah semut =5, nilai parameter γ = 0,1, dan panjang *fold* = 3.

6.3.2 Hasil Skenario Pengujian 3

Pengujian dilakukan sebanyak 10 kali. Hasil pengujian pada Gambar 6.3.



Gambar 6.3 Grafik Hasil Pengujian Nilai Parameter λ

Untuk pengujian nilai parameter *augmenting factor* (λ) digunakan rentang nilai antara 0,1 – 1. Dari grafik pada gambar 6.3 dapat dilihat bahwa untuk nilai parameter λ pada *range* 0,1 – 1 didapat nilai akurasi terbaik 0,8587 yaitu pada saat percobaan ke 2 dengan nilai parameter sebesar 0.2 dan nilai akurasi terendah 0,5907 pada saat percobaan ke 10 dengan nilai parameter sebesar 1. Dengan demikian nilai parameter λ = 0,2 akan digunakan untuk pengujian keempat.

6.3.3 Analisis Hasil Skenario Pengujian 3

Dari hasil pengujian yang telah dilakukan dapat diketahui bahwa hasil akurasi rata-rata yang dihasilkan oleh 10 percobaan menghasilkan akurasi rata-rata terbaik pada saat percobaan ke 2, semakin besar nilai parameter λ maka akurasi rata-rata yang dihasilkan akan semakin menurun. Nilai *augmenting factor* (λ) yang terlalu besar akan mengakibatkan rendahnya konvergensi dan membuat proses *learning* tidak stabil (Vijayakumar, 1999).

6.4 Pengujian Nilai Parameter γ

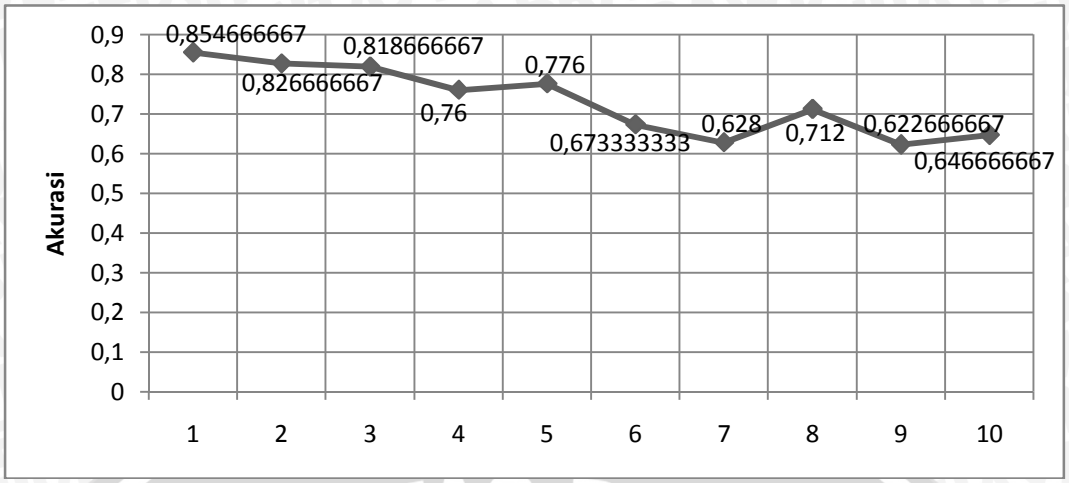
Pada sub bab ini akan menjelaskan tentang skenario pengujian nilai parameter γ terhadap akurasi rata-rata yang dihasilkan, hasil skenario pengujian keempat, dan análisis hasil skenario pengujian keempat.

6.4.1 Skenario Pengujian 4

Pengujian nilai parameter γ yaitu uji coba yang dilakukan untuk mengetahui berapa nilai parameter γ yang optimal yang diperlukan untuk menghasilkan akurasi rata-rata terbaik. Pada pengujian ini nilai parameter γ yang akan diuji yaitu 0,1, 0,2, 0,3, 0,4, 0,5, 0,6, 0,7, 0,8, 0,9, dan 1. Pada pengujian ketiga untuk nilai parameter γ menggunakan parameter tetap yaitu batas parameter C = 0,0001 ~ 9,9999, batas parameter σ = 10,000 ~ 99,999, jumlah iterasi ACO = 5, jumlah semut = 5, nilai parameter λ = 0,2, dan panjang *fold* = 3.

6.4.2 Hasil Skenario Pengujian 4

Pengujian nilai parameter γ ini dilakukan sebanyak 10 kali untuk masing-masing nilai. Hasil pengujian dapat dilihat pada Gambar 6.4. Dari grafik pada Gambar 6.4 dapat dilihat bahwa untuk pengujian nilai parameter γ didapatkan nilai akurasi rata-rata terbaik ketika percobaan pertama yaitu ketika nilai parameter γ = 0,1 dan untuk akurasi rata-rata terendah terjadi ketika percobaan ke 9 ketika nilai parameter γ = 0,9. Dengan demikian nilai parameter γ = 0,1 akan digunakan untuk pengujian kelima.



Gambar 6.4 Grafik Hasil Pengujian Nilai Parameter *γ*

6.4.3 Analisis Hasil Skenario Pengujian 4

Dari pengujian yang telah dilakukan dapat dilihat bahwa untuk nilai parameter *γ* yang semakin besar tidak membuat nilai akurasi rata-rata menjadi lebih baik, namun malah sebaliknya. Dapat dilihat ketika nilai parameter semakin kecil akurasi yang dihasilkan semakin baik. Semakin kecil nilai *γ* maka proses *learning* akan semakin lama namun hasil yang didapatkan akan semakin *konvergen* (Vijayakumar, 1999).

6.5 Pengujian Jumlah Semut

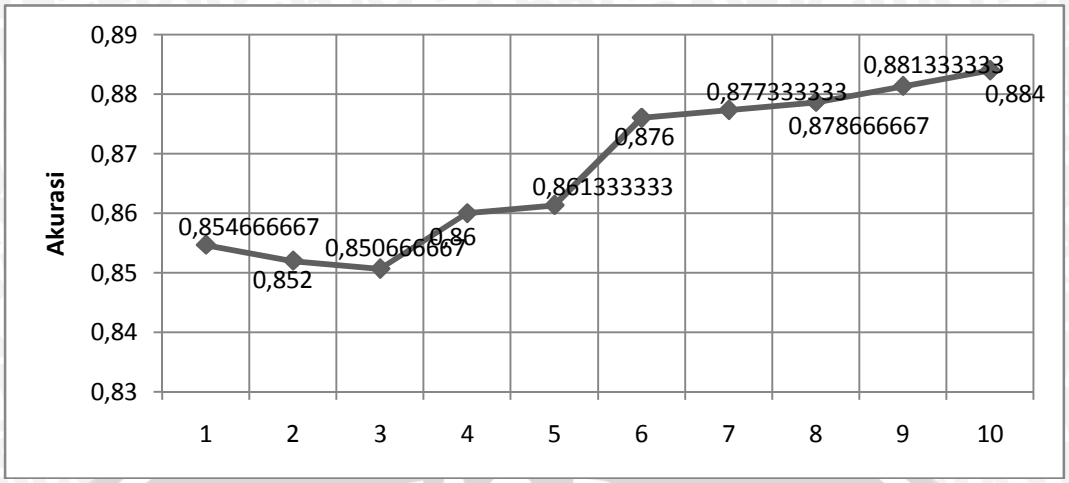
Pada sub bab ini akan menjelaskan tentang skenario pengujian jumlah semut terhadap akurasi rata-rata yang dihasilkan, hasil skenario pengujian kelima, dan análisis hasil skenario pengujian kelima.

6.5.1 Skenario Pengujian 5

Pengujian jumlah semut yaitu uji coba yang dilakukan untuk mengetahui berapa jumlah semut yang diperlukan untuk menghasilkan akurasi rata-rata terbaik. Pada pengujian ini jumlah semut yang akan diuji yaitu 5, 10, 15, 20, 30, 40, 50, 70, 80, dan 100. Pada pengujian keempat untuk jumlah semut menggunakan parameter tetap yaitu batas parameter C = 0,0001 ~ 9,9999, batas parameter σ = 10,000 ~ 99,999, jumlah iterasi ACO = 5, nilai parameter λ = 0,2, nilai parameter γ = 0,1, dan panjang *fold* = 3.

6.5.2 Hasil Skenario Pengujian 5

Pengujian jumlah semut ini dilakukan sebanyak 10 kali untuk masing-masing nilai. Hasil pengujian dapat dilihat pada Gambar 6.5.



Gambar 6.5 Grafik Hasil Pengujian Jumlah Semut

Berdasarkan grafik hasil pengujian pada Gambar 6.5, akurasi rata-rata terbaik yang dihasilkan didapat ketika dilakukan percobaan ke 10 yaitu ketika jumlah semut = 100 dengan nilai akurasi rata-rata sebesar 0,884 dan akurasi rata-rata terendah sebesar 0,8507 didapat saat percobaan pertama ketika jumlah semut = 15. Dengan demikian untuk percobaan keenam akan digunakan jumlah semut sebesar 100.

6.5.3 Analisis Hasil Skenario Pengujian 5

Dari pengujian yang telah dilakukan dapat dilihat bahwa semakin banyak jumlah semut yang digunakan untuk melakukan pencarian nilai parameter maka akurasi yang dihasilkan akan semakin meningkat. Ini dikarenakan semakin banyak jumlah semut yang digunakan maka kemungkinan seluruh titik untuk dieksplorasi oleh semut juga semakin besar sehingga akan didapatkan nilai parameter yang semakin optimal. Kekurangan ketika menggunakan jumlah semut yang banyak terletak pada *running time* yang akan semakin besar sehingga proses menjadi semakin lambat.

6.6 Pengujian Jumlah Iterasi

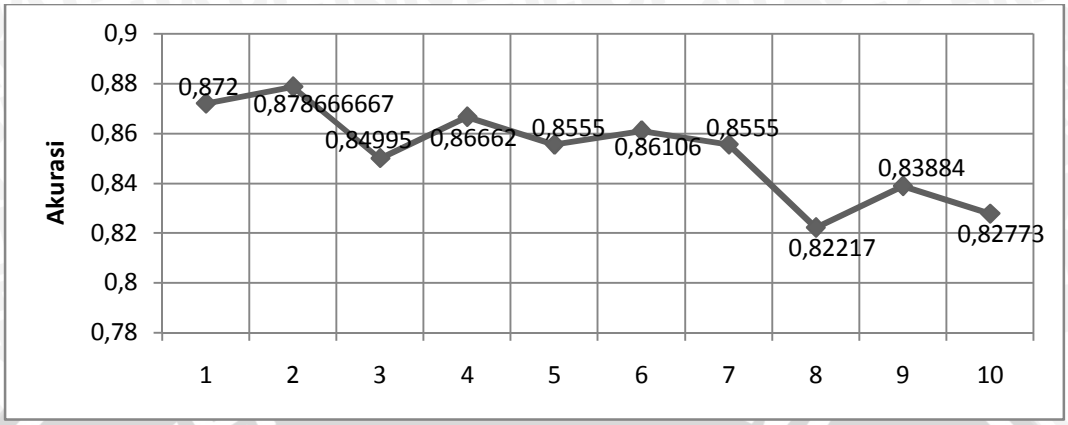
Pada sub bab ini akan menjelaskan tentang skenario pengujian jumlah iterasi terhadap akurasi rata-rata yang dihasilkan, hasil skenario pengujian keenam, dan análisis hasil skenario pengujian keenam.

6.6.1 Skenario Pengujian 6

Pengujian jumlah iterasi yaitu uji coba yang dilakukan untuk mengetahui berapa jumlah iterasi yang diperlukan untuk menghasilkan akurasi rata-rata terbaik. Pada pengujian ini iterasi semut yang akan diuji yaitu 5, 10, 15, 20, 25, 40, 60, 80, 100, dan 200. Pada pengujian keenam untuk jumlah iterasi menggunakan parameter tetap yaitu batas parameter C = 0,0001 ~ 9,9999, batas parameter σ = 10,000 ~ 99,999, jumlah semut = 100, nilai parameter λ = 0,2, nilai parameter γ = 0,1, dan panjang *fold* = 3.

6.6.2 Hasil Skenario Pengujian 6

Pengujian jumlah iterasi ini dilakukan sebanyak 10 kali untuk masing-masing nilai. Hasil pengujian dapat dilihat pada Gambar 6.6.



Gambar 6.6 Grafik Hasil Pengujian Jumlah Iterasi

Berdasarkan grafik hasil pengujian jumlah iterasi pada Gambar 6.6 dapat dilihat untuk akurasi rata-rata terbaik didapat ketika dilakukan percobaan kedua ketika jumlah iterasi = 10. Namun dapat dilihat dari percobaan pertama sampai terakhir dihasilkan nilai akurasi rata-rata yang cenderung stabil.

6.6.3 Analisis Hasil Skenario Pengujian 6

Dari hasil pengujian dapat dilihat bahwa semakin besar jumlah iterasi yang digunakan tidak dapat menjamin bahwa akurasi yang dihasilkan akan semakin meningkat, namun ketika iterasi yang digunakan semakin banyak akan membuat *running time* yang semakin besar dan proses yang semakin lambat. Semakin banyak iterasi yang digunakan akan berpengaruh pada konvergensi semut-semut untuk memilih titik yang sama.

6.7 Pengujian Panjang *fold*

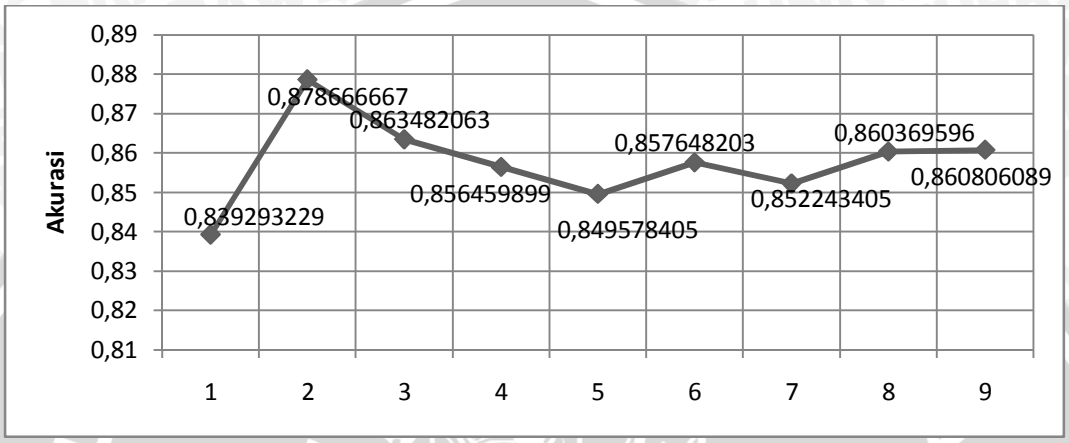
Pada sub bab ini akan menjelaskan tentang skenario pengujian panjang *fold* terhadap akurasi rata-rata yang dihasilkan, hasil skenario pengujian ketujuh, dan análisis hasil skenario pengujian ketujuh.

6.7.1 Skenario Pengujian 7

Pengujian panjang *fold* yaitu uji coba yang dilakukan untuk mengetahui berapa panjang *fold* yang optimal yang diperlukan untuk menghasilkan akurasi rata-rata terbaik. Pada pengujian ini panjang *fold* yang akan diuji yaitu 2, 3, 4, 5, 6, 7, 8, 9, dan 10. Pada pengujian ketujuh untuk panjang *fold* menggunakan parameter tetap yaitu batas parameter C = 0,0001 ~ 9,9999, batas parameter σ = 10,000 ~ 99,999, jumlah semut = 100, jumlah iterasi = 10, nilai parameter λ = 0,2, dan nilai parameter γ = 0,1.

6.7.2 Hasil Skenario Pengujian 7

Pengujian panjang *fold* ini dilakukan sebanyak 9 kali untuk masing-masing panjang *fold*. Hasil pengujian dapat dilihat pada Gambar 6.7. Dari grafik hasil pengujian pada gambar 6.7 dapat dilihat bahwa akurasi rata-rata terbaik yang dihasilkan sebesar 0,87867 yaitu ketika percobaan kedua, ketika panjang *fold* = 3. Akurasi rata-rata terendah dihasilkan sebesar 0,8392 ketika percobaan pertama ketika panjang *fold* = 2.



Gambar 6.7 Grafik Hasil Pengujian Panjang *fold*

6.7.3 Analisis Hasil Skenario Pengujian 7

Dari pengujian yang telah dilakukan dapat dilihat bahwa untuk panjang *fold* = 2 akurasi rata-rata yang dihasilkan jauh menurun. Ini dikarenakan saat panjang *fold* = 2 maka data latih yang digunakan jumlahnya akan sama dengan data uji, sehingga proses pelatihan akan semakin sedikit atau setengah dari keseluruhan data. Panjang *fold* yang optimal adalah ketika panjang *fold* = 3. Semakin panjang *fold* yang digunakan akan mengurangi nilai akurasi rata-rata yang dihasilkan karena nilai pembagi rata-rata akurasi juga semakin besar, menyesuaikan dengan panjang *fold* yang digunakan.

BAB 7 PENUTUP

Bab ini membahas tentang kesimpulan dan saran dari penelitian yang telah dilakukan berdasarkan hasil perancangan, implementasi dan pengujian sistem.

7.1 Kesimpulan

Berdasarkan pada hasil perancangan, implementasi dan pengujian yang telah dilakukan maka dapat diambil kesimpulan sebagai berikut:

1. Untuk menerapkan metode *Support Vector Machine* berbasis *Ant Colony Optimization* untuk klasifikasi tingkat resiko penyakit stroke adalah dengan mendapatkan nilai parameter *C* dan *σ* menggunakan algoritma ACO dengan bantuan *Roulette Wheel*, melakukan pelatihan SVM dengan melakukan metode *one against all*, menghitung kernel, *sequential learning*, menghitung nilai bias, menghitung akurasi dengan *K-fold cross validation*, dan terakhir memperbarui tabel *pheromone*. Proses diawali dengan menentukan parameter yang akan digunakan oleh ACO, kemudian semua semut dengan bantuan *Roulette Wheel* akan memilih titik mana yang akan dipilih. Setelah menempuh semua jalur, akan dilakukan kalkulasi dengan batasan nilai yang telah ditetapkan sehingga nanti akan didapatkan nilai parameter yang akan digunakan untuk pelatihan SVM. Kemudian dihitung nilai akurasi rata-rata dengan *K-fold cross validation*. Akurasi rata-rata terbaik dari semua semut akan digunakan untuk melakukan pembaruan tabel *pheromone*. Untuk proses SVM digunakan metode *multi class one against all* karena data pada penelitian ini dibagi menjadi 3 kelas, sehingga perhitungan SVM dilakukan 2 kali. Lalu menghitung nilai kernel RBF karena data yang dimiliki berupa data *non*-linier. Kemudian dilakukan sequential training untuk mendapatkan nilai *α* baru. Setelah itu dihitung nilai bias *b* untuk nantinya akan dihitung fungsi yang akan digunakan untuk melakukan klasifikasi sehingga dapat dihitung nilai akurasinya. Nilai akurasi ini merupakan bagian dari *K-fold cross validation*.
2. Berdasarkan hasil pengujian yang telah dilakukan dapat dilihat bahwa akurasi rata-rata terbaik yang dihasilkan sebesar 0,87867 dengan batas nilai parameter *C* = 0,0001 ~ 9,9999, batas nilai parameter *σ* = 10,000 ~ 99,999, nilai parameter *augmenting factor* (*λ*) = 0,2, nilai parameter *γ* = 0,1, jumlah semut = 100, jumlah iterasi = 10, dan panjang *fold* = 3.

7.2 Saran

Untuk pengembangan sistem lebih lanjut perlu dilakukan beberapa penelitian salah satunya mengenai penggunaan metode *multi class* pada SVM. Perlu dilakukan studi perbandingan untuk studi kasus ini antara metode *multi class One Against All*, *One Against One*, dan *Directed Acyclic Graph Support Vector Machine* (DAGSVM). Penelitian lebih lanjut juga dapat dilakukan terhadap penggunaan *kernel trick* selain kernel RBF untuk mengetahui kernel mana yang

lebih sesuai dengan data yang digunakan pada penelitian ini. yang terakhir perlu dilakukan pengembangan dalam tahapan pengujian agar nantinya didapatkan hasil akurasi yang lebih baik.

DAFTAR PUSTAKA

Akbar, A. L., Yudistira, N., dan Cholissodin, Imam. 2015. "Implementasi Algoritma SVM (*Support Vector Machine*) untuk mengetahui tingkat resiko penyakit stroke". Universitas Brawijaya, Malang.

A. P. Yudis, F. Chastine. 2009."Implementasi Metode *Ant Colony Optimization* Untuk pemilihan Fitur Pada Kategorisasi Dokumen Teks". Institut Teknologi Sepuluh November.

Boswell, Dustin (2002). "*Introduction to Support Vector Machines*".

C. Cortes and V. N. Vapnik, "*Support-vector networks*," *Machine Learning*, vol. 20, no. 3, pp. 273–297, 1995.

Christianini, Nello. *Support Vector and Kernel Machines*. ICML tutorial, 2001.

Chao Zhang, H-C Mei, Hao Yang. 2014. "*A Parallel Way to Select the Parameters of SVM Based on the Ant Optimization Algorithm*". SiChuan University, China.

Donnan GA, Fisher M, Macleod M, Davis SM (May 2008). "Stroke". Lancet 371 (9624): 1612–23.

Dorigo,M. 1996. "*The Ant System: Optimization by a colony of cooperating agents",* IEEE *transactions of Systems*, *Man*, *and Cybernetics*-Part B,Vol.26, No.1.

Hacke W, Kaste M, Bogousslavsky J, Brainin M, Chamorro A, Lees K et al.. *Ischemic Stroke Prophylaxis and Treatment - European Stroke Initiative Recommendations* 2003.

Han, J., Kamber, M. 2006. "*Data Mining: Concepts ant Techniques*". University of Illinois at Urbana-Champaign, 285.

Hastie, T., Tibshirani, R. & Friedman, J., 2009. *The Elements of Statistical Learning* Second., New York: Springer-Verlag.

Khan, Aurangzeb, Baharum Baharudin, Lam Hong Lee, and Khairullah Khan. 2010, "*A Review of Machine Learning Algoritms for Text-Documents Classification*", *Journal of Advances in Information Technology*, Vol. 1, No.1, hal 4-20.

Kementrian Kesehatan RI. 2013. "Riset Kesehatan Dasar". Jakarta: Kementrian Kesehatan RI, 1 Desember, 91 - 94.

Mozaffarian D, Benjamin EJ, *et al*; *on behalf of the American Heart Association Statistics Committee and Stroke Statistics Subcommittee. Heart disease and stroke statistics— 2015 update: a report from the American Heart Association.*

Nugroho, Anto Satrio, Witarto, Arief Budi, dan Handoko, Dwi. 2003. "*Support Vector Machine* – Teori dan Aplikasinya dalam Bioinformatika".

Santosa, Budi. 2012. "*Ant Colony Optimization*", Teknik Industri, Institut Teknologi Sepuluh Nopember, Surabaya.

Santosa, Budi. 2012. "*Support Vector Machine*", Teknik Industri, Institut Teknologi Sepuluh Nopember, Surabaya.

Sembiring, Krisantus. 2007."Penerapan Teknik *Support Vector Machine* untuk Pendeteksian Intrusi pada Jaringan". Sekolah Teknik Elektro dan Informatika, ITB.

Sims NR, Muyderman H. 2009. "*Mitochondria, oxidative metabolism and cell death in stroke*". *Biochimica et Biophysica Acta* 1802 (1): 80–91.

Vijayakumar, Sethu and Si Wu. 1999, "Sequential Support Vector Classifiers and Regression", *Proceeding International Conference on Soft Computing* (SOCO '99), Genoa, Italy, pp. 610-619.

WHO. MONICA. *Manual Version* 1: 1. 1986.

Yayasan Stroke Indonesia. 2011. Sekilas Tentang Stroke. Jakarta.

LAMPIRAN A DATA

No	Umur	*Cholestrol Total*	HDL	LDL	Trigliserida	Kelompok
1	44	174	36,5	123,3	71	1
2	60	169	36,9	110,1	110	1
3	23	196	40,3	137,1	93	1
4	41	213	47,4	143,2	112	1
5	40	253	46,5	179,3	136	1
6	48	197	30,5	120,5	79	1
7	49	171	40,1	104,1	134	1
8	40	198	40,5	139,5	90	1
9	49	184	38,4	119,2	133	1
10	59	280	30,5	202,8	128	1
11	60	201	41,6	136,4	115	1
12	29	145	30,5	94,7	99	1
13	46	181	37,7	118,3	125	1
14	66	278	57,2	201,2	98	1
15	48	186	36,2	121,8	140	1
16	66	186	38,1	132,1	79	1
17	61	205	45,2	140,6	96	1
18	28	182	37,7	116,9	137	1
19	34	140	38,8	137,8	128	1
20	75	170	37,4	110,2	112	1
21	59	280	47,6	203,2	146	1
22	45	197	37,4	144,4	76	1
23	49	186	36,1	127,5	112	1
24	37	188	36,4	126,4	126	1
25	39	159	36,7	149,9	142	1
26	53	277	51,7	208,1	86	1
27	72	253	35,1	188,1	149	1
28	54	232	47,9	164,9	96	1

No	Umur	*Cholestrol Total*	HDL	LDL	Trigliserida	Kelompok
29	71	150	34,5	134,6	138	1
30	32	229	43,5	166,3	96	1
31	54	171	38,1	115,9	85	1
32	52	166	37,8	112,6	33,8	1
33	52	203	38,6	135,6	144	1
34	58	194	40,1	65,9	129	1
35	62	168	35,1	110,1	114	1
36	71	174	35,7	122,1	81	1
37	34	251	47,1	182,9	105	1
38	45	207	46,3	138,9	109	1
39	60	180	42,1	121,1	84	1
40	48	278	30,2	218,8	145	1
41	47	284	52,8	201,4	149	1
42	50	184	33,6	111,8	93	1
43	53	170	39,7	103,5	134	1
44	43	211	35,9	152,3	114	1
45	35	169	37,1	117,1	74	1
46	70	170	47,1	150,7	109	1
47	49	236	42,8	171,4	109	1
48	49	178	42,1	118,9	85	1
49	62	179	40,1	120,9	90	1
50	35	207	40,2	145,4	107	1
51	36	209	44,1	145,3	98	1
52	59	191	36,1	126,2	142	1
53	52	211	42,7	139,1	146	1
54	62	169	35,5	110	104	1
55	59	189	38,2	134,6	81	1
56	46	232	41,6	163,8	133	1
57	32	170	36,5	113,9	98	1
58	61	193	36,7	138,7	88	1

No	Umur	*Cholestrol Total*	HDL	LDL	Trigliserida	Kelompok
59	57	206	37,6	140	142	1
60	38	300	48,1	225,3	133	1
61	56	178	38,4	110,8	144	1
62	28	276	54,6	202,6	94	1
63	39	276	34,5	293,6	115	1
64	49	182	38,6	122,6	104	1
65	70	216	42,5	144,1	147	1
66	73	200	47,1	133,1	99	1
67	50	200	40	141	95	1
68	68	219	41,6	148,4	145	1
69	76	197	37,6	137,4	110	1
70	56	261	40,5	193,9	133	1
71	30	199	40,2	141,6	86	1
72	23	223	44,6	158,8	98	1
73	63	249	41,5	183,7	119	1
74	57	229	40,8	159,8	142	1
75	66	224	37,2	157,4	147	1
76	30	180	38,4	122,4	96	1
77	73	161	37,8	105,6	88	1
78	62	160	37,2	107	79	1
79	57	240	44,2	171,6	121	1
80	50	184	36,7	125,9	107	1
81	56	148	32,8	102,8	67	1
82	57	179	39,5	120,7	94	1
83	49	258	40,5	192,7	124	1
84	60	182	37,9	125,5	93	1
85	38	246	45,1	141,1	134	1
86	41	238	44,2	168	129	1
87	56	240	45,1	176,3	133	1
88	56	274	39,5	206,1	142	1

No	Umur	*Cholestrol Total*	HDL	LDL	Trigliserida	Kelompok
89	58	164	39,5	107,1	87	1
90	60	192	37,2	126,8	140	1
91	70	173	36,2	111	132	1
92	52	197	40,6	136,6	94	1
93	60	169	37,8	102,6	143	1
94	53	163	40,1	105,8	83	1
95	54	221	39,6	156,3	126	1
96	57	170	37,6	109,4	115	1
97	43	175	38,6	117	97	1
98	78	199	36,6	145,2	86	1
99	42	207	45,3	133,3	142	1
100	58	225	38,4	159,2	137	1
101	39	179	38,4	116,2	122	1
102	27	165	39,2	106,8	95	1
103	40	211	45,6	167,8	104	1
104	30	176	36,7	121,6	86	1
105	60	281	48,7	206,5	129	1
106	48	210	34,8	213,6	116	1
107	43	206	33,4	107,8	91	1
108	62	147	35,1	94,5	87	1
109	95	192	39,2	123,8	145	1
110	52	241	40,8	177,4	114	1
111	43	181	39,8	124,8	82	1
112	46	255	53,7	110,3	90	1
113	33	227	40,1	164,5	112	1
114	56	162	35,4	112	73	1
115	70	200	36,5	136,1	137	1
116	45	219	41,5	152,1	127	1
117	54	198	38,7	142,1	86	1
118	50	172	38,7	112,5	106	1

No	Umur	*Cholestrol Total*	HDL	LDL	Trigliserida	Kelompok
119	55	255	37,8	184,6	163	2
120	42	191	40,1	117,1	169	2
121	42	242	42,5	162,5	185	2
122	25	210	40,1	134,1	179	2
123	35	197	39,6	119,6	189	2
124	52	157	30,6	92,6	169	2
125	51	194	40,3	129,5	180	2
126	43	199	35,4	131,4	161	2
127	52	176	40,1	104,5	157	2
128	61	267	41,5	188,1	187	2
129	49	212	40,5	139,7	159	2
130	57	168	40,8	88,8	192	2
131	60	203	45,1	124,9	165	2
132	28	185	42,3	111,5	156	2
133	40	224	45,6	193	186	2
134	52	319	57,2	228	169	2
135	29	213	36,7	145,3	155	2
136	37	239	40	267,9	187	2
137	46	207	38,2	136,8	160	2
138	57	226	42,5	145,5	190	2
139	54	186	36,6	109,8	198	2
140	45	249	44,2	170,6	171	2
141	52	230	36,2	107,6	181	2
142	54	199	42,2	119,2	188	2
143	50	67	42,1	112,1	164	2
144	47	249	42,3	119,7	150	2
145	41	209	37,1	138,3	168	2
146	56	418	45,7	338,1	171	2
147	54	202	38,1	129,5	172	2
148	57	176	39,8	105,6	153	2

No	Umur	*Cholestrol Total*	HDL	LDL	Trigliserida	Kelompok
149	47	235	45,2	152	189	2
150	55	169	33,9	95,8	198	2
151	65	254	38,6	183,6	159	2
152	49	302	52,4	212	187	2
153	32	237	46,4	159	158	2
154	51	166	34,1	100,5	157	2
155	50	189	35,6	119,2	171	2
156	53	245	45,7	56,5	214	3
157	44	291	46,7	180,5	319	3
158	53	223	39,4	132,2	252	3
159	26	266	40,2	181,2	223	3
160	67	276	51,4	188,2	182	3
161	79	340	31,2	208,9	162	3
162	38	287	51,6	172,2	316	3
163	53	266	45,7	165,5	274	3
164	62	162	36,2	81	224	3
165	45	265	47,6	150,2	241	3
166	57	212	36,9	102,9	361	3
167	44	268	43,6	161,2	316	3
168	85	278	45,3	140,5	261	3
169	69	163	32,5	100	152	3
170	41	173	37,3	113,5	281	3
171	33	259	34,1	166,3	293	3
172	52	287	36,9	193,3	284	3
173	56	189	32,1	91,7	326	3
174	49	324	34	252,2	189	3
175	68	202	37,8	133	156	3
176	55	180	36,7	102,5	204	3
177	65	226	36,1	145,3	223	3
178	42	228	38,6	135,8	268	3

No	Umur	*Cholestrol Total*	HDL	LDL	Trigliserida	Kelompok
179	49	204	33,6	38,3	461	3
180	51	235	41,6	146,2	236	3
181	56	260	42,2	170,8	235	3
182	50	180	35,1	68,7	381	3
183	29	201	36,1	147,3	226	3
184	58	259	44,2	174,6	201	3
185	53	246	44,9	153,5	238	3
186	56	261	40,6	176,2	221	3
187	50	187	36,1	91,1	295	3
188	64	192	36,1	101,9	260	3
189	53	264	36,9	159,3	319	3
190	51	253	39,3	162,5	256	3
191	48	201	39,5	116,7	224	3
192	50	196	34,5	105,1	282	3
193	41	193	35,7	113,1	221	3
194	91	212	30,4	167,9	155	3
195	69	240	48,3	160,7	155	3
196	80	247	40,5	159,9	233	3
197	81	233	40,7	159,5	164	3
198	46	250	40,8	169	201	3
199	70	175	35,1	99,7	201	3
200	52	514	32,8	318,6	813	3

Keterangan :

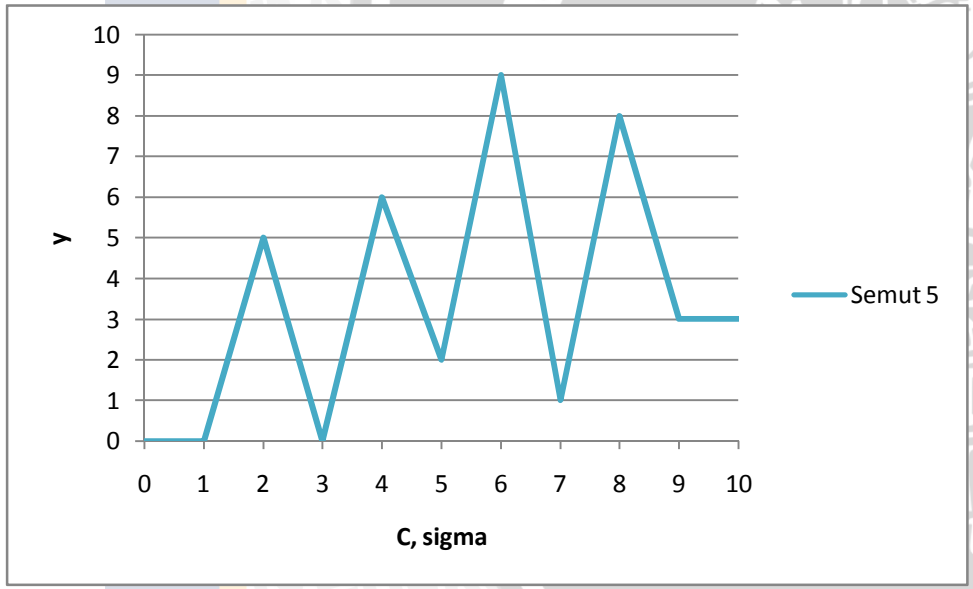
- Kelompok 1 = Normal
- Kelompok 2 = Rentan
- Kelompok 3 = Mengkhawatirkan

LAMPIRAN B HASIL PENGUJIAN

Pengujian Batas Nilai Parameter C

- **Nilai Parameter Hasil ACO**

Semut	C	sigma	Akurasi Rata-rata
5	0,5062	9,1833	0,666666667



- **Nilai *α***

i	*fold 1*		*fold 2*		*fold 3*	
	α Level 1	*α Level 2*	*α Level 1*	*α Level 2*	*α Level 1*	*α Level 2*
1	0,41863043	0,412731467	0,370247349	0,4105889	0,3548226	0,3968083
2	0,44389187	0,394972793	0,444459708	0,3931301	0,4612034	0,3539126
3	0,35608397	0,411180383	0,356334438	0,4302197	0,3168163	0,429865
4	0,36644293	0,395002517	0,353403893	0,403589	0,350536	0,3849321
5	0,35465642	0,376667943	0,366520508	0,4097922	0,3466049	0,3969145
6	0,3664714	0,395001515	0,37234801	0,391426	0,3577612	0,3698377
7	0,32408213	0,424017483	0,355282587	0,4539022	0,3322742	0,4658465
8	0,36647044	0,423301333	0,333660897	0,4092291	0,3201381	0,4220423
9	0,3664714	0,394665913	0,372245361	0,4091615	0,3577204	0,4150238
10	0,36568649	0,422735312	0,37232842	0,4037538	0,3514854	0,3962701
11	0,43068404	0,412094591	0,441449059	0,4092047	0,4525428	0,4220195
12	0,3661489	0,415520543	0,366813851	0,4097911	0,3572049	0,396955
13	0,36685729	0,392736716	0,441629124	0,4028258	0,4435033	0,3312918
14	0,35549658	0,361762225	0,446814729	0,4098233	0,4613149	0,3969998
15	0,34850914	0,443673547	0,432909817	0,4107199	0,4604884	0,4263786
16	0,35209785	0,376365026	0,372369322	0,449062	0,3578407	0,4279836
17	0,36430093	0,394783358	0,372346406	0,4130573	0,3577994	0,4003453
18	0,4514287	0,395002517	0,435379021	0,4043773	0,4541636	0,3557852
19	0,45254858	0,387910212	0,358999021	0,4092047	0,2897051	0,4219976
20	0,39262672	0,394999236	0,372361811	0,4092047	0,3578414	0,396205
21	0,33464913	0,395001513	0,42403055	0,4038399	0,4352369	0,4220204

i	*fold 1*		*fold 2*		*fold 3*	
	α Level 1	*α Level 2*	*α Level 1*	*α Level 2*	*α Level 1*	*α Level 2*
22	0,34842327	0,422350603	0,37146262	0,4086689	0,3542729	0,3347163
23	0,32276485	0,394180748	0,336541195	0,4088568	0,3143979	0,4220203
24	0,36626146	0,364036904	0,425265201	0,4092047	0,4393217	0,3969997
25	0,3664714	0,395002517	0,36721543	0,4092047	0,3528084	0,3961916
26	0,44500302	0,395002516	0,440941345	0,4098153	0,4287821	0,4219763
27	0,35967743	0,424017483	0,367796925	0,408321	0,4476557	0,3970783
28	0,36646819	0,391730096	0,372369321	0,404421	0,3191597	0,3774159
29	0,36647044	0,390323122	0,372369321	0,4135352	0,3578208	0,4418692
30	0,365837	0,410459739	0,374691894	0,4081392	0,3570925	0,4150238
31	0,45254423	0,420035682	0,444434733	0,403732	0,3578414	0,4220191
32	0,3656841	0,411846391	0,371861832	0,4092126	0,2968817	0,4167115
33	0,33683018	0,395002517	0,372039753	-	0,3578415	0,299349
34	0,3664714	-	0,372369322	-	0,3578415	0,4219984
35	0,43038841	-	0,426765485	-	0,3570795	0,4220203
36	0,44901235	-	0,444515934	-	0,459781	-
37	0,3664714	-	0,372369321	-	0,3578015	-
38	0,44959845	-	0,443933838	-	0,355601	-
39	0,3664714	-	0,372369322	-	0,3394447	-
40	0,36333849	-	0,371532181	-	0,3400839	-
41	0,44829785	-	0,44284757	-	0,4049953	-
42	0,3789814	-	0,386123353	-	0,3514854	-
43	0,33338813	-	0,379584161	-	0,4611781	-

i	*fold 1*		*fold 2*		*fold 3*	
	α Level 1	*α Level 2*	*α Level 1*	*α Level 2*	*α Level 1*	*α Level 2*
44	0,36185744	-	0,367838316	-	0,4611776	-
45	0,35409759	-	0,343959047	-	0,4124447	-
46	0,45254423	-	0,44664631	-	0,3578404	-
47	0,36280771	-	0,370784689	-	0,3394114	-
48	0,45141482	-	0,446639911	-	0,2604369	-
49	0,35526759	-	0,366637724	-	0,3578216	-
50	0,3664714	-	0,372361806	-	0,3578415	-

- **Nilai Bias**

fold 1		*fold 2*		*fold 3*	
bias Level 1	*bias Level 2*	*bias Level 1*	*bias Level 2*	*bias Level 1*	*bias Level 2*
-0,0437444	-0,017715	-0,0428566	-0,01080457	-0,0514138	-0,0081188

- **Nilai *f(x)***

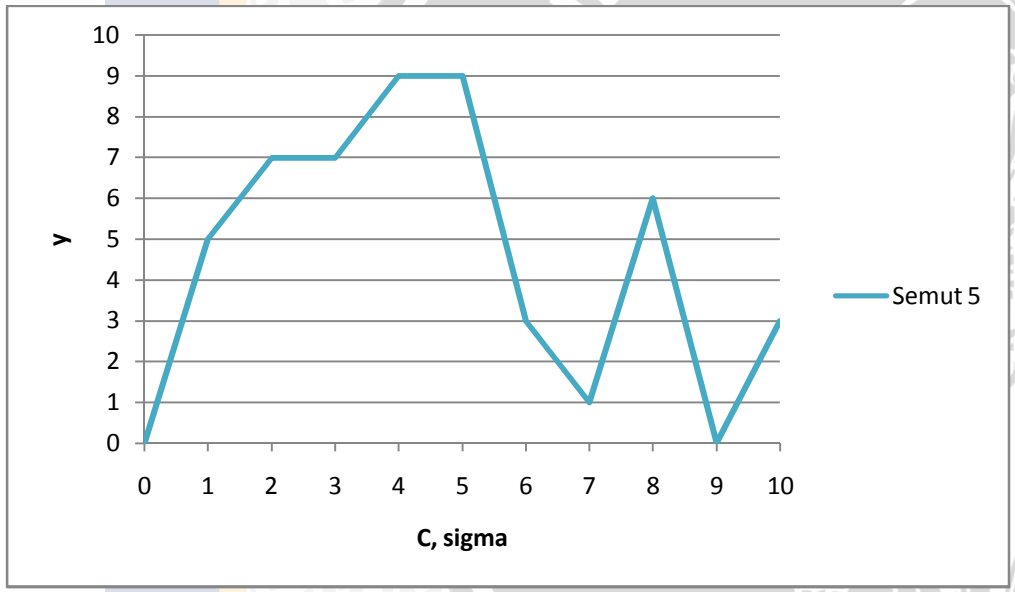
fold 1					*fold 2*					*fold 3*				
No	*f(x) Level 1*	*f(x) Level 2*	kelas prediksi	kelas aktual	No	*f(x) Level 1*	*f(x) Level 2*	kelas prediksi	kelas aktual	No	*f(x) Level 1*	*f(x) Level 2*	kelas prediksi	kelas aktual
124	-0,04839	-0,01236	3	2	23	-0,01092	-0,01080	3	1	6	-0,03975	-0,00812	3	1
21	-0,03207	-0,01774	3	1	11	0,01142	-0,01082	1	1	180	-0,06222	-0,02088	3	3
125	-0,22793	0,19590	2	2	142	-0,20265	0,17056	2	2	159	-0,05143	-0,00814	3	3
175	-0,13593	0,09029	2	3	172	-0,04289	-0,01085	3	3	173	-0,05142	-0,00812	3	3
135	-0,08244	0,02736	2	2	121	-0,09738	0,04923	2	2	132	-0,04575	0,02734	2	2

fold 1					*fold 2*					*fold 3*				
No	*f(x) Level 1*	*f(x) Level 2*	kelas prediksi	kelas aktual	No	*f(x) Level 1*	*f(x) Level 2*	kelas prediksi	kelas aktual	No	*f(x) Level 1*	*f(x) Level 2*	kelas prediksi	kelas aktual
141	-0,04421	-0,01718	3	2	161	-0,04286	-0,01080	3	3	10	-0,03949	-0,00812	3	1
120	-0,16545	0,12509	2	2	126	-0,20450	0,16132	2	2	171	-0,05275	-0,00970	3	3
176	-0,04547	-0,01572	3	3	166	-0,04286	-0,01081	3	3	157	-0,19513	-0,17761	3	3
164	-0,04374	-0,01772	3	3	136	-0,04286	-0,01080	3	2	156	-0,05141	-0,00812	3	3
178	-0,07494	-0,05134	3	3	138	-0,04357	-0,01070	3	2	8	0,14465	-0,00812	1	1
17	0,02742	-0,01772	1	1	22	0,07531	-0,01080	1	1	4	-0,03070	-0,00812	3	1
119	-0,07428	0,01714	2	2	163	-0,04424	-0,01232	3	3	170	-0,05141	-0,00812	3	3
18	0,04999	0,01781	1	1	127	-0,05176	-0,00468	3	2	1	-0,05015	-0,00812	3	1
5	-0,04386	-0,01758	3	1	123	-0,10660	0,06034	2	2	143	-0,05141	-0,00812	3	2
3	0,03025	-0,01772	1	1	160	-0,04326	-0,01033	3	3	167	-0,06277	-0,02151	3	3
168	-0,04375	-0,01772	3	3	13	0,34591	-0,01069	1	1	16	-0,04093	-0,00812	3	1
133	-0,04404	-0,01738	3	2	158	-0,08507	-0,05720	3	3	15	0,12170	0,00303	1	1
7	0,01988	-0,00233	1	1	19	-0,03706	-0,01080	3	1	9	0,37129	-0,00555	1	1
129	-0,36090	0,35189	2	2	12	-0,04285	-0,01080	3	1	177	-0,05204	-0,00817	3	3
134	-0,04378	-0,01775	3	2	24	0,15069	-0,01055	1	1	131	-0,22619	0,01759	2	2
2	-0,02822	-0,01771	3	1	162	-0,20405	-0,18795	3	3	14	-0,05141	-0,00812	3	1
169	-0,05687	-0,00253	3	3	128	-0,04681	-0,00645	3	2	122	-0,07406	0,01720	2	2
139	-0,08607	0,03135	2	2	137	-0,39728	0,35524	2	2	25	-0,04539	-0,00812	3	1
20	-0,04307	-0,01772	3	1	165	-0,04378	-0,01182	3	3	140	-0,13131	0,08071	2	2
130	-0,04378	-0,01767	3	2	179	-0,04286	-0,01080	3	3	174	-0,05145	-0,00808	3	3

Pengujian Batas Nilai Parameter *σ*

- **Nilai Parameter Hasil ACO**

Semut	C	sigma	Akurasi Rata-rata
5	5,7799	31,603	0,866666667



- **Nilai *α***

i	*fold 1*		*fold 2*		*fold 3*	
	α Level 1	*α Level 2*	*α Level 1*	*α Level 2*	*α Level 1*	*α Level 2*
1	0,38476897	0,321306301	0,269853112	0,4299866	0,274618	0,4032248
2	0,15379666	0,228734037	0,334438039	0,3649801	0,326505	0,3512108
3	0,17074384	0,403937589	0,352687523	0,4260347	0,358284	0,3847895
4	0,3044696	0,250468667	0,154784733	0,3996761	0,1549867	0,433679
5	0,10832491	0,296934156	0,168464674	0,1654237	0,0879608	0,1697534
6	0,37125157	0,400226873	0,033547185	0,2080959	0,0364074	0,2631523
7	0,39327577	0,164473167	0,201676888	0,2156874	0,1667988	0,2037641
8	0,23519441	0,295748898	0,171026605	0,2501319	0,3060969	0,2241548
9	0,16294541	0,298473515	0,399317429	0,2787079	0,2739886	0,3052459
10	0,37930991	0,396224995	0,17766066	0,536258	0,1599133	0,5563063
11	0,14971533	0,401145101	0,402883063	0,6100238	0,3314881	0,5398144
12	0,26111662	0,288390656	0,105941921	0,4112616	0,1329817	0,4246995
13	0,28154977	0,43980535	0,367833739	0,3872482	0,3619326	0,3822545
14	0,21204184	0,18577625	0,307127082	0,3847538	0,4462818	0,3441401
15	0,38017683	0,137623149	0,275803173	0,414684	0,2272765	0,4252308
16	0,45393773	0,244532283	0,354462932	0,4287354	0,31909	0,3763084
17	0,1391377	0,232176528	0,373096483	0,2258316	0,3661438	0,2302772
18	0,3465294	0,273654329	0,289481584	0,2641653	0,3810221	0,3091247
19	0,07757053	0,300493184	0,224387816	0,2955072	0,1929629	0,3218911
20	0,21685471	0,230758194	0,152268436	0,6714333	0,1092799	0,6747247
21	0,28354122	0,275417393	0,24116259	0,3213949	0,286495	0,3326497

i	*fold 1*		*fold 2*		*fold 3*	
	α Level 1	*α Level 2*	*α Level 1*	*α Level 2*	*α Level 1*	*α Level 2*
22	0,2880936	0,186034163	0,270924741	0,3111323	0,2977581	0,3327859
23	0,1904061	0,461354873	0,491394218	0,3555892	0,4718395	0,4006083
24	0,22885226	0,392086691	0,266043182	0,3400055	0,223012	0,2865072
25	0,15741396	0,557899712	0,321301476	0,337069	0,3676834	0,2852885
26	0,09701438	0,25791531	0,181665719	0,1929437	0,405299	0,3617165
27	0,22397751	0,211163091	0,066878202	0,4531269	0,2616826	0,2028
28	0,1257893	0,132841204	0,256572156	0,4025195	0,32402	0,3466523
29	0,22647945	0,141699519	0,273938771	0,4596308	0,3087291	0,310594
30	0,15428347	0,284505597	0,264494632	0,251142	0,1176131	0,3803436
31	0,13557975	-	0,218692571	0,2523267	0,3477499	0,3930936
32	0,22097577	-	0,278177089	0,1981428	0,4252681	0,2948987
33	0,03851459	-	0,111765669	0,1830413	0,2668258	0,3850969
34	0,15630078	-	0,085980681	0,2816461	0,1910556	0,2252862
35	0,35992713	-	0,358731146	-	0,3356015	0,1706554
36	0,36452819	-	0,335598518	-	0,1272597	0,2921537
37	0,15865263	-	0,158105946	-	0,3663379	-
38	0,23422472	-	0,278274561	-	0,2874586	-
39	0,3307952	-	0,385096978	-	0,3077147	-
40	0,47803429	-	0,494747722	-	0,3652658	-
41	0,25396478	-	0,2936188	-	0,2422725	-
42	0,2033467	-	0,242058258	-	0,2775943	-
43	0,27158295	-	0,342030614	-	0,3517685	-

i	*fold 1*		*fold 2*		*fold 3*	
	α Level 1	*α Level 2*	*α Level 1*	*α Level 2*	*α Level 1*	*α Level 2*
44	0,24618462	-	0,229827514	-	0,1838587	-
45	0,20217554	-	0,230960953	-	0,2810732	-
46	0,32054675	-	0,330233704	-	0,257512	-
47	0,14944645	-	0,223296351	-	0,1454902	-
48	0,25514866	-	0,207159787	-	0,2441364	-
49	0,16308667	-	0,073073275	-	0,3358503	-
50	0,2682852	-	0,257432226	-	0,2264426	-

- **Nilai Bias**

fold 1		*fold 2*		*fold 3*	
bias Level 1	*bias Level 2*	*bias Level 1*	*bias Level 2*	*bias Level 1*	*bias Level 2*
-0,1427346	-0,195637	-0,0479756	-0,311831555	-0,0714528	-0,3393993

- **Nilai *f(x)***

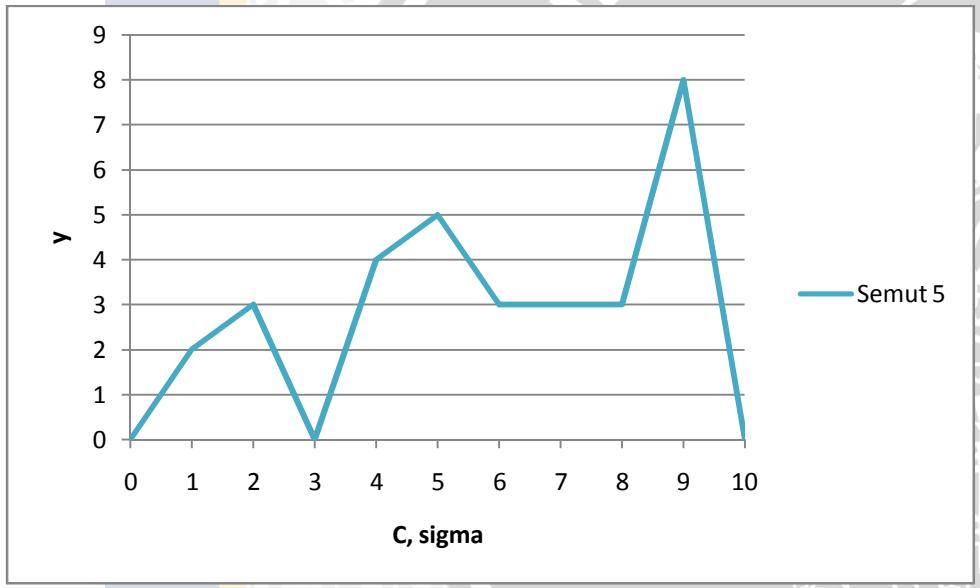
fold 1					*fold 2*					*fold 3*				
No	*f(x) Level 1*	*f(x) Level 2*	kelas prediksi	kelas aktual	No	*f(x) Level 1*	*f(x) Level 2*	kelas prediksi	kelas aktual	No	*f(x) Level 1*	*f(x) Level 2*	kelas prediksi	kelas aktual
164	-0,29572	-0,06700	3	3	14	0,16318	-0,30931	1	1	121	-0,77233	0,30984	2	2
170	-0,23543	-0,28318	3	3	22	0,83253	-0,31204	1	1	139	-1,10755	-0,00871	3	2
161	-0,25836	-0,13157	3	3	24	1,09116	-0,05455	1	1	17	1,04032	-0,32733	1	1
130	-0,57007	0,25815	2	2	168	-0,28399	-0,58476	3	3	133	-0,43333	0,05816	2	2

fold 1					*fold 2*					*fold 3*				
No	*f(x) Level 1*	*f(x) Level 2*	kelas prediksi	kelas aktual	No	*f(x) Level 1*	*f(x) Level 2*	kelas prediksi	kelas aktual	No	*f(x) Level 1*	*f(x) Level 2*	kelas prediksi	kelas aktual
120	-0,80939	0,90594	2	2	138	-1,11043	0,30720	2	2	140	-0,53819	0,40331	2	2
142	-1,14154	0,81877	2	2	156	-0,11045	-0,27916	3	3	6	0,91989	-0,33130	1	1
129	-0,84338	0,93981	2	2	10	0,33103	-0,30847	1	1	158	-0,79149	-1,14272	3	3
13	1,05365	0,19156	1	1	178	-0,59962	-0,95282	3	3	23	1,32776	-0,21979	1	1
132	-0,20635	0,56980	2	2	180	-0,77428	-0,89508	3	3	123	-1,12660	0,53457	2	2
141	-0,81627	0,36642	2	2	166	-0,19467	-0,47120	3	3	134	-0,33410	-0,58926	3	2
169	-0,07055	0,20864	2	3	122	-0,87382	0,68255	2	2	25	0,35214	-0,11497	1	1
176	-0,76436	0,38317	2	3	119	-0,25649	0,00835	2	2	2	1,08939	-0,35220	1	1
136	-0,17128	-0,17783	3	2	157	-0,69471	-1,01965	3	3	9	0,59122	0,05967	1	1
7	0,55594	0,21553	1	1	128	-0,58566	-0,38461	3	2	15	0,21635	0,20786	1	1
124	-0,35858	0,23264	2	2	179	-0,04798	-0,31184	3	3	21	0,20112	-0,07224	1	1
173	-0,30058	-0,36293	2	3	127	-0,31566	0,19428	2	2	159	-0,43215	-0,44417	3	3
143	-0,14115	-0,19471	3	2	11	0,89955	-0,29754	1	1	1	0,75453	-0,33752	1	1
19	0,44773	-0,09656	1	1	174	-0,26677	-0,17482	3	3	160	-0,48803	0,15823	2	3
177	-0,94174	-0,39167	3	3	8	1,02122	-0,29601	1	1	171	-0,64000	-0,96775	3	3
131	-0,78562	0,82518	2	2	4	0,64115	-0,18592	1	1	163	-0,78297	-1,13631	3	3
167	-0,83449	-0,93280	3	3	135	-0,45995	0,50394	2	2	18	0,37016	0,28345	1	1
172	-0,65945	-0,75417	3	3	126	-0,62623	0,61521	2	2	16	0,83328	-0,35107	1	1
5	-0,14797	0,26507	2	1	165	-0,62619	-0,87698	3	3	137	-0,87192	0,66926	2	2
175	-0,44004	0,62501	2	3	20	0,84526	-0,47219	1	1	125	-1,19483	0,54396	2	2
12	0,23560	-0,16615	1	1	3	0,81643	-0,27359	1	1	162	-0,69827	-1,01789	3	3

Pengujian Nilai Parameter λ

- **Nilai Parameter Hasil ACO**

Semut	C	sigma	Akurasi Rata-rata
5	2,3045	33,38	0,906666667



- **Nilai *α***

i	*fold 1*		*fold 2*		*fold 3*	
	α Level 1	*α Level 2*	*α Level 1*	*α Level 2*	*α Level 1*	*α Level 2*
1	0,377399529	0,409806	0,347030977	0,452977192	0,327067661	0,499260254
2	0,360611153	0,40982	0,338640375	0,367990375	0,31863989	0,330964461
3	0,37570499	0,175023	0,177051586	0,550098175	0,110777545	0,578174898
4	0,057237824	0,324957	0,223384973	0,301228011	0,187794573	0,22357285
5	0,181827029	0,241959	0,075001454	0,199334865	0,132816825	0,219272326
6	0,216689912	0,374278	0,304163295	0,427726394	0,292500315	0,406247887
7	0,364316609	0,318224	0,345351445	0,30001768	0,346684139	0,273351455
8	0,483375701	0,330069	0,275554438	0,231658965	0,264659592	0,209195845
9	0,318807011	0,324636	0,151505435	0,255943545	0,216569634	0,224667267
10	0,116506761	0,130336	0,372926407	0,54463325	0,340730526	0,542185046
11	0,210023676	0,490828	0,224241194	0,408073777	0,219101855	0,413387816
12	0,206162295	0,310786	0,382285784	0,408579234	0,387850495	0,397607849
13	0,34512955	0,234301	0,239503029	0,291278541	0,182737916	0,221090129
14	0,293009012	0,161341	0,196551283	0,286599895	0,155727224	0,318164983
15	0,217443726	0,426727	0,150613557	0,399740829	0,090486788	0,387922569
16	0,301947761	0,406214	0,226303246	0,23465292	0,188259874	0,22498146
17	0,187129268	0,553073	0,349763397	0,410930262	0,331636467	0,430281808
18	0,301934228	0,374859	0,279653374	0,544715234	0,279726076	0,418321723
19	0,244985864	0,258491	0,34696484	0,395273126	0,349725723	0,252651062
20	0,286337281	0,402517	0,211917362	0,209531067	0,160799479	0,25716078

i	*fold 1*		*fold 2*		*fold 3*	
	α Level 1	*α Level 2*	*α Level 1*	*α Level 2*	*α Level 1*	*α Level 2*
21	0,177789032	0,293503	0,267782029	0,413627413	0,21373981	0,352080555
22	0,128136531	0,233555	0,151964875	0,311593257	0,188650108	0,363751976
23	0,166122448	0,13608	0,368111969	0,316909518	0,376619116	0,234051091
24	0,171832399	0,166604	0,216684482	0,152968642	0,219250586	0,423205053
25	0,337120765	0,142172	0,223258788	0,195797646	0,182845206	0,323647903
26	0,368596331	0,267448	0,359822908	0,168045782	0,377728537	0,213959743
27	0,284149465	0,263719	0,257859736	0,234310661	0,399164694	0,419332485
28	0,345584103	0,708114	0,363959851	0,28877627	0,32927129	0,222581548
29	0,2881381	0,143589	0,272371555	0,679745731	0,076276637	0,39283893
30	0,243506441	0,396814	0,190404119	0,210485078	0,188193283	0,220895032
31	0,402451313	0,248055	0,374766209	0,394694624	0,178299578	0,344523083
32	0,216382415	0,113711	0,209799248	0,256482805	0,389851647	-
33	0,212460522	0,292614	0,223497501	0,177672325	0,438623152	-
34	0,367345151	0,188721	0,31780061	0,332795974	0,280543626	-
35	0,09423624	-	0,114444045	0,226753191	0,085042416	-
36	0,232281072	-	0,268745148	-	0,138530185	-
37	0,163591984	-	0,183495138	-	0,150931204	-
38	0,233466853	-	0,262435068	-	0,353361911	-
39	0,165225171	-	0,190581988	-	0,22590262	-
40	0,43512374	-	0,579962081	-	0,16080988	-
41	0,045773013	-	0,080285597	-	0,315996628	-
42	0,200522796	-	0,206385033	-	0,301524716	-

i	*fold 1*		*fold 2*		*fold 3*	
	α Level 1	*α Level 2*	*α Level 1*	*α Level 2*	*α Level 1*	*α Level 2*
43	0,277508868	-	0,269240665	-	0,257839337	-
44	0,073956838	-	0,078167821	-	0,155077176	-
45	0,238962266	-	0,238971619	-	0,214733567	-
46	0,429903677	-	0,425151479	-	0,373722643	-
47	0,082931538	-	0,113139746	-	0,070664823	-
48	0,154095169	-	0,149147049	-	0,290286181	-
49	0,301318301	-	0,250008421	-	0,186739748	-
50	0,347768917	-	0,353130069	-	0,323542687	-

- **Nilai Bias**

fold 1		*fold 2*		*fold 3*	
bias Level 1	*bias Level 2*	*bias Level 1*	*bias Level 2*	*bias Level 1*	*bias Level 2*
-0,0119962	-0,406593	0,112170661	-0,336601334	-0,0933093	-0,2112762

- **Nilai *f(x)***

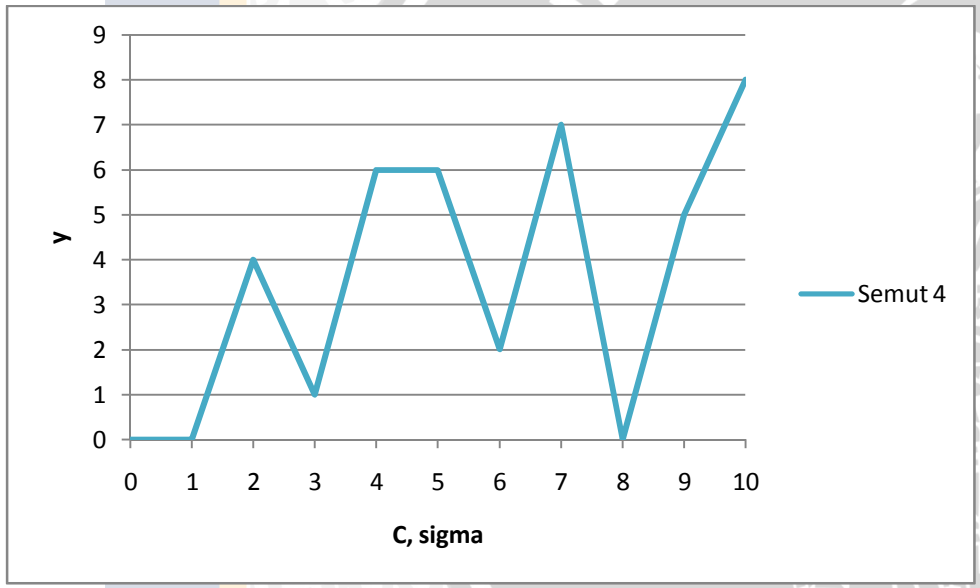
fold 1					*fold 2*					*fold 3*				
No	*f(x) Level 1*	*f(x) Level 2*	Kelas prediksi	kelas aktual	No	*f(x) Level 1*	*f(x) Level 2*	Kelas prediksi	Kelas aktual	No	*f(x) Level 1*	*f(x) Level 2*	Kelas prediksi	Kelas aktual
134	-0,11372	-0,63787	3	2	143	0,11684	-0,33316	1	2	136	-0,10635	-0,21127	3	2
170	-0,17144	-0,57420	3	3	10	0,27925	-0,20626	1	1	160	-0,23376	-0,01411	3	3
176	-0,92482	0,14784	2	3	174	-0,12712	-0,18649	3	3	166	-0,26077	-0,38387	3	3
158	-0,54564	-1,00278	3	3	142	-1,05145	0,36981	2	2	4	0,80804	0,03127	1	1

fold 1					*fold 2*					*fold 3*				
No	*f(x) Level 1*	*f(x) Level 2*	Kelas prediksi	kelas aktual	No	*f(x) Level 1*	*f(x) Level 2*	Kelas prediksi	Kelas aktual	No	*f(x) Level 1*	*f(x) Level 2*	Kelas prediksi	Kelas aktual
125	-1,13122	0,56454	2	2	180	-0,72475	-0,83596	3	3	163	-0,91915	-1,09477	3	3
19	0,45834	-0,26808	1	1	13	1,36155	-0,06705	1	1	18	0,62946	0,19552	1	1
161	-0,02345	-0,50221	3	3	14	0,25709	-0,30970	1	1	128	-0,29359	0,00313	2	2
172	-0,51694	-0,97057	3	3	5	0,00504	0,06992	1	1	133	-0,34740	0,04119	2	2
131	-0,83939	0,32179	2	2	9	1,00211	0,02473	1	1	127	-0,37692	0,37023	2	2
7	0,70953	0,02339	1	1	139	-0,89888	-0,06993	3	2	122	-0,81446	0,60985	2	2
178	-0,43543	-0,95365	3	3	6	1,24058	-0,33021	1	1	135	-0,38462	0,59412	2	2
169	-0,00136	-0,00035	3	3	17	1,25462	-0,33419	1	1	2	1,02833	-0,23652	1	1
1	0,87983	-0,39595	1	1	173	-0,14729	-0,62069	3	3	20	0,68044	-0,27320	1	1
3	0,98916	-0,32895	1	1	157	-0,34758	-0,84080	3	3	137	-0,66070	0,86820	2	2
8	1,19922	-0,35823	1	1	130	-0,66830	-0,25880	3	2	21	0,27195	-0,03172	1	1
24	0,97802	-0,00804	1	1	168	-0,21870	-0,70827	3	3	138	-1,04036	0,34502	2	2
25	0,28429	-0,13780	1	1	126	-0,51043	0,65432	2	2	165	-0,70594	-0,82911	3	3
124	-0,43521	0,16521	2	2	164	-0,21599	-0,56048	3	3	175	-0,53066	0,65224	2	3
156	-0,06757	-0,39169	3	3	22	1,15512	-0,32946	1	1	123	-1,03870	0,61224	2	2
16	1,00501	-0,40975	1	1	162	-0,40062	-0,90065	3	3	159	-0,35576	-0,33123	3	3
167	-0,59290	-1,04474	3	3	140	-0,72550	0,49342	2	2	15	0,42936	0,34985	1	1
141	-0,75241	0,14397	2	2	23	1,68734	-0,22729	1	1	121	-0,68657	0,26187	2	2
179	-0,01257	-0,40721	3	3	129	-0,60089	0,60832	2	2	120	-0,78669	0,80583	2	2
171	-0,55056	-1,02792	3	3	177	-0,61935	-0,37832	3	3	119	-0,07997	0,18481	2	2
11	1,05170	-0,32467	1	1	12	0,67988	-0,30830	1	1	132	-0,13444	0,46507	2	2

Pengujian Nilai Parameter *γ*

- **Nilai Parameter Hasil ACO**

Semut	C	sigma	Akurasi Rata-rata
4	0,4166	27,058	0,88



- **Nilai *α***

i	*fold 1*		*fold 2*		*fold 3*	
	α Level 1	*α Level 2*	*α Level 1*	*α Level 2*	*α Level 1*	*α Level 2*
1	0,36109706	0,39336517	0,145480997	0,33392596	0,17368271	0,27610754
2	0,33882092	0,2267709	0,204348536	0,4166	0,21133563	0,4166
3	0,24881581	0,23888969	0,346537842	0,21956891	0,34143033	0,16829752
4	0,21250561	0,35886219	0,377174784	0,40802414	0,38736478	0,40993533
5	0,26843572	0,20207191	0,12751165	0,26793548	0,07847274	0,26276297
6	0,22372514	0,17622395	0,32967779	0,4166	0,29482366	0,4166
7	0,27351397	0,4166	0,219642979	0,4166	0,21225806	0,4166
8	0,15206678	0,39162419	0,375411506	0,4166	0,37575732	0,32048872
9	0,16165096	0,22761987	0,4166	0,32467158	0,4166	0,2435971
10	0,27896122	0,18798895	0,254616232	0,33823838	0,21476682	0,3339811
11	0,250098	0,39247406	0,34412809	0,37113245	0,35175329	0,26901233
12	0,26107858	0,29896535	0,306499224	0,28933142	0,34018339	0,26398605
13	0,09124103	0,36798561	0,265103175	0,40101167	0,27560825	0,41306892
14	0,14171512	0,4166	0,238009368	0,20261561	0,15467438	0,21753892
15	0,38545085	0,34519948	0,255930211	0,4166	0,23915133	0,40583439
16	0,27880327	0,21993769	0,35869871	0,2942091	0,31140526	0,4166
17	0,27562794	0,3575374	0,184294426	0,4166	0,18428258	0,35012833
18	0,34698608	0,39122921	0,207831819	0,30356547	0,11165976	0,39846891
19	0,34323684	0,4166	0,29562447	0,19288723	0,27227317	0,4166
20	0,35234268	0,27463899	0,377598659	0,41212374	0,36696424	0,19416821

i	*fold 1*		*fold 2*		*fold 3*	
	α Level 1	*α Level 2*	*α Level 1*	*α Level 2*	*α Level 1*	*α Level 2*
21	0,23237059	0,4166	0,4166	0,2750359	0,4166	0,17628297
22	0,15206269	0,24679628	0,23412985	0,4166	0,14902492	0,4166
23	0,33698221	0,18971679	0,219102277	0,33826749	0,21577899	0,40935387
24	0,31169743	0,38731245	0,378082162	0,30411253	0,36671008	0,14966904
25	0,3659153	0,23009473	0,258583107	0,27571657	0,27909692	0,15111903
26	0,25184267	0,4166	0,2587172	0,38644886	0,36104802	0,22411031
27	0,4166	0,2920963	0,369600823	0,32159333	0,2855445	0,3179289
28	0,20302308	0,24302155	0,175461499	0,23846458	0,19647029	0,39121979
29	0,21288167	0,33962187	0,173573935	0,41650859	0,31118503	0,4166
30	0,3837827	0,36762372	0,328034827	0,37908831	0,19503266	0,28250441
31	0,23142634	0,25410675	0,284527497	0,29276393	0,35351356	0,2097279
32	0,14252497	0,24126809	0,259107797	-	0,23534576	0,39118325
33	0,36822219	0,41105891	0,359887275	-	0,16693667	0,31715808
34	0,22468247	0,34019553	0,323129769	-	0,21053924	0,40434987
35	0,34240935	0,27840371	0,303403162	-	0,24550822	-
36	0,27459166	-	0,317392617	-	0,26925235	-
37	0,22770313	-	0,284553944	-	0,29089213	-
38	0,16329337	-	0,108553965	-	0,07803712	-
39	0,30719825	-	0,384285894	-	0,16037907	-
40	0,34661433	-	0,363249974	-	0,36851982	-
41	0,39541612	-	0,365301298	-	0,22035954	-
42	0,23827406	-	0,301297219	-	0,26387016	-

i	*fold 1*		*fold 2*		*fold 3*	
	α Level 1	*α Level 2*	*α Level 1*	*α Level 2*	*α Level 1*	*α Level 2*
43	0,4166	-	0,394510858	-	0,34674663	-
44	0,37626378	-	0,346989997	-	0,27695197	-
45	0,26869628	-	0,232297517	-	0,37636708	-
46	0,12631535	-	0,159170036	-	0,20918659	-
47	0,22245079	-	0,224936708	-	0,19379147	-
48	0,32088395	-	0,356344697	-	0,34671584	-
49	0,34324694	-	0,328002193	-	0,26902408	-
50	0,26197875	-	0,274722999	-	0,36488421	-

- **Nilai Bias**

fold 1		*fold 2*		*fold 3*	
bias Level 1	*bias Level 2*	*bias Level 1*	*bias Level 2*	*bias Level 1*	*bias Level 2*
-0,091191566	-0,0738024	-0,091191816	-0,287123426	0,0309873	-0,145125862

- **Nilai *f(x)***

fold 1					*fold 2*					*fold 3*				
No	*f(x) Level 1*	*f(x) Level 2*	kelas prediksi	kelas aktual	No	*f(x) Level 1*	*f(x) Level 2*	kelas prediksi	kelas aktual	No	*f(x) Level 1*	*f(x) Level 2*	kelas prediksi	kelas aktual
139	-0,99663	0,40681	2	2	156	-0,10442	-0,28542	3	3	158	-0,35786	-0,61156	3	3
2	1,01626	-0,07420	1	1	20	0,67831	-0,20417	1	1	10	0,29605	-0,03191	1	1
174	-0,12251	-0,10437	3	3	17	0,83187	-0,24774	1	1	8	1,28899	-0,13413	1	1

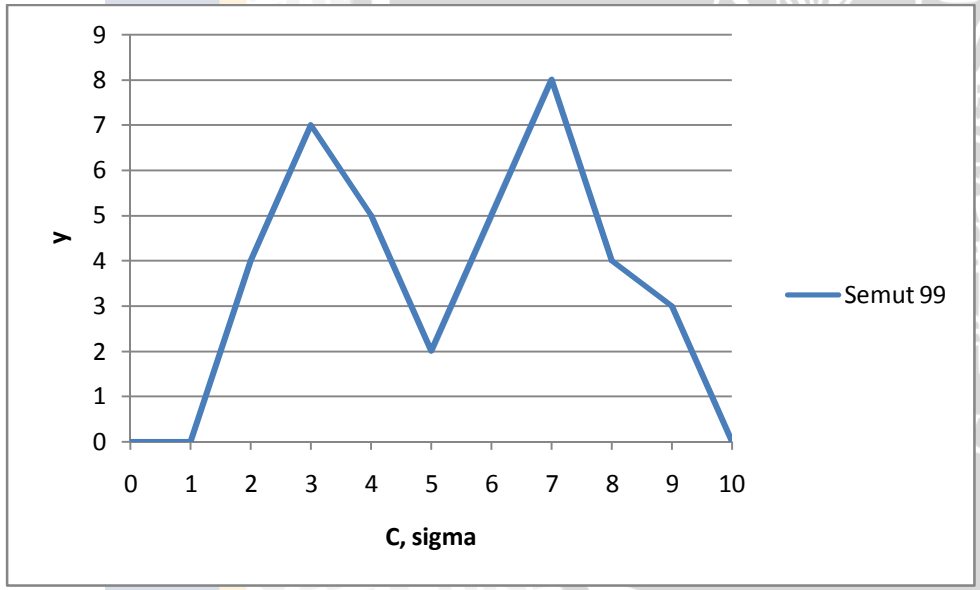
fold 1					*fold 2*					*fold 3*				
No	*f(x) Level 1*	*f(x) Level 2*	kelas prediksi	kelas aktual	No	*f(x) Level 1*	*f(x) Level 2*	kelas prediksi	kelas aktual	No	*f(x) Level 1*	*f(x) Level 2*	kelas prediksi	kelas aktual
15	0,42768	0,47375	1	1	163	-0,71703	-0,96534	3	3	176	-0,66397	0,28769	2	3
142	-1,14474	0,75402	2	2	11	0,85626	-0,10661	1	1	7	0,65401	-0,00641	1	1
159	-0,32439	-0,19363	3	3	162	-0,73943	-0,97926	3	3	171	-0,29196	-0,51639	3	3
24	1,02769	0,26035	1	1	177	-0,70743	-0,46609	3	3	137	-0,69481	0,84982	2	2
132	-0,19376	0,77815	2	2	122	-0,90592	0,62723	2	2	161	-0,05522	-0,08392	3	3
5	-0,14310	0,12853	2	1	126	-0,75070	0,86524	2	2	135	-0,36995	0,61975	2	2
16	0,71413	-0,07536	1	1	22	0,84421	-0,28034	1	1	160	-0,36314	0,38290	2	3
134	-0,17609	-0,18393	3	2	175	-0,47582	0,50073	2	3	172	-0,21267	-0,42287	3	3
164	-0,35572	-0,02355	3	3	124	-0,47516	0,20610	2	2	167	-0,30116	-0,52025	3	3
128	-0,58201	-0,18984	3	2	125	-1,08739	0,87225	2	2	23	1,61581	-0,09639	1	1
121	-0,79397	0,48920	2	2	120	-0,96770	0,93050	2	2	127	-0,33118	0,37666	2	2
180	-0,83705	-0,79815	3	3	18	0,44282	0,27266	1	1	170	-0,03182	-0,21419	3	3
119	-0,49416	0,17200	2	2	140	-0,72255	0,44705	2	2	12	0,28008	-0,14337	1	1
13	1,14574	0,16297	1	1	165	-0,60663	-0,78656	3	3	157	-0,33037	-0,55256	3	3
138	-0,89998	0,41013	2	2	166	-0,09362	-0,28972	3	3	14	0,14175	-0,13430	1	1
4	0,68474	0,07631	1	1	9	0,78026	0,25897	1	1	25	0,31056	-0,01928	1	1
179	-0,09121	-0,07382	3	3	169	-0,08825	0,14031	2	3	3	0,93409	-0,12558	1	1
21	0,07911	-0,15962	1	1	141	-0,61408	0,26417	2	2	123	-0,91413	0,91561	2	2
131	-0,97660	0,76628	2	2	129	-0,78545	0,76684	2	2	130	-0,62400	0,13629	2	2
6	0,75757	-0,06993	1	1	173	-0,14119	-0,34043	3	3	168	-0,14520	-0,35571	3	3
143	-0,08985	-0,07296	3	2	133	-0,44712	0,04726	2	2	19	0,29080	-0,11403	1	1

fold 1					*fold 2*					*fold 3*				
No	*f(x) Level 1*	*f(x) Level 2*	kelas prediksi	kelas aktual	No	*f(x) Level 1*	*f(x) Level 2*	kelas prediksi	kelas aktual	No	*f(x) Level 1*	*f(x) Level 2*	kelas prediksi	kelas aktual
1	0,54047	-0,07317	1	1	136	-0,09563	-0,28620	3	2	178	-0,23215	-0,48293	3	3

Pengujian Jumlah Semut

- **Nilai Parameter Hasil ACO**

Semut	C	sigma	Akurasi Rata-rata
99	0,4752	58,43	0,906666667



- **Nilai *α***

i	*fold 1*		*fold 2*		*fold 3*	
	α Level 1	*α Level 2*	*α Level 1*	*α Level 2*	*α Level 1*	*α Level 2*
1	0,14865767	0,15571674	0,02615294	0,13006355	0,06272927	0,22447374
2	0,24849282	0,4752	0,4752	0,37100534	0,4752	0,4017655
3	0,03126166	0,4752	0,225398402	0,22810767	0,2527275	0,28718476
4	0,12940326	0,22787214	0,349327903	0,24551493	0,30129517	0,30481847
5	0,0718487	0,4752	0,238483036	0,26489137	0,27385718	0,28291776
6	0,35707149	0,21543803	0,33429314	0,09605993	0,20991123	0,15512973
7	0,33032037	0,4752	0,186340934	0,26880062	0,12214131	0,27884499
8	0,22750931	0,39574596	0,365301801	0,11013156	0,32821709	0,13823052
9	0,06863017	0,45323076	0,125248625	0,11119952	0,11572153	0,23979281
10	0,34631593	0,18506453	0,071404585	0,160029	0,09532692	0,21632292
11	0,39136637	0,2715235	0,39916081	0,41104084	0,42012363	0,38929761
12	0,32471093	0,3322647	0,177333409	0,27379142	0,24367773	0,28859626
13	0,31010981	0,20185057	0,215771141	0,4519864	0,22747748	0,41024843
14	0,0486824	0,4752	0,102965501	0,21373244	0,0552969	0,24858827
15	0,14350451	0,35031681	0,279385068	0,37547523	0,28739872	0,42511894
16	0,23137312	0,21204895	0,419745459	0,22907048	0,31205811	0,26914537
17	0,36677809	0,32197304	0,086203327	0,3087213	0,10942699	0,37162324
18	0,41297892	0,21199662	0,128853386	0,16605418	0,13459328	0,14934518
19	0,02796042	0,45846123	0,400609957	0,4752	0,40309445	0,4752
20	0,27666713	0,41207927	0,369404662	0,434939	0,35124335	0,4752
21	0,15365222	0,28068064	0,181872032	0,22466896	0,17570839	0,22964138

i	*fold 1*		*fold 2*		*fold 3*	
	α Level 1	*α Level 2*	*α Level 1*	*α Level 2*	*α Level 1*	*α Level 2*
22	0,25231756	0,38795445	0,24360994	0,29818063	0,13577743	0,4752
23	0,16427281	0,44458577	0,088018868	0,43237461	0,08473465	0,22337427
24	0,3541381	0,17728853	0,396958141	0,08764715	0,39024168	0,4752
25	0,04997963	0,25203141	0	0,13249126	0	0,38888923
26	0,19556128	0,30741555	0,220392841	0,23729543	0,15184915	0,42935065
27	0,17328807	0,45952165	0,149476256	0,37210769	0,15391408	0,17012508
28	0,12249879	0,16943991	0,137167002	0,11633446	0,12659341	0,25994143
29	0,20899546	0,33527448	0,212199903	0,21089794	0,15637897	0,28910052
30	0,04697244	0,17534396	0,086175273	0,0850628	0,10261416	0,22361956
31	0,12186173	0,27621052	0,161570371	0,2115691	0,39933576	0,4752
32	0,06249312	0,20880082	0,081682254	0,17833623	0,35589841	0,33801287
33	0	0,19376991	0	0,17849055	0,26207265	0,21545643
34	0,30926965	-	0,319378768	-	0,09553963	0,28169354
35	0,46050896	-	0,4752	-	0,27538469	-
36	0,36000921	-	0,396984971	-	0,37223308	-
37	0,22158982	-	0,15791712	-	0,26870568	-
38	0,44413423	-	0,447897764	-	0,28441635	-
39	0,21241706	-	0,224390012	-	0,01761364	-
40	0,19326121	-	0,144935806	-	0,04170368	-
41	0,29782803	-	0,323484555	-	0,26919154	-
42	0,31380954	-	0,32181418	-	0,34538719	-
43	0,0799506	-	0,085423862	-	0,4752	-

i	*fold 1*		*fold 2*		*fold 3*	
	α Level 1	*α Level 2*	*α Level 1*	*α Level 2*	*α Level 1*	*α Level 2*
44	0,44087043	-	0,425285503	-	0,10863455	-
45	0,25192451	-	0,31303726	-	0,26677413	-
46	0,01728579	-	0	-	0,15451228	-
47	0,08878342	-	0,124309112	-	0,21091537	-
48	0,22556055	-	0,191801747	-	0,16790062	-
49	0,274327	-	0,286009164	-	0,42769566	-
50	0,17410349	-	0,158083175	-	0,013563	-

- **Nilai Bias**

fold 1		*fold 2*		*fold 3*	
bias Level 1	*bias Level 2*	*bias Level 1*	*bias Level 2*	*bias Level 1*	*bias Level 2*
0,250783645	-0,40128345	-0,004430031	-0,199997153	0,261083556	-0,126789983

- **Nilai *f(x)***

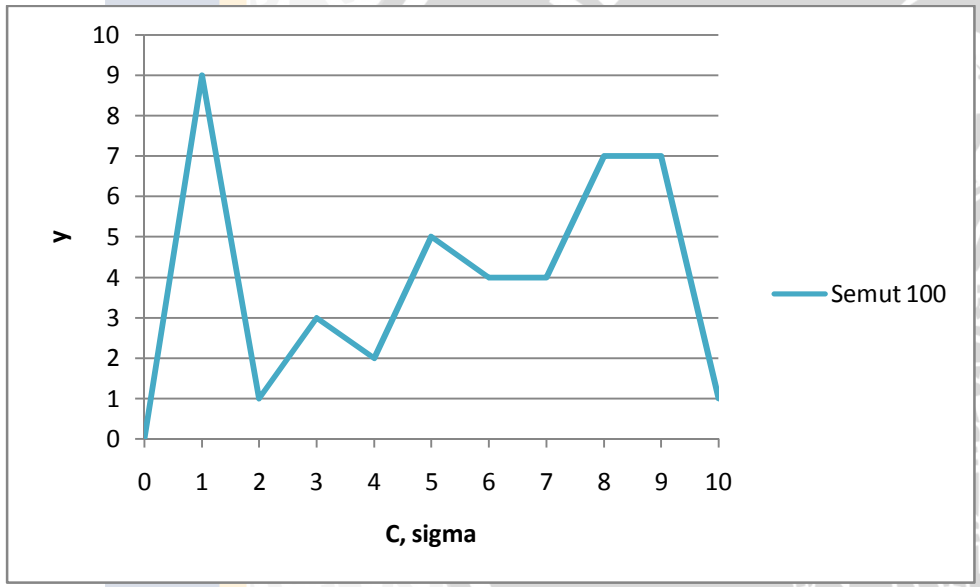
fold 1					*fold 2*					*fold 3*				
No	*f(x) Level 1*	*f(x) Level 2*	kelas prediksi	kelas aktual	*No*	*f(x) Level 1*	*f(x) Level 2*	kelas prediksi	kelas aktual	*No*	*f(x) Level 1*	*f(x) Level 2*	kelas prediksi	kelas aktual
178	-0,93852	-1,34504	3	3	171	-0,98779	-1,42228	3	3	12	1,03308	0,06504	1	1
21	0,22501	-0,03237	1	1	160	-0,54330	0,18966	2	3	122	-0,84835	1,05383	2	2
4	0,80755	0,42350	1	1	176	-1,26955	0,15713	2	3	164	-0,56655	-0,04650	3	3
136	0,02287	-0,34487	1	2	125	-1,02835	0,77579	2	2	156	-0,35359	-0,00208	3	3
173	-0,12318	-0,92666	3	3	8	1,25072	0,42669	1	1	142	-1,00996	0,77252	2	2

fold 1					*fold 2*					*fold 3*				
No	*f(x) Level 1*	*f(x) Level 2*	kelas prediksi	kelas aktual	*No*	*f(x) Level 1*	*f(x) Level 2*	kelas prediksi	kelas aktual	*No*	*f(x) Level 1*	*f(x) Level 2*	kelas prediksi	kelas aktual
119	-0,24061	0,23212	2	2	25	0,29403	0,79218	1	1	130	-0,62203	0,29758	2	2
133	-0,55826	0,23966	2	2	175	-0,35205	0,92743	2	3	1	1,45495	0,04384	1	1
135	-0,35797	0,84960	2	2	131	-0,66726	0,90007	2	2	177	-1,18941	-0,09015	3	3
141	-1,23683	0,30268	2	2	22	1,25570	0,24184	1	1	137	-0,45581	1,11534	2	2
163	-0,89441	-1,49976	3	3	134	-0,15340	-0,19084	3	2	5	0,06561	0,74853	1	1
132	-0,42632	0,76155	2	2	179	-0,05062	-0,24551	3	3	127	-0,07969	0,60335	2	2
162	-0,57381	-1,31431	3	3	174	-0,26717	-0,24635	3	3	170	-0,46783	-0,61595	3	3
11	0,68788	0,39839	1	1	129	-0,52253	1,01623	2	2	10	0,17928	0,21205	1	1
159	-0,83500	-0,59507	3	3	165	-1,16594	-1,04551	3	3	128	-0,80916	0,11269	2	2
166	-0,02397	-0,76514	3	3	121	-1,04132	0,48916	2	2	167	-0,50698	-1,16258	3	3
161	-0,06955	-0,36006	3	3	120	-0,76069	0,92089	2	2	124	-0,21128	0,32849	2	2
123	-1,29603	0,55673	2	2	9	0,44259	0,93401	1	1	126	-0,40633	1,06663	2	2
17	1,04585	0,13719	1	1	169	-0,14133	0,73306	2	3	6	1,43730	0,15005	1	1
143	0,22948	-0,30907	1	2	139	-1,27658	0,31974	2	2	15	0,38120	0,76043	1	1
7	0,18566	0,46648	1	1	13	0,69117	0,85053	1	1	19	0,81591	0,35346	1	1
2	0,81204	0,16834	1	1	168	-0,80173	-1,09804	3	3	158	-0,99346	-0,69985	3	3
140	-0,55611	0,37039	2	2	20	0,84915	0,52378	1	1	3	1,30831	0,38582	1	1
16	1,20441	-0,09892	1	1	23	1,01685	0,71360	1	1	157	-0,35483	-0,94279	3	3
18	0,21859	0,66891	1	1	14	0,59252	0,08913	1	1	24	0,80550	0,71226	1	1
180	-1,26223	-0,84600	3	3	138	-1,21868	0,42523	2	2	172	-0,49582	-1,07574	3	3

Pengujian Jumlah Iterasi

- **Nilai Parameter Hasil ACO**

Semut	C	sigma	Akurasi Rata-rata
100	9,1325	44,771	0,92



- **Nilai *α***

i	*fold 1*		*fold 2*		*fold 3*	
	α Level 1	*α Level 2*	*α Level 1*	*α Level 2*	*α Level 1*	*α Level 2*
1	0,0770381	0,31122603	0,313334106	0,16775829	0,29441047	0,18289061
2	0,24340463	0,19747441	0,505584166	0,21378826	0,50203703	0,15775576
3	0,39578527	0,43048739	0,210264509	0,29610533	0,15062833	0,31477637
4	0,09648904	0,54011185	0,309476501	0,27581909	0,21860628	0,30352716
5	0,50059686	0,18707357	0,188891839	0,77397298	0,29657964	0,7274717
6	0,08003882	0,21393202	0,178972648	0,33953942	0,21018961	0,4314453
7	0,10620492	0,15277095	0,300724788	0,15791118	0,28091563	0,15072644
8	0,21185079	0,24053212	0,238296209	0,3056188	0,23557571	0,32111304
9	0,13122261	0,12086268	0,19615353	0,22015649	0,13972468	0,24556052
10	0,24454187	0,40446996	0,295721081	0,21006626	0,30382416	0,17578231
11	0,13561187	0,15885998	0,37936244	0,33478891	0,26805766	0,3337535
12	0,29499141	0,54382064	0,050410146	0,23275372	0,03915913	0,2061449
13	0,51156107	0,26607429	0,320393562	0,3875641	0,31367863	0,40444263
14	0,1168082	0,17639973	0,229251697	0,34679831	0,33772039	0,33875242
15	0,24418128	0,19544402	0,225706654	0,18449644	0,19893546	0,17158984
16	0,31173367	0,49296764	0,372284883	0,24815468	0,36308575	0,22082876
17	0,28798683	0,39431418	0,363609005	0,22397084	0,34221633	0,21411791
18	0,32000174	0,41280977	0,210291918	0,4679905	0,28382435	0,50207071
19	0,45038476	0,29550332	0,145372522	0,25421102	0,10972174	0,30093705
20	0,36972277	0,12714851	0,069552859	0,15489948	0,07365324	0,21245568
21	0,07592671	0,32425036	0,122054309	0,45077152	0,24973019	0,41500305

i	*fold 1*		*fold 2*		*fold 3*	
	α Level 1	*α Level 2*	*α Level 1*	*α Level 2*	*α Level 1*	*α Level 2*
22	0,14063894	0,28670426	0,239330946	0,2257906	0,21133807	0,63643197
23	0,21919667	0,32237296	0,018550849	0,31381031	0,02235116	0,29430192
24	0,38117177	0,23672704	0,110157711	0,39073116	0,18953447	0,24113805
25	0,30073508	0,25691358	0,194481911	0,21432022	0,2066978	0,22224915
26	0,13362549	0,525051	0,124958721	0,6374422	0,12553584	0,20573074
27	0,10149631	0,41259718	0,114512605	0,3036256	0,16661765	0,13302128
28	0,19198249	0,40404934	0,203710901	0,42620857	0,37573862	0,41221302
29	0,3257847	0,29334857	0,349909191	0,25819557	0,20238447	0,15680086
30	0,27100359	0,34125047	0,236580192	0,25380961	0,34249073	0,47884839
31	0,15363072	0,3687508	0,208297781	0,48482429	0,12286144	0,18117636
32	0,08513414	0,13256325	0,075040538	0,14856671	0,172222	0,26221178
33	0,10045373	-	0,087208615	-	0,20963754	0,15454879
34	0,19316658	-	0,191967052	-	0,14868769	0,5433912
35	0,18143651	-	0,200735503	-	0,18290455	0,39599097
36	0,11313288	-	0,113692534	-	0,10469822	0,46571999
37	0,30193741	-	0,273752013	-	0,29338779	-
38	0,54176851	-	0,501981089	-	0,41305226	-
39	0,01706471	-	0,112158841	-	0,17458128	-
40	0,264441	-	0,211053721	-	0,33064969	-
41	0,15450032	-	0,054309422	-	0,22209012	-
42	0,12153499	-	0,028921608	-	0,3517337	-
43	0,29752772	-	0,318459659	-	0,40670213	-

i	*fold 1*		*fold 2*		*fold 3*	
	α Level 1	*α Level 2*	*α Level 1*	*α Level 2*	*α Level 1*	*α Level 2*
44	0,2345589	-	0,225413739	-	0,29350158	-
45	0,31999567	-	0,403592475	-	0,51098542	-
46	0,0148339	-	0,012002063	-	0,10955076	-
47	0,27630062	-	0,301390371	-	0,10546279	-
48	0,28413988	-	0,308085805	-	0,28290072	-
49	0,01990125	-	0,073084639	-	0,37068271	-
50	0,28177917	-	0,238209263	-	0,31928727	-

- **Nilai Bias**

fold 1		*fold 2*		*fold 3*	
bias Level 1	*bias Level 2*	*bias Level 1*	*bias Level 2*	*bias Level 1*	*bias Level 2*
-0,106434503	0,03687713	0,010226903	-0,595312082	0,046327833	-0,332364899

- **Nilai *f(x)***

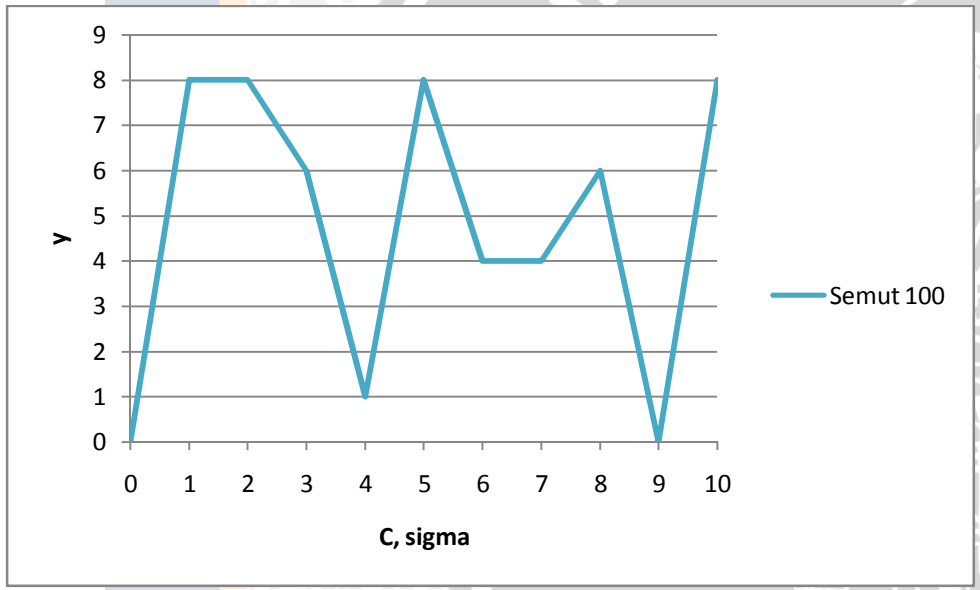
fold 1					*fold 2*					*fold 3*				
No	*f(x) Level 1*	*f(x) Level 2*	kelas prediksi	kelas aktual	No	*f(x) Level 1*	*f(x) Level 2*	kelas prediksi	kelas aktual	No	*f(x) Level 1*	*f(x) Level 2*	kelas prediksi	kelas aktual
135	-0,45249	1,26297	2	2	121	-1,06525	0,15180	2	2	17	1,24760	-0,28454	1	1
21	-0,16029	0,25276	2	1	131	-0,75484	0,24825	2	2	141	-0,98462	0,16593	2	2
171	-0,87204	-0,90689	3	3	143	-0,00377	-0,54981	3	2	13	0,87012	0,04494	1	1
124	-0,38394	0,03910	2	2	23	1,26825	-0,24187	1	1	18	0,41061	0,35560	1	1
20	0,82044	0,05998	1	1	169	-0,04321	-0,02521	3	3	129	-0,52762	0,52607	2	2

fold 1					*fold 2*					*fold 3*				
No	*f(x) Level 1*	*f(x) Level 2*	kelas prediksi	kelas aktual	No	*f(x) Level 1*	*f(x) Level 2*	kelas prediksi	kelas aktual	No	*f(x) Level 1*	*f(x) Level 2*	kelas prediksi	kelas aktual
133	-0,72353	0,53920	2	2	158	-0,82649	-1,38025	3	3	159	-0,77850	-0,43545	3	3
175	-0,46839	0,95122	2	3	16	1,18144	-0,54764	1	1	8	1,38738	-0,22482	1	1
128	-0,86589	0,08923	2	2	168	-0,41208	-1,09510	3	3	6	1,25744	-0,30642	1	1
120	-0,73770	0,87539	2	2	178	-0,75938	-1,50969	3	3	11	1,04708	-0,16010	1	1
173	-0,32184	-0,21966	3	3	172	-0,66757	-1,33646	3	3	140	-0,57996	0,31069	2	2
127	-0,22298	0,42235	2	2	163	-0,80911	-1,56015	3	3	1	1,21726	-0,32253	1	1
123	-1,17414	0,72851	2	2	156	-0,25560	-0,54466	3	3	170	-0,41984	-0,81650	3	3
166	-0,20930	-0,07858	3	3	10	0,45996	-0,29238	1	1	5	0,34206	0,07570	1	1
24	0,84428	0,66572	1	1	22	1,14942	-0,52027	1	1	180	-1,03719	-0,78653	3	3
162	-0,67226	-0,58049	3	3	19	0,65626	-0,28596	1	1	167	-0,79426	-1,30493	3	3
14	0,26884	0,15042	1	1	126	-0,58101	0,39544	2	2	176	-1,12003	0,10077	2	3
136	-0,21711	0,02972	2	2	134	-0,08454	-0,64998	3	2	139	-1,17745	0,26951	2	2
4	0,80203	0,67004	1	1	25	0,34666	-0,11084	1	1	7	0,39245	0,06993	1	1
130	-0,80789	-0,02492	3	2	132	-0,12471	0,33475	2	2	12	0,71678	-0,23711	1	1
125	-1,05064	0,87325	2	2	15	0,38938	0,14972	1	1	161	-0,10168	-0,28873	3	3
2	1,06221	0,12037	1	1	165	-0,79600	-1,25281	3	3	138	-1,10594	0,30115	2	2
157	-0,63740	-0,53112	3	3	122	-0,96537	0,34958	2	2	119	-0,23264	0,16356	2	2
142	-1,24472	0,66584	2	2	160	-0,44278	-0,11024	3	3	9	0,57058	0,13649	1	1
3	1,07457	0,31674	1	1	179	-0,00026	-0,60626	3	3	177	-1,02903	-0,43079	3	3
164	-0,67634	-0,27216	3	3	174	-0,16689	-0,65384	3	3	137	-0,54100	0,58403	2	2

Pengujian Panjang *fold*

- **Nilai Parameter Hasil ACO**

Semut	C	sigma	Akurasi Rata-rata
100	8,8618	44,608	0,92



- **Nilai *α***

i	*fold 1*		*fold 2*		*fold 3*	
	α Level 1	*α Level 2*	*α Level 1*	*α Level 2*	*α Level 1*	*α Level 2*
1	0,07727502	0,3109517	0,312871553	0,16820326	0,294001396	0,18331557
2	0,24287961	0,19729448	0,505040903	0,21463201	0,501517477	0,15848126
3	0,39545816	0,43043777	0,211073216	0,29622503	0,151265538	0,31510657
4	0,09597594	0,53908449	0,309183613	0,27612341	0,218431089	0,30390087
5	0,50033749	0,18717432	0,18893988	0,77329649	0,296521731	0,72685659
6	0,08122796	0,21482006	0,17948774	0,33940121	0,210734995	0,43156188
7	0,1067586	0,15329432	0,300262935	0,1577366	0,280601749	0,15040438
8	0,21278252	0,24142075	0,23880845	0,30629716	0,236152495	0,32176086
9	0,13224826	0,12152187	0,195639873	0,22031834	0,139261447	0,24575791
10	0,24517577	0,40443321	0,296202768	0,20976858	0,304268186	0,17533351
11	0,13618638	0,1587236	0,379096937	0,33535891	0,267662114	0,33430167
12	0,29595703	0,5434645	0,050393939	0,23316503	0,039375303	0,20677838
13	0,51098174	0,26647982	0,320807421	0,387827	0,314039685	0,40460381
14	0,11747757	0,17650005	0,228716948	0,34690202	0,337304551	0,33881808
15	0,24435863	0,1951384	0,225934202	0,18429196	0,19929572	0,1712875
16	0,31115265	0,49288315	0,372501143	0,24855009	0,363347208	0,22140441
17	0,28819234	0,39428532	0,363782608	0,22369248	0,342467501	0,21367545
18	0,31989044	0,41297251	0,210375459	0,46765147	0,283971732	0,5017625
19	0,4502221	0,2952122	0,145393906	0,25431984	0,110078912	0,30098457
20	0,36954657	0,1269161	0,069096271	0,15516504	0,073462431	0,21253248
21	0,07698261	0,32440285	0,12181201	0,45055201	0,249490372	0,41512082

i	*fold 1*		*fold 2*		*fold 3*	
	α Level 1	*α Level 2*	*α Level 1*	*α Level 2*	*α Level 1*	*α Level 2*
22	0,14038849	0,2866679	0,239557243	0,2257984	0,211664489	0,63574417
23	0,2199896	0,32294752	0,018315327	0,31449726	0,022535761	0,29441808
24	0,38108442	0,23650326	0,110588098	0,39048851	0,189864415	0,24196387
25	0,30102397	0,25783223	0,195429976	0,21499516	0,20786918	0,22268616
26	0,1338103	0,52493427	0,125270407	0,63743133	0,126047807	0,20632922
27	0,10180289	0,41260416	0,114781195	0,30346842	0,166260711	0,13343911
28	0,19147376	0,40406175	0,203064865	0,42591148	0,375607527	0,41206972
29	0,32562125	0,29285486	0,349600949	0,25826193	0,201826404	0,15681794
30	0,27027759	0,34146551	0,235889667	0,25386748	0,342255365	0,47881271
31	0,15482618	0,3686417	0,209437145	0,48442797	0,124521065	0,18130042
32	0,08536926	0,13233048	0,075323167	0,14875762	0,172862362	0,26237578
33	0,10092038	-	0,087865096	-	0,210641909	0,15440706
34	0,19298982	-	0,191749627	-	0,150023275	0,54315399
35	0,18161363	-	0,200743382	-	0,183559587	0,39604711
36	0,11396519	-	0,114706964	-	0,105380133	0,46544803
37	0,30265938	-	0,27490764	-	0,294456072	-
38	0,54159802	-	0,50194765	-	0,412651132	-
39	0,01818478	-	0,113916806	-	0,175375549	-
40	0,26492675	-	0,211501272	-	0,330522332	-
41	0,15507495	-	0,054685553	-	0,221621708	-
42	0,12183352	-	0,029058108	-	0,35167109	-
43	0,29733382	-	0,318152641	-	0,406331305	-

i	*fold 1*		*fold 2*		*fold 3*	
	α Level 1	*α Level 2*	*α Level 1*	*α Level 2*	*α Level 1*	*α Level 2*
44	0,23539749	-	0,226248201	-	0,293224801	-
45	0,32020451	-	0,403419904	-	0,510915676	-
46	0,01492002	-	0,012123866	-	0,110873792	-
47	0,27666605	-	0,301570038	-	0,105523737	-
48	0,28377661	-	0,307619686	-	0,283382964	-
49	0,02107418	-	0,07462476	-	0,370642963	-
50	0,28107796	-	0,237550803	-	0,319382513	-

- **Nilai Bias**

fold 1		*fold 2*		*fold 3*	
bias Level 1	*bias Level 2*	*bias Level 1*	*bias Level 2*	*bias Level 1*	*bias Level 2*
-0,106667085	0,03750904	0,009576377	-0,59369422	0,045582044	-0,331582423

- **Nilai *f(x)***

fold 1					*fold 2*					*fold 3*				
No	*f(x) Level 1*	*f(x) Level 2*	kelas prediksi	kelas aktual	No	*f(x) Level 1*	*f(x) Level 2*	kelas prediksi	kelas aktual	No	*f(x) Level 1*	*f(x) Level 2*	kelas prediksi	kelas aktual
135	-0,45456	1,26287	2	2	121	-1,06616	0,15421	2	2	17	1,24648	-0,28544	1	1
21	-0,15891	0,25241	2	1	131	-0,75616	0,24928	2	2	141	-0,98247	0,16731	2	2
171	-0,87187	-0,90385	3	3	143	-0,00433	-0,54889	3	2	13	0,87226	0,04341	1	1
124	-0,38409	0,03763	2	2	23	1,27101	-0,24379	1	1	18	0,41142	0,35485	1	1
20	0,82041	0,05827	1	1	169	-0,04376	-0,02423	3	3	129	-0,52927	0,52615	2	2

fold 1					*fold 2*					*fold 3*				
No	*f(x) Level 1*	*f(x) Level 2*	kelas prediksi	kelas aktual	No	*f(x) Level 1*	*f(x) Level 2*	kelas prediksi	kelas aktual	No	*f(x) Level 1*	*f(x) Level 2*	kelas prediksi	kelas aktual
133	-0,72292	0,53910	2	2	158	-0,82237	-1,37709	3	3	159	-0,77575	-0,43429	3	3
175	-0,46967	0,95135	2	3	16	1,17959	-0,54759	1	1	8	1,38667	-0,22570	1	1
128	-0,86535	0,08962	2	2	168	-0,40916	-1,08995	3	3	6	1,25566	-0,30649	1	1
120	-0,73912	0,87660	2	2	178	-0,75564	-1,50470	3	3	11	1,04777	-0,16207	1	1
173	-0,32060	-0,21659	3	3	172	-0,66601	-1,33250	3	3	140	-0,58115	0,31089	2	2
127	-0,22316	0,42122	2	2	163	-0,80731	-1,55606	3	3	1	1,21417	-0,32229	1	1
123	-1,17479	0,73047	2	2	156	-0,25283	-0,54295	3	3	170	-0,41728	-0,81316	3	3
166	-0,20793	-0,07595	3	3	10	0,46080	-0,29242	1	1	5	0,34049	0,07378	1	1
24	0,84639	0,66350	1	1	22	1,14717	-0,52037	1	1	180	-1,03460	-0,78647	3	3
162	-0,67168	-0,57809	3	3	19	0,65528	-0,28700	1	1	167	-0,79409	-1,30209	3	3
14	0,26805	0,14956	1	1	126	-0,58208	0,39675	2	2	176	-1,11816	0,10278	2	3
136	-0,21600	0,03037	2	2	134	-0,08454	-0,64920	3	2	139	-1,17610	0,27199	2	2
4	0,80177	0,66646	1	1	25	0,34685	-0,11233	1	1	7	0,39301	0,06889	1	1
130	-0,80691	-0,02500	3	2	132	-0,12432	0,33571	2	2	12	0,71402	-0,23787	1	1
125	-1,05220	0,87512	2	2	15	0,39093	0,14900	1	1	161	-0,10240	-0,28755	3	3
2	1,06352	0,11857	1	1	165	-0,79332	-1,25068	3	3	138	-1,10504	0,30270	2	2
157	-0,63673	-0,52887	3	3	122	-0,96545	0,35189	2	2	119	-0,23347	0,16297	2	2
142	-1,24561	0,66797	2	2	160	-0,44246	-0,10761	3	3	9	0,57213	0,13513	1	1
3	1,07351	0,31413	1	1	179	-0,00064	-0,60435	3	3	177	-1,02548	-0,42994	3	3
164	-0,67322	-0,27045	3	3	174	-0,16612	-0,65241	3	3	137	-0,54248	0,58445	2	2