IMPLEMENTASI ALGORITME *SUPPORT VECTOR MACHINE* (SVM) UNTUK KLASIFIKASI PENYAKIT DENGAN GEJALA DEMAM

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:
Nurul Ihsani Fadilah
NIM: 135150200111036

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

PENGESAHAN

IMPLEMENTASI ALGORITME *SUPPORT VECTOR MACHINE* (SVM) UNTUK KLASIFIKASI PENYAKIT DENGAN GEJALA DEMAM

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh :
Nurul Ihsani Fadilah
NIM: 135150200111036

Skripsi ini telah diuji dan dinyatakan lulus pada
9 juli 2018
Telah diperiksa dan disetujui oleh :

Dosen Pembimbing I	Dosen Pembimbing II
Bayu Rahayudi, S.T, M.T	M. Tanzil Furqon, S.Kom, M.CompSc
NIP: 19740712 200604 1 001	NIP: 19820930 200801 1 004

Mengetahui
Ketua Jurusan Teknik Informatika

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

PERNYATAAN ORISINALITAS

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

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

Malang, 30 Mei 2018

Nurul Ihsani Fadilah

NIM: 135150200111036

KATA PENGANTAR

Puji dan syukur kehadirat Allah SWT atas segala limpahan karunia, rahmat, dan pertolonganNya sehingga skripsi dengan judul "Implementasi Algoritma *Support Vector Machine* (SVM) Untuk Klasifikasi Penyakit Dengan Gejala Demam" dapat terselesaikan.

Dalam proses penyelesaian skripsi ini banyak sekali dukungan serta bantuan yang penulis dapatkan dari banyak pihak. Karena itu penulis ingin mengucapkan rasa hormat dan terimakasih kepada :

1. Bapak Bayu Rahayudi, S.T, M.T dan Bapak M. Tanzil Furqon, S.Kom, M.CompSc selaku pembimbing I dan pembimbing II yang telah memberikan segala bentuk bimbingan dan nasehat dengan sabar sehingga penulis dapat menyelesaikan skripsi ini.
2. Bapak Tri Astoto Kurniawan, S.T, M.T, Ph.D selaku Ketua Jurusan Teknik Informatika dan Bapak Agus Wahyu Widodo, S.T, M.Cs selaku ketua Program Studi Teknik Informatika serta seluruh Bapak dan Ibu dosen yang telah tulus memberikan ilmu selama penulis menempuh endidikan di Fakultas Ilmu Komputer Universitas Brawijaya.
3. Kedua Orangtua saya atas segala pengorbanan dan cinta yang telah diberikan, nasehat yang selalu menguatkan serta doa yang tidak pernah habis demi terselesaikannya skripsi ini. Terimakasih Ummi Darmawati dan Abi Mansur yang selalu memberikan apapun yang terbaik untuk anaknya.
4. Adik-adik spesial yang menjadi motivasi besar dalam menyelesaikan skripsi ini, kepada Muhammad Hidayat, Afif Hudzaifah, Nurul Izzah, Muhammad Adib Al-Banna, Hilmi Zuhdi Al-Faiz, Nurul Tazkiyatul Ummah, Syamil Dhiyaul Haq dan Afnan Faris Ayyasi.
5. Seluruh sahabat yang selalu membersamai selama proses belajar di FILKOM, UB dan juga Malang yaitu Hatin, Rika, Dela, Caca, Cika, Nyau, pejuang label merah perpustakaan, dan seluruh sahabat dan teman perjuangan.

Penulis menyadari bahwa dalam penulisan skripsi ini pasti memiliki banyak kesalahan dan kekurangan. Oleh karena itu penulis sangat menerima segala masukan berupa kritik dan saran yang dapat membangun dan memperbaiki. Semoga dengan selesainya skripsi ini dapat memberikan banyak manfaat dan kebaikan bagi semua pihak yang menggunakannya.

Malang, 30 Mei 2018

Penulis

aniynurihsaniy@gmail.com

ABSTRAK

Penyakit menular pada manusia memiliki salah satu gejala umum yaitu gejala demam. Dalam beberapa kasus, terdapat gejala demam yang menular pada tubuh manusia melalui media vektor penyakit yang disebabkan oleh serangga atau disebut juga dengan istilah penularan penyakit melalui media *Arthropod-borne disease*. Terdapat tiga penyakit dengan gejala demam yang penularannya terjadi melalui media *Arthropod-borne disease* yaitu Demam Berdarah, Malaria dan Demam Tifoid. Penyakit tersebut memiliki gejala klinis yang hampir sama sehingga cukup sulit melakukan diagnosis penyakit yang diderita oleh pasien. Dengan jumlah penderita yang besar dan resiko kematian yang tinggi pada penyakit ini, perlu adanya sebuah sistem yang dapat membedakan ketiga penyakit ini dengan cepat dan tepat. Untuk memecahkan permasalah tersebut, dibuat sebuah sistem yang dapat melakukan klasifikasi penyakit dengan gejala demam menggunakan algoritme *Support Vector Machine* (SVM). Penelitian ini menggunakan 130 dataset yang memiliki 15 parameter. Dataset dibagi menjadi data latih dan data uji dengan menggunakan metode *K-Fold Cross Validation,* dengan k=10. Sebelum mendapatkan hasil klasifikasi penyakit dengan gejala demam, dilakukan proses *training* dan *testing* dengan melakukan proses perhitungan yaitu perhitungan kernel dengan menggunakan kernel *polynomial,* perhitungan matriks Hessian, perhitungan nilai Ei, perhitungan nilai *delta alpha*, perhitungan nilai *alpha*, perhitungan nilai bobot, perhitungan bias, perhitungan nilai f(x) level 1, dan perhitungan nilai f(x) level 2. Hasil akhir dari implementasi algoritma *Support Vector Machine* untuk peyakit dengan gejala demam adalah akurasi dari ketepatan sistem dalam mengklasifikasi kelas demam berdarah, kelas malaria dan kelas tifoid sehingga didapatkan hasil akurasi terbaik dengan menggunakan metode *k-fold cross validation*, dengan k=10, pembagian rasio data = 90%:10%, dan parameter yang digunakan adalah *lamda* = 0.5, *gamma* = 0. 01, C (*complexity*) = 1, *epsilon* = 0.0001, iterasi maksimum = 20, sehingga rata-rata akurasi yang didapatkan yaitu 99.23%.

Kata Kunci : klasifikasi, demam, *Support Vector Machine* (SVM), *K-fold Cross Validation,* kernel *polynomial*

ABSTRACT

Infectious disease in humans have one of the general indications, that is Fever. In some cases, there are symptoms of fever that are transmitted to the human body through a vector-borne disease caused vector or also called disease transmission through the Arthropod-borne disease medium. There are three diseases with symptoms of fever that transmission of disease occurs by the media Arthropod-borne disease, such as Dengue Fever, Malaria, and Typhoid. The disease has almost the same clinical symptoms, so it is difficult to make a diagnosis of the disease suffered by the patient. Because of a large number of patient and a high risk of death in this disease, need a system that can distinguish these three diseases quickly and precisely. To solve the problem, the system is needed to classify the disease with fever symptoms using the Support Vector Machine (SVM) algorithm. This research uses 130 datasets that have 15 parameters. The dataset is divided into train data and test data by using K-Fold Cross Validation method, with k=10. Before getting the classification of the disease with symptoms of fever, the process of training and *testing* must be done by calculating using *polynomial* kernel, Hessian matrix calculation, Ei value calculation, *delta alpha* value calculation, *alpha* value calculation, weight value calculation, bias value calculation, calculating the value of f(x) in level 1, and calculating the value of (x) in level2. The final result from Support Vector Machine algorithm implementation for disease with symptoms of fever is the accuracy of the system capabilities in classifying dengue fever class, malaria class, and typhoid class. So, the best average value of accuracy in this implementation is 99.23%, using k-fold cross validation, with k=10, division of data ratio=90%:10%, and the parameters used are lamda=0.5, gamma=0.01, C(Complexity)=1, epsilon=0.0001, maximum iteration=20.

Keywords**:** classification, fever, Support Vector Machine (SVM), K-fold Cross Validation, *polynomial* kernel

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR LAMPIRAN

BAB 1 PENDAHULUAN

Pada bab ini akan dijelaskan tentang latar belakang masalah, rumusan masalah, tujuan penelitian, manfaat penelitian, batasan masalah, dan sistematika penulisan.

1.1 Latar belakang

Penyakit menular adalah salah satu masalah yang sering muncul di Indonesia sebagai pengaruh dari rendahnya kesadaran masyarakat dalam menjaga kesehatan dan kebersihan lingkungan. Salah satu gejala umum dari penyakit menular adalah gejala demam yang menjangkit tubuh manusia melalui media *Arthropod-borne disease*. Gejala demam tersebut merupakan demam yang menular pada tubuh manusia melalui media vektor penyakit yang disebabkan oleh serangga. Beberapa penyakit demam yang disebabkan oleh *vector-borne diseases* adalah demam berdarah yang penularannya dapat melalui gigitan nyamuk *Aedes aegypti*, Malaria dengan penularan penyakit melalui parasit malaria pada nyamuk *Anopheles* dan Demam Tifoid yang ditularkan secara mekanis oleh lalat rumah (Chandra, 2006).

Pusat Data dan Informasi Kementerian Kesehatan Republik Indonesia (2015) menyebutkan bahwa terdapat peningkatan jumlah penderita demam berdarah dan peningkatan kasus kematian, yang mana penderita demam berdarah pada tahun 2014 sebesar 100.347 jiwa naik menjadi 126.675 jiwa dan kasus kematian dari 907 jiwa naik menjadi 1.229 jiwa.

Penderita demam tifoid disebutkan sebanyak 22 juta per tahun di dunia dengan angka kematian sebesar 600.000 orang (Purba, et al., 2016). Di Indonesia penderita demam tifoid setiap tahunnya cenderung mengalami peningkatan rata-rata 800 jiwa per 100.000 orang (Depkes RI, 2013 disitasi dalam Paputungan, 2016, p.267). Dengan begitu sangat jelas bahwa ketiga penyakit ini sangat berpotensi dalam menyebabkan kesakitan dan juga kematian di wilayah negara-negara tropis (Beriajaya, 2009).

Data perkiraan terbaru oleh *World Health Organization* menyebutkan 198 juta kasus malaria terjadi secara global pada tahun 2013 dan menyebabkan 584.000 kematian (World Malaria Report, 2014). Kementerian Kesehatan Republik Indonesia (2016) juga menyebutkan bahwa jumlah angka yang terkena penyakit malaria secara nasional pada tahun 2015 adalah 0,85 per 1000 penduduk. Penyakit Malaria disertai gejala-gejala yaitu demam dengan fluktuasi suhu tubuh yang teratur atau serangan demam dengan interval tertentu (*parokisme*), lemah, sakit kepala, mual, tidak nafsu makan, muntah, kurang darah, adanya pigmen dalam jaringan, dan juga pembesaran limpa. Diagnosis dengan dasar manifesti atau gejala klinis pada penyakit demam berdarah, malaria dan demam tifoid seringkali tidak khas dan sulit untuk dibedakan (Arsin, 2012).

Ketiga penyakit tersebut memiliki penderita dengan angka yang relatif tinggi serta memililki resiko kematian yang cukup besar, sehingga dibutuhkan

otomatisasi sistem klasifikasi yang dapat mengurangi kesalahan diagnosa awal penyakit tersebut dengan lebih cepat dan tepat. Metode klasifikasi telah banyak berkembang dalam dunia teknologi, salah satunya adalah klasifikasi dengan menggunakan metode *Support Vector Machine*. Dengan menggunakan metode ini, beberapa data yang didapatkan dari hasil pemeriksaan pasien demam, diklasifikasikan kedalam suatu label berdasarkan kemiripan dari data tersebut, sehingga beberapa data yang memiliki karakteristik yang sama dikelompokan dalam satu label, sebaliknya yang memiliki karakteristik berbeda dikelompokan dalam label yang lain yang memiliki karakteristik yang sama.

Beberape penelitian dalam mendiagnosis penyakit Demam Berdarah, Tifoid dan Malaria telah dilakukan, salah satunya dilakukan oleh Ramadhani, Wardhani dan Nugroho (2012) dengan menggunakan metode Bayesian Network dan memberikan kesamaan akurasi sebesar 78% setelah dibandingkan dengan diagnosis yang diberikan oleh dokter berdasarkan analisis dengan data sampel. Pada penelitian berikutnya dilakukan oleh Shofia, Putri dan Arwan (2017) dengan menggunakan metode *K-nearest Neighbor - Certainty Factor,* pada 143 data latih. Penelitian tersebut menunjukkan bahwa akurasi yang didapatkan sama dengan ketika menggunakan metode *K-nearest Neighbor*, yaitu sebesar 84.79%.

Berbagai metode klasifikasi telah digunakan dalam menyelesaikan masalah dalam kehidupan ini. Tidak hanya metode *Bayesian Network* dan *K-nearest Neighbor* yang dapat digunakan. *Support Vector Machine* juga merupakan algoritme unggulan yang dapat digunakan dalam permasalahan klasifikasi, seperti yang ditunjukkan pada penelitian yang dilakukan oleh Darsyah (2013) dalam mengklasifikasi kanker payudara jinak dan ganas. Penelitian tersebut memberikan output berupa ketepatan akurasi dari klasifikasi y dengan tingkat akurasi metode *Support Vector Machine* sebesar 99% dengan menggunakan kernel RBF dan 96 % dengan menggunakan kernel *polynomial*. Selain itu, pada penelitian yang dilakukan oleh Nugraheini dan Mutijarsa (2016) memaparkan tentang analisis perbandingan dari metode KNN, SVM, dan metode *Random Forests* untuk klasifikasi ekspresi muka. Penelitian tersebut menunjukkan bahwa algoritme SVM memiliki akurasi yang tinggi pada data *training* sebesar 165 dan data *testing* sebesar 82 data.

Berdasarkan beberapa penelitian diatas, Support Vector Machine terbukti memiliki nilai akurasi yang cukup tinggi dalam mengklasifikasi sebuah permasalahan yang ada, sehingga penulis memutuskan melakukan penelitian dengan judul "Implementasi metode Support Vector Machine untuk klasifikasi penyakit dengan gejala demam". Diharapkan dengan penerapan metode Support Vector Machine untuk klasifikasi penyakit dengan gejala demam dapat meminimalisasi kesalahan dalam melakukan diagnosis, khususnya dapat membedakan penyakit Demam berdarah, Tifoid dan Malaria.

1.2 Rumusan masalah

Berdasarkan uraian pada latar belakang tersebut, maka dapat dirumuskan beberapa permasalahan sebagai berikut:

1. Bagaimana hasil implementasi algoritme *Support Vector Machine* dalam menyelesaikan masalah klasifikasi penyakit dengan gejala demam.
2. Bagaimana hasil akurasi dari implementasi algoritme *Support Vector Machine* dalam menyelesaikan masalah klasifikasi penyakit dengan gejala demam.

1.3 Tujuan

Adapun tujuan yang ingin dicapai dari penelitian ini adalah:

1. Menerapkan algoritme *Support Vector Machine* dalam menyelesaikan masalah klasifikasi penyakit dengan gejala demam.
2. Mendapatkan hasil akurasi dari implementasi algoritme *Support Vector Machine* dalam menyelesaikan masalah klasifikasi penyakit dengan gejala demam.

1.4 Manfaat

Adapun manfaat yang diharapkan dari hasil penelitian ini adalah sebagai berikut:

1. Dapat memahami dan menerapkan algoritme *Support Vector Machine* dalam menyelesaikan masalah klasifikasi penyakit dengan gejala demam.
2. Dapat membantu meminimalisasi kesalahan dalam mendiagnosa penyakit dengan gejala demam khususnya membedakan penyakit Demam Berdarah, Malaria dan Tifoid.

1.5 Batasan masalah

Penelitian ini memiliki beberapa batasan, untuk menghindari adanya perluasan masalah. Sehingga batasan-batasan tersebut dipaparkan sebagai berikut:

1. Data yang digunakan dalam penelitian ini bersumber dari data penelitian sebelumnya yang dilakukan oleh Fakihatin Wafiyah, yang berasal dari Rumah Sakit Selasih Kabupaten Riau.
2. Kriteria yang digunakan pada penelitian ini dalam membahas perhitungan klasifikasi penyakit dengan gejala demam adalah demam intermitten (putus-putus), demam menggigil, pembesaran hati, pembesaran limpa, nyeri perut, bintik merah (*ptekie*) pada kulit, demam terutama malam hari, demam lebih 1 minggu, sakit kepala, sakit tulang dan sendi, mual dan muntah, mencret dan susah BAB (konstipasi), lidah kotor (*coated tongue*), *bradikardi relatif*, kulit lembab/keringat.
3. Sistem ini hanya menghasilkan 3 kelompok penyakit yaitu Demam Berdarah, Malaria, dan Tifoid. Jumlah seluruh data yang digunakan terdiri dari 130 Data.
4. Sistem yang dibangun bertujuan sebagai pendukung dalam memberikan diagnosis awal, bukan sebagai penentu keputusan akhir.

1.6 Sistematika pembahasan

Berikut adalah sistematika yang digunakan dalam menyusun laporan penelitian:

BAB 1: Pendahuan

Bab ini berisi tentang latar belakang, rumusan masalah, batasan masalah, tujuan, manfaat penelitian, serta sistematika penulisan dari laporan terkait implementasi algoritme *Support Vector Machine* untuk klasifikasi peyakit dengan gejala demam.

BAB 2: Tinjauan Kepustakaan

Bab ini berisi tentang tinjauan penelitian terdahulu serta dasar teori yang dibutuhkan dalam penjelasan-penjelasan mengenai implementasi algoritme *Support Vector Machine* untuk klasifikasi penyakit dengan gejala demam.

BAB 3: Metodologi Penelitian

Bab ini berisi tentang penjelasan metode atau sejumlah langkah yang akan diimplementasikan dalam penelitian meliputi studi literatur, pengumpulan data, preproses data, analisis kebutuhan, perancangan sistem, implementasi sistem, pengujian sistem dan analisis sistem.

BAB 4: Analisis dan Perancangan

Bab ini berisi tentang proses analisis kebutuhan serta perancangan sistem yang digunakan dalam implemenasi algoritme SVM untuk klasifikasi penyakit dengan gejala demam.

BAB 5: Implementasi

Bab ini berisi tentang penjelasan proses implementasi, batasan-batasan implementasi, serta penerapan sistem berdasarkan analisis dan perancangan yang sebelumnya telah dibahas pada bab analisis dan perancangan.

BAB 6: Pengujian dan Analisis

Bab ini berisi tentang pengujian sistem dan analisis dari hasil pengujian serta hasil akurasi implementasi algoritme *Support Vector Machine* untuk klasifikasi penyakit dengan gejala demam.

BAB 7: Penutup

Bab ini berisi kesimpulan dan saran dari hasil penelitian algoritme *Support Vector Machine* untuk pengelompokan penyakit dengan gejala demam.

BAB 2 LANDASAN KEPUSTAKAAN

Pada bab ini berisi penjabaran tentang uraian yang berisi pembahasan penelitian terdahulu, serta terdapat teori umum, konsep, model dan metode yang berhubungan dalam penyakit dengan gejala demam, klasifikasi, dan algoritme *Support Vector Machine* (SVM).

2.1 Tinjauan Penelitian Terdahulu

Sebagai referensi dalam penelitian implementasi algoritme SVM untuk klasifikasi penyakit demam, berikut penjelasan penelitian-penelitian sebelumnya yang telah menggunakan metode SVM. Penelitian yang dilakukan oleh Luqman pada tahun 2016 menggunakan algoritme *Support Vecor Machine* (SVM) dalam pemilihan beasiswa. Penelitian ini menggunakan 272 data dengan 10 parameter (9 atribut predictor, 1 atribut hasil) dan menunjukkan bahwa model SVM menghasilkan tingkat akurasi 83.94%, sehingga dalam penelitian tersebut menyebutkan bahwa algoritme SVM adalah model algoritme yang dapat digunakan dengan baik.

Pada penelitian pertama tentang peramalan pada penyakit kanker oleh Ai, jia dan Xin (2016) mendapatkan tingkat akurasi yang berbeda pada fungsi kernel yang berbeda pula, yang mana untuk penggunanaan fungsi kernel linear mendapatkan akurasi sebesar 70%, untuk penggunaan fungsi kernel *polynomial* 65%, fungsi kernel radial basis 15% dan untuk penggunaan fungsi sigmoid mendapatkan akurasi sebesar 15%. Sehingga pada penelitian tersebut kita dapat melihat tingkat akurasi terbesar terdapat pada penggunaan fungsi kernel menggunakan fungsi linear.

Penggunaan algoritme SVM juga digunakan pada pengenalan karakter bahasa Oriya (bahasa di negara barat India) oleh Mohanty dan Bebartta (2011) dengan membandingkan nilai akurasi pada algoritme SVM dan KNN yang menghasilkan nilai 98.9% untuk SVM sedangkan KKN menghasilkan nilai akurasi sebesar 96.47%. Data ini menunjukkan bahwa SVM telah digunakan secara luas oleh banyak peneliti dan telah membuat kontribusi besar dalam menyelesaikan permasalahan secara efektif baik itu pada pembelajaran maupun penelitian terutama pada bidang pengenalan pola, klasifikasi pada artikel, pengenalan wajah, pengenalan suara pengenalan digital dan sebagainya. Algoritme SVM memiliki ketelitian yang lebih tinggi dari metode pembelajaran tradisional (Xu Xin Ai, Hu jia dan Lu Xin, 2016).

Penelitian ketiga dilakukan oleh Darsyah (2013), peneliti tersebut menggunakan algoritme SVM untuk mengklasifikasi kanker payudara jinak dan kanker payudara ganas. Hasil dari penelitian tersebut adalah nilai akurasi yang diberikan algoritme SVM dengan kernel yang berbeda. Pada penelitian menggunakan kernel RBF menghasilkan akurasi sebesar 99% sedangkan pada penelitian menggunakan kernel *Polynomial* menghasilkan akurasi sebesar 96%.

Pada penelitian berikutnya dilakukan oleh Ratna Astuti Nugraheini tentang analisis perbandingan dari KNN, SVM dan algoritme Random Forests dalam klasifikasi ekspresi wajah. Pada penggunaan data *training* sebanyak 165 dan data *testing* sebanyak 82, penelitian ini menghasilkan nilai akurasi sebesar 75.15% untuk K-Nearest Neighbir(KNN), 80% untuk SVM, dan 76.97 untuk algoritme random forest.

oleh karena itu berdasarkan penelitian-penelitian di atas, penulis mencoba mengimplementasikan algoritme *support vector machine* untuk klasifikasi penyakit demam. terdapat lima belas kriteria yang digunakan, yaitu demam intermitten (putus-putus), demam menggigil, pembesaran hati, pembesaran limpa, nyeri perut, bintik merah (ptekie) pada kulit, demam terutama malam hari, demam lebih 1 minggu, sakit kepala, sakit tulang dan sendi, mual dan muntah, mencret dan susah BAB (konstipasi), lidah kotor (coated tongue), bradikardi relatif, kulit lembab/keringat.

2.2 Demam

Menurut Dinarello & Gelfand (2005 disitasi dalam Utama dan Merati, 2015) demam adalah tingginya suhu tubuh dari suhu yang normal sehari-hari dan berhubungan dengan tingginya titik patokan suhu di hipotalamus. Disamping itu, Kanashiro & Zieve (2010 disitasi dalam Utama dan Merati, 2015) menjelaskan suhu tubuh normal memiliki ukuran standar sekitar 36,5-37,2 derajat Celsius, sehingga suhu tubuh yang termasuk demam adalah *rectal temperature* ≥ 38,0 derajat Celsius / oral *temperature* ≥ 37,5 derajat Celsius / *axillary temperature* ≥ 37,2 derajat Celsius.

Demam disebabkan oleh zat yang bernama pirogen. Terdapat dua bagian pirogen, yaitu pirogen eksogen dan pirogen endogen. Pirogen eksogen adalah pirogen yang berasala dari luar tubuh manusia, contohnya adalah produk mikroorganisme seperti toksin atau mikroorganisame seutuhnya. Sedangkan pirogen endogen merupakan pirogen yang berasal dari tubuh manusia, contohnya antara lain IL-1, IL-6, TNF-α dan IFN.

Dalal & Zhukovsky (2006 disitasi dalam Utama dan Merati, 2015) menyebutkan bahwa demam memiliki tiga fase, fase yang pertama adalah fase kedinginan, yaitu fase saat suhu tubuh meningkat dan ditandai dengan vasokonstriksi pembuluh darah serta aktifitas otot yang meningkat memproduksi panas sehingga mengakibatkan kedinginan dan menggigil. Fase kedua adalah demam, yaitu keseimbangan memproduksi panas dan dilain sisi kehilangan panas pada titik patokan yang mana suhu yang sebelumnya telah meningkat. Fase terakhir adalah kemerahan, yaitu penurunan suhu ditandai oleh vasodilatasi pembuluh darah serta munculnya keringat yang menghilangkan panas.

Terdapat tipe-tipe demam, yang disebutkan pada tabel di bawah ini:

Tabel 2. 1 Tipe-tipe demam

No	Jenis Demam	Penjelasan
1	Demam Septik	Suhu tubuh meningkat pada malam hari, berkeringat serta menggigil, dan pada siang hari kembali normal.
2	Demam hektik	Suhu badan berangsur-angsur naik ke ukuran yang tinggi sekali saat malah hari lalu kemudian kembali turun ke suhu semula.
3	Demam Remiten	Suhu badan dapat turun dalam sehari atau setiap harinya tetapi suhu tidak pernah mencapai tingkat yang normal.
4	Demam Intermiten	Suhu badan turun pada tingkat seperti biasa (normal) selama beberapa jam dalam satu harinya.
5	Demam Kontinyu	Suhu sepanjang hari bisa bervariasi dan variasi tidak berbeda lebih dari satu derajat.
6	Demam Siklik	Suhu badan beberapa hari mengalami kenaikan, dan kenaikan tersebut diikuti oleh masa bebas demam untuk beberapa hari saja dan kemudian kembali seperti suhu semula.

(Sumber : Utama dan Merati, 2015)

2.2.1 Demam Berdarah *Dangue*

Virus *Dangue* adalah bagian dari *arthropode-borne* virus, sekarang dikenal sebagai genus Flavivirus dan masuk dalam family Flaviviridae. Virus *Dangue* pada umumnya ditularkan dari gigitan nyamuk dari famili Stegomya, diantaranya *Aedes albopictus dan Aedes aegypti* (Suhardiono,2005). Nyamuk tersebut akan menularkan penyakit Demam Berdarah *Dangue* (DBD) atau *Dangue* Haemorrhagic Fever (DHF) pada manusia. Suhardiono (2005) menyebutkan bahwa penderita Demam diketahui melalui kriteria klinis atau diagnosis yaitu, muncul demam yang tinggi dan mendadak secara terus menerus selama kisaran dua sampai tujuh hari, pembesaran pada organ hati, terdapat pendarahan baik berupa bintik kecil berwarna merah berbentuk bulat pada kulit, mimisan, muntah darah, ataupun pendarahan pada saluran pencernaan. Demam Berdarah *Dangue* dibagi menjadi empat derajat penyakit, sebagai berikut :

1. Derajat ke-1

   Penderita merasakan demam yang datang secara mendadak, dalam kisaran waktu dua sampai tujuh hari, dengan gejala berupa manifesti pendarahan.

2. Derajat ke-2

   Penderita mengalami beberapa hal yang sama pada derajat ke-1 ditambah dengan pendarahan yang spontan pada kulit atau pendarahan pada daerah lain.

3. Derajat ke-3

   Penderita mengalami sirkulasi yang gagal, dengan ditandai nadi yang berdetak cepat tapi lemah, serta tekanan nadi menjadi turun dalam kisaran mmHg kurang dari 20, dan mengalami gelisah dengan kulit yang dingin. Penderita juga mengalami gejala tekanan darah rendah pada derajat ini.

4. Derajat ke-4

   Penderita mengalami renjatan atau kegagalan peredaran darah yang ditandai dengan nadi yang tidak dapat diraba serta tidak dapat mengukur berapa tekanan darah penderita tersebut.

2.2.2 Demam Malaria

Protozoa *Plasmodium* merupakan penyebab dari penyakit infeksi malaria, parasit *Plasmodium* ini akan ditransmisikan pada manusia lewat perantara dari nyamuk *Anopheles* betina, selanjutnya parasit tersebut akan berkembang biak pada sel dari darah manusia. Terdapat lima spesies dari parasit *Plasmodium* yang saat ini diketahui menjadi penyebab dari demam malaria, yaitu P. *knowlesi*, P. *falciparum,* P. *vivax,* P. *malariae* (Liwan,2015 dan Putra,2011). Dalam manifesti klinis yang dimiliki oleh penderita malaria, Liwan (2015) menyebutkan bahwa terdapat gejala utama dari malaria yaitu, demam tinggi yang memiliki sifat paroksismal dengan disertai menggigil, nyeri di bagian kepala, serta berkeringat. Kemudian gejala lainnya adalah sering merasa lelah, tidak nafsu makan, nyeri punggung, nyeri otot/pegal-pegal, pucat dan muntah. Pada pemeriksaan fisik untuk penderita malaria, putra (2011) menyebutkan beberapa tambahan gejala dari malaria, yaitu, demam, pucat, pembesaran limpa (splenomegali), pembesaran hati(hepatomegali).

Demam paroksisme pada penyakit malaria memiliki sebuah interval yang ditentukan dari waktu menghasilkan sizon matang dan infeksi dari spesies *plasmodium.* Terdapat 3 stadium demam peroksisme yang biasanya terjadi berurutan, sebagai berikut :

1. *Stadium Frigoris* (menggigil)

   Pada stadium awal ini, diawali dengan rasa dingin dan menggigil, berlangsung dalam waktu lima belas menit sampai sekitar satu jam. Penderita mengalami pucat pada bagian bibir, jari, dan kulit yang kering.

2. *Stadium acme* (pucak demam)

   Pada stadium pertengahan ini, dimulai dengan serangan demam. Penderita mungkin merasa kulit menjadi kering dan merasa panas yang terbakar, muka yang merah serta merasa mual dan sakit kepala. Selain itu, suhu badan

meningkat sampai 41 derajat Celcius, berlangsung selama dua sampai empat jam.

3. *Stadium sudoris* (berkeringat banyak, suhu turun)

   Pada stadium akhir, penderita mengeluarkan keringat yang sangat banyak, walaupun suhu dari badan penderita turun dan sampai menjadi normal. Stadium ini berlangsung sekitar dua sampai empat jam. Pada stadium ini dapat terjadi gangguan fungsi ginjal dan anuria.

2.2.3 Demam Tifoid

Salmonella enterica serovar typi atau disebut juga *S.typi* merupakan bakteri yang menyebabkan demam tifoid. *Salmonella enterica serovar typi* merupakan salah satu dari bakteri Gram Negatif yang memliki sifat patogen sehingga dapat menyebabkan bahaya bagi organ yang ditumpanginya. Salah satu faktor alami dan yang menjadi tempat menampung bakteri tersebut adalah manusia, yang mana penyakit demam tifoid tersebar karena penjagaan kebersihan dari setiap individu yang kurang baik, selain itu pengolahan air bersih, pengendalian limbah pemakaian kamar mandi dan pengawasan terhadap penyedia makanan yang kurang diperhatikan.

Cita (2011) menyebutkan bahwa penderita yang terkena demam tifoid akan mengalami gejala klinis yaitu, demam yang naik terus-terusan di minggu awal, kemudian di minggu kedua akan mengalami demam yang kontinyu atau disebut juga dengan istilah remiten. Pada beberapa penderita penyakit tifoid, demam bisa jadi muncul mendadak dan menjadi parah pada kisaran waktu satu sampai dua hari. Demam pada tifoid biasanya terjadi pada sore atau malah hari. Gejala klinis lainnya adalah sakit kepala, nyeri perut, diare (obstipas), nyeri otot (mialgia), anoreksi, muntah dan mual, lidah yang berslaput, pembesaran organ hati, dan pembesaran limfa. Pada penderita penyakit demam tifoid yang mengalami sakit dalam kisaran waktu 2 minggu atau lebih berpotensi memiliki gajala demam tifoid yang lebih kompleks (Nelwan, 2012). Gejala dari penderita demam tifoid yang mengalami komplikasi, yaitu, terjadi pendarahan pada *gastrointestnal* (saluran pada sistem pencernaan), reaktif hepatitis, perforasi (terjadi lubang) pada usus, *ensefalopati tifosa* (penyakit degeneratif pada otak), dan juga komplikasi hematologi.

2.3 Konsep Klasifikasi

Menurut Prasetyo (2014) klasifikasi adalah sebuah pekerjaan yang melakukan pembelajaran atau pelatihan kepada fungsi target f, fungsi tersebut berguna untuk memetakan setiap set data x ke salah satu kelas dari label kelas y. Dalam klasifikasi dibutuhkan sebuah data latih dan data uji. Data latih atau *training data* adalah data yang sebelumnya telah diketahui label dari kelasnya dan akan digunakan untuk menciptakan sebuah model klasifikator. Sedangkan data uji atau *testing data* adalah data yang dianggap belum diketahui label kelasnya dan akan di prediksi label kelasnya dengan menggunakan model klasifikator yang telah

dibangun. Sistem inilah yang disebut dengan *pembelajaran terbimbing (supervised learning).*

Algoritme klasifikasi dibagi menjadi dua macam berdasarkan cara pelatihan, yaitu, *eager learner* dan *lazy learner. Eager learner* merupakan kategori algoritme yang memiliki proses pelatihan yang lama sedangkan proses prediksinya cepat karena tidak menggunakan data latih ketika proses prediksi, sedangkan *lazy learner* merupakan algoritme yang memiliki proses pelatihan yang cepat dengan proses prediksi yang lebih lama karena menggunakan data latih ketika prediksi. Algoritme yang termasuk kategori *Eager Learner* diantaranya adalah *Bayesian*, *Decision Tree*, *Support Vector Machine*, *Artificial Neural Network* sedangkan yang termasuk kategori *lazy learner* adalah *K-Nearest Neighbor* (K-NN), regresi linear, *Fuzzy K-Nearest Neighbor* (FK-NN).

2.4 Algoritme *Support Vector Machine*

Support Vector Machine bermula dari teori pembelajaran statistika, yang mana perhitungan akhirnya memiliki hasil yang lebih baik daripada metode statistika yang lain. Pada proses pembangunan model klasifikasi di SVM, terdapat sejumlah data latih yang akan terpilih dalam proses pelatihan untuk dipelajari. Kemudian terdapat juga sebagian kecil dari data latih yang akan disimpan untuk digunakan pada proses prediksi. Sehingga pada proses iterasi pelatihan di SVM, tidak semua data latih akan terlibat. Poin inilah yang menjadi pembeda SVM dari algoritme yang lain dan menjadikan algoritme ini disebut dengan *Support Vector Machine*, yang mana terdapat sebuah *support vector* atau data-data yang berkontribusi pada kedua proses pelatihan dan prediksi tersebut (Prasetyo, 2014).

Pada penerapannya, SVM memiliki kemampuan yang baik dalam mengolah sebuah set data yang memiliki dimensi yang tinggi. Teknik SVM yang lain menggunakan penambahan teknik kernel dalam pemetaan sebuah set data dari dimensi asal kepada dimensi lain yang justru lebih tinggi. Algoritme SVM memaksimalkan batas nilai *hyperplane* dengan nilai dari margin yang maksimal sehingga generalisasi pada metode klasifikasi menjadi lebih baik. Inti dari proses pembelajaran atau pelatihan pada SVM adalah pencarian lokasi *hyperplane.* Garis *hyperplane* dapat ditemukan dengan menghitung jarak (*margin*) maksimal antara garis *hyperplane* dengan data yang paling dekat dengan *hyperplane* (*support vector*) pada masing-masing kelas.

SVM sebenarnya menggunakan *hyperplane* linear, yang mana *hyperplane* tersebut diaplikasikan pada pemisahan kelas suatu data dengan garis linear. Sedangkan pada praktiknya terdapat kelas suatu data yang tidak bisa dipisahkan dengan garis linear dan harus menggunakan tambahan fungsi kernel atau dapat disebut dengan *kernel trick*. Sehingga pada SVM dibedakan menjadi SVM linear dan SVM non Linear. SVM Nonlinear memerlukan pemetaan pada data yang awalnya berdimensi rendah, diubah menjadi dimensi yang lebih tinggi.

Dalam perhitungan SVM, pada sebuah data yang di nyatakan dengan x_i, y_i yang mana $i = 1,2, \dots, N, x_i = x_{i1},\ x_{i2}, \dots, x_{ik}$ adalah parameter dari data latih ke-

i untuk $y_i \in \{-1, +1\}$ menandakan label dari suatu kelas. Sebelum menemukan hasil klasifikasi dari tiap-tiap kelas, akan dilakukan proses *training* terlebih dahulu, proses ini bertujuan untuk menghitung nilai dari α_i yang dibutuhkan saat mencari nilai dari *w* dan *b*. Nilai *w* diartikan sebagai vektor bobot dan *b* diartikan sebagai nilai bias. Nilai dari α_i akan dihitung dengan menggunakan *algoritme sequential training SVM.*

Langkah awalnya adalah dengan menginisiasi nilai dari $\alpha_i = 0, \gamma, \lambda, C, \varepsilon$, *Iterasi Maximal*, dan menginisialisai parameter kernel apabila menggunakan SVM Nonlinear. γ merupakan *learning rate, dan* ε merupakan nilai *epsilon*. Setelah inisialisasi selasai, maka harus menghitung terlebih dahulu matriks dari D_{ij}, yang mana $i, j = 1, \dots, n$ (banyaknya *data training*). Berikut notasi dari perhitungan matrik D_{ij}:

$$D_{ij} = y_i y_j (K(x_i, x_j) + \lambda^2) \quad (2.1)$$

Kemudian, setelah perhitungan pada persamaan 2.1 selesai, dilakukan iterasi pada tiga perhitungan yaitu pada persamaan 2.2, 2.3, dan 2.4.

$$E_i = \sum_{j=1}^{n} \alpha_j D_{ij} \quad (2.2)$$

$$\delta\alpha_i = \min\{\max[\gamma(1 - E_i), \alpha_i], C - \alpha_i\} \quad (2.3)$$

$$\alpha_i = \alpha_i + \delta\alpha_i \quad (2.4)$$

Iterasi terus dilakukan, sampai kondisi yang ditetapkan menjadikan perhitungan tersebut berhenti dilakukan. Kondisi berhenti adalah saat memenuhi syarat yang mana $\max(|\delta\alpha_i|) < \varepsilon$ atau perhitungan tersebut dapat berhenti saat telah memenuhi *Iterasi Maximal.* Saat salah satu dari kedua syarat tersebut tidak dipenuhi maka perhitungan kembali pada persamaan 2.2, kemudian 2.3 dan 2.4. Setelah menyelesaikan perhitungan *sequential training SVM* dan mendapatkan nilai dari α_i, selanjutnya adalah menghitung nilai dari *w* dan *b* dengan menggunakan persamaan 2.5 dan 2.6.

$$w = \alpha_i y_i \phi(x_i) \quad (2.5)$$

$$b = -\frac{1}{2}[w.x^+ + w.x^-] \quad (2.6)$$

Dari persamaan yang ditunjukkan pada 2.5 dan 2.6 untuk menemukan nilai dari *b* dapat diturunkan menjadi persamaan 2.7, 2.8 dan 2.9.

$$b = -\frac{1}{2}[\sum_{i=1}^{m} \alpha_i y_i \mathrm{x}_i \,.\, x^+ + \sum_{i=1}^{m} \alpha_i y_i \mathrm{x}_i \,.\, x^-] \quad (2.7)$$

$$b = -\frac{1}{2}[\sum_{i=1}^{m} \alpha_i y_i \phi(\mathrm{x}_i)\ \phi(x^+) + \sum_{i=1}^{m} \alpha_i y_i \phi(\mathrm{x}_i)\ \phi(x^-)] \quad (2.8)$$

$$b = -\frac{1}{2}[\sum_{i=1}^{m} \alpha_i y_i K(\mathrm{x}_i\,, x^+) + \sum_{i=1}^{m} \alpha_i y_i K(\mathrm{x}_i\,, x^-)] \quad (2.9)$$

m pada persamaan diatas, merupakan jumlah dari support vektor atau titik suatu data yang memiliki α_i besar dari 0. Setelah mendapatkan nilai dari *b*, selanjutnya adalah mencari hasil klasifikasi dengan menggunakan fungsi klasifikasi yaitu *sign(f(x)) ,* Persamaan untuk mendapatkan hasil dari klasifikasi tersebut terdapat pada persamaan 2.10, 2.11, 2.12, dan 2.13.

$$f(x) = w.x + b \quad (2.10)$$

$$f(x) = \sum_{i=1}^{m} \alpha_i y_i \mathrm{x}_i \,.\, x + b \quad (2.11)$$

$$f(x) = \sum_{i=1}^{m} \alpha_i y_i \phi(\mathrm{x}_i)\ \phi(x) + b \quad (2.12)$$

$$f(x) = \sum_{i=1}^{m} \alpha_i y_i \mathrm{K}(\mathrm{x}_i, x) + b \quad (2.13)$$

Pada persamaan 2.10, 2.11, 2.12 dan 2.13 variabel *x* merupakan suatu data yang akan diklasifikasikan.

Pada SVM Nonlinear dibutuhkan fungsi kernel, fungsi kernel berguna untuk pemetaan fitur yang lama pada data ke fitur yang baru. Penggunaan fungsi kernel pada SVM Nonlinear dapat juga disebut *kernel trick. Kernel trick* ini digunakan pada permulaan proses perhitungan dari SVM Nonlinear. Di bawah ini merupakan beberapa pilihan dalam menentukan fungsi kernel yang akan digunakan pada perhitungan SVM Nonlinear:

1. Kernel Linear, definisi fungsi terdapat pada persamaan 2.14.

   $$K(x, y) = x.y \quad (2.14)$$

2. Kernel *Polynomial*, definisi fungsi terdapat pada persamaan 2.15.

   $$K(x, y) = (x.y + c)^d \quad (2.15)$$

3. Kernel Gaussian, definisi fungsi terdapat pada persamaan 2.16.

$$K(x,y) = exp\left(\frac{-\|x-y\|^2}{2.\sigma^2}\right) \quad (2.16)$$

4. Kernel Sigmoid (Tangen Hiperbolik), definisi fungsi terdapat pada persamaan berikut 2.17.

$$K(x,y) = \tanh(\sigma(x.y) + c) \quad (2.17)$$

Pemilihan fungsi kernel ini sangat penting, karena pemilihan kernel ini berpengaruh terhadap penentuan fitur baru yang mana akan berpengaruh juga pada *hyperplane* (fungsi klasifikasi) yang akan dicari.

2.4.2 SVM Multikelas

SVM merupakan algoritme yang hanya dapat mengklasifikasikan data menjadi dua kelas, tapi dengan pendekatan pada metode SVM Multikelas, SVM dapat mengklasifikasikan tiga atau lebih kelas yang dibutuhkan. Diantara pendekatan dari SVM Multikelas adalah *one-againts-all*. Pendekatan yang dilakukan pada metode *one-againts-all* adalah mendekomposisi sebuah permasalahan dari pemisahan kelas menjadi *K* biner, yang mana $y_i \in Y$, dan untuk permasalahan klasifikasi SVM dibentuk menjadi semua yang berada pada kelas y_i dianggap sebagai kelas positif sedangkan yang lainnya dianggap sebagai kelas negative. Dan untuk memisahkan kelas lain yang masih tergabung dengan kelas yang sebelumnya telah diklasifikasikan, maka akan dilakukan pengklasifikasian lagi sebanyak *K*. Berikut adalah gambaran penyelesaian klasifikasi dengan menggunakan metode *one-againts-all* untuk klasifikasi tiga kelas.

Tabel 2. 2 gambaran penyelesaian klasifikasi dengan *one-againts-all*

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

Sumber : Sembiring K (2007)

2.5 Akurasi

Dalam mengukur kinerja sebuah sistem klasifikasi digunakan matriks confusion, matrix confusion ini adalah cara umum untuk mengetahui seberapa baik sistem klasifikasi melakukan pengklasifikasian pada sebuah data dengan benar (prasetyo,2014). Matrix Confusion terbentuk dari sebuah tabel yang berisi catatan dari hasil yang di berikan sistem klasifikasi. Di bawah ini adalah matriks confusion untuk klasifikasi pada dua kelas:

Tabel 2. 3 Matrix confusion, klasifikasi pada dua kelas

f_{ij}		Kelas setelah di prediksi (*j*)	
		Kelas=1	*Kelas=0*
	Kelas=1	f_{11}	f_{10}

Kelas sebenarnya(*i*)	Kelas=0	f_{01}	f_{00}

Berdasarkan tabel diatas, dapat diketahui bahwa data yang diklasifikasikan dengan benar berjumlah sekian pada $f_{11} + f_{00}$ dan data yang diklasifikasikan dengan salah berjumlah sekian pada $f_{01} + f_{10}$. Dalam perhitungan matriks confusion dapat dihasilkan dua nilai, nilai tersebut adalah laju error dan akurasi. Nilai akurasi didapatkan dari jumlah suatu data yang berhasil diklasifikasi dengan benar, sedangkan laju error didapatkan dari jumlah suatu data yang tidak berhasil diklasifikasi dengan benar. Untuk menghitung akurasi dan laju error didapatkan dari fungsi perhitungan pada persamaan 2.18 dan persamaan 2.19.

$$Perhitungan\ akurasi = \frac{jumlah\ data\ yang\ diklasifikasi\ benar}{jumlah\ seluruh\ data\ yang\ diklasifikasi} \quad (2.18)$$

$$Perhitungan\ laju\ error = \frac{jumlah\ data\ yang\ diklasifikasi\ salah}{jumlah\ seluruh\ data\ yang\ diklasifikasi} \quad (2.19)$$

Secara umum model klasifikasi yang telah dibuat dapat mengklasifikasi secara benar apabila digunakan pada seluruh data latih, namun saat digunakan pada data uji kinerja dari model klasifikasi ini lah yang benar-benar dihitung kinerjanya apakah dapat mengklasifikasikan secara benar atau tidak.

BAB 3 METODOLOGI

Pada metodologi penelitian ini akan dibahas metode atau langkah-langkah yang akan digunakan dalam implementasi algoritme SVM ke dalam klasifikasi penyakit dengan gejala deman yang dibagi menjadi demam berdarah, demam tifoid dan demam malaria.

3.1 Teknik Pengumpulan Data

Dalam penelitian ini, peneliti menggunakan data sekunder yang mana data tersebut adalah data statistik pasien yang menderita penyakit dengan gejala demam. Data tersebut didapatkan dari penelitian yang telah dilakukan sebelumnya oleh Wafiyah (2017). Dari data pasien yang menderita penyakit dengan gejala demam, diperoleh jumlah data sebanyak 130 data. Terdapat pembagian kelas pada data tersebut, yaitu demam tifoid, demam malaria dan demam berdarah, masing masing data telah diklasifikasi dari pihak Rumah Sakit Umum Selasih di Riau berdasarkan bobot masing masing dari 15 gejala penyakit yang telah didefinisikan.

3.2 Algoritme yang Digunakan

Pada penelitian ini, penulis menerapkan algoritme *Support Vector Machine* (SVM) untuk klasifikasi pada penyakit dengan gejala demam. Klasifikasi penyakit dengan gejala demam menggunakan algoritme SVM, dilakukan karena SVM merupakan algoritme yang baik dalam melakukan pengelompokan kelas (Muis dan Affander, 2015). Algoritme SVM termasuk motode supervised learning, yang mana motode ini telah banyak digunakan dalam memberi solusi berupa global optimal dari permasalahan yang sering di temukan seperti klasifikasi dalam masalah kedokteran. Perhitungan dalam SVM dibagi menjadi proses *training* dan proses *testing*. Pada penelitian ini teradapat 100 data yang akan digunakan sebagai data latih (*training data*) pada proses *training* dan 33 data yang akan digunakan sebagai data uji (*testing* data) pada proses *testing*. Selain itu terdapat pula 15 gejala dari penyakit yang akan diklasifikasikan dalam setiap data yang akan digunakan sebagai parameter dalam perhitungan algoritme SVM. Jumlah kelas yang akan diklasifikasi menggunakan SVM Nonlinear sebanyak 3 kelas (kelas demam berdarah, demam tifoid dan demam berdarah), sehingga penelitian ini menggunakan Multiclass SVM dengan metode *one against all*.

Data yang digunakan dalam implementasi algoritme SVM, merupakan data nonlinear yang mana persebaran dari data tidak dapat dipisahkan hanya dengan garis yang linear. Oleh karena itu digunakan teknik algoritme SVM Nonlinear. Penggunaan algoritme dengan teknik SVM Nonlinear membutuhkan penambahan perhitungan kenel. Penambahan perhitungan kernel ini bertujuan untuk mentransformasi atau memberi pemetaan pada data yang awalnya berdimensi rendah, diubah menjadi dimensi yang lebih tinggi (prasetyo, 2014). Setelah perhitungan normalisasi dan perhitungan kernel dilakukan, perhitungan SVM Nonlinear dilanjutkan dengan menghitung algoritme *sequential training.*

Perhitungan algoritme sequential *training* bertujuan untuk mendapatkan nilai *alpha* yang akan digunakan pada proses perhitungan SVM Nonlinear yang selanjutnya. Kemudian proses perhitungan SVM Nonlinear dilanjutkan dengan mencari nilai bobot dan bilai bias yang akan dibutuhkan dalam perhitungan akhir untuk mendapatkan hasil klasifikasi. Dagram alir dari perhitungan SVM yang akan diimplementasikan pada penelitian ini terdapat pada gambar 3.2.

Gambar 3. 1 Diagram Alir Klasifikasi penyakit dengan gejala demam menggunakan SVM

3.3 Kebutuhan Sistem

Analisis kebutuhan berguna untuk mendapatkan apa saja yang diperlukan dalam pembuatan sistem ini. Kebutuhan-kebutuhan tersebut adalah sebagai berikut:

1. Kebutuhan Hardware, meliputi: komputer PC, RAM 4,00 GB dan monitor 14".
2. Kebutuhan Software, meliputi: sistem operasi Microsoft Windows 8, database server MySQL, Apache, menggunakan bahasa PHP.
3. Kebutuhan data, meliputi: data statistik penderita penyakit demam berdarah, demam tifoid, dan demam malaria.

3.4 Pengujian Algoritme

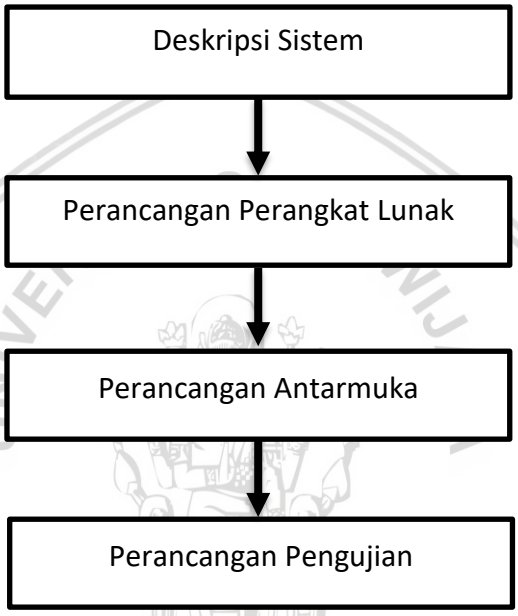
Untuk mendapatkan perbandingan antara data klasifikasi yang telah dilakukan oleh rumah sakit dengan hasil dari klasifikasi yang diselesaikan oleh sistem, maka dibutuhkan sebuah pengujian. Pada penelitian ini, pengujian akan dilakukan dalam beberapa jenis dan dilihat dari akurasi yang dihasilkan pada setiap pengujian. Pada setap pengujian akan dilakukan enam bentuk skenerio. Skenario pada pengujian pertama dilakukan untuk menguji pengaruh perbandingan jumlah data *training* dan data *testing* pada nilai dari akurasi yang didapatkan. Skenario pengujian kedua sampai keenam dilakukan untuk menguji pengaruh nilai *lamda* (λ), nilai *gamma* (γ), nilai *epsilon* (ε), nilai iterasi maksimum dan nilai C (*complexity*) pada proses *sequential training* untuk mencari nilai *alpha* (α).

3.5 Kesimpulan dan Saran

Pada tahap akhir penelitian ini, akan diberikan kesimpulan dari penelitian yang telah dilakukan. Kesimpulan didasarkan dari hasil yang diberikan sistem klasifikasi *Support Vector Machine* serta analisa dari pengujian yang dilakukan. Dalam penulisan kesimpulan ini, diharapkan dapat menjawab rumusan masalah yang sebelumnya telah diuraikan, berupa hasil implementasi algoritme Support Vector Machine dalam menyelasaikan masalah klasifikasi penyakit dengan gejala demam dan hasil akurasi dari implementasi algoritme SVM dalam menyelesaikan masalah klasifikasi penyakit dengan gejala demam.

BAB 4 PERANCANGAN

Pada bab 4 ini akan diuraikan perancangan yang bertujuan menggambarkan bagaimana detail dari rancangan sistem yang akan dibangun. Penjalasan pada bab ini akan dimulai dengan uraian tentang deskripsi dari sistem yang diangkat, kemudian diikuti penjelasan perancangan yang meliputi: perancangan perangkat lunak, perhitungan manualisasi, perancangan antarmuka, dan perancangan pengujian. Berikut diagram alir pada gambar 4.1, sebagai gambaran alur perancangan yang akan dilakukan pada bab ini.



Gambar 4. 1 Diagram Alir Alur Perancangan

4.1 Deskripsi Sistem

Pada penelitian ini, algoritme *Support Vector Machine* akan diimplementasikan untuk klasifikasi penyakit dengan gejala demam. Sistem ini akan membedakan penyakit demam kedalam tiga kelas, yaitu : demam bedarah, demam malaria, dan demam tifoid. Data yang akan diproses adalah data yang telah didapatkan dari penelitian sebelumnya, Wafiyah (2017). Proses inti dari sistem ini adalah proses *training* dan proses *testing*. Pada proses *training*, didahului dengan perhitungan kernel SVM. Data pada algoritme ini, menggunakan data yang nonlinear sehingga teknik SVM Nonlinear diimplementasikan dengan penambahan perhitungan kernel *polynomial*.

Proses *Training* dilakukan dengan metode *sequential training* untuk mendapatkan nilai *alpha* yang dibutuhan pada proses selanjutnya. Kemudian dilanjutkan pada proses *testing*. Proses *Testing* inilah yang akan memberikan

output berupa prediksi kelas dari setiap data *testing*. Ketiga kelas ini akan dibedakan sesuai dengan parameter atau data beberapa gejala yang terdapat pada ketiga penyakit dengan gejala demam tersebut. Gejala penyakit penelitian berjumlah 15 gejala dengan bobot yang telah ditentukan pada masing-masing gejala tersebut. Bobot dari 15 gejala dapat dilihat pada tabel 4.1.

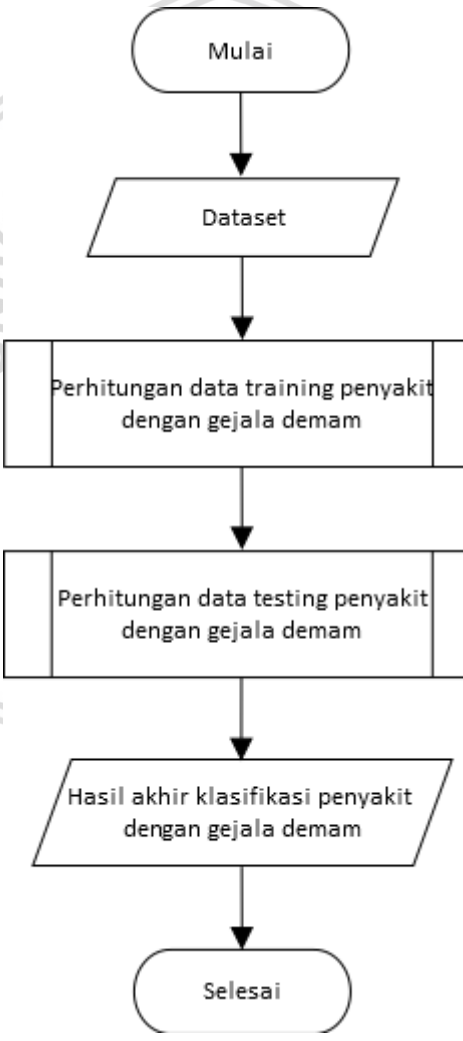
Tabel 4. 1 Gejala penyakit demam, keluhan serta bobot

NO	Gejala	Keluhan	Bobot
A1	Demam Intermitten	Ya	0.8
		Tidak	0
A2	Demam Menggigil	Berat	0.85
		Sedang	0.5
		Tidak	0
A3	Demam Terutama Malam Hari	Ya	0.75
		Tidak	0
A4	Lama Demam	> 7 Hari	0.75
		4 -7 Hari	0.5
		1 - 3 Hari	0.2
A5	Sakit Kepala	Berat	0.9
		Sedang	0.7
		Tidak	0
A6	Sakit Tulang dan Sendi	Berat	0.9
		Sedang	0.75
		Tidak	0
A7	Mual dan Muntah	Berat	0.85
		Sedang	0.6
		Tidak	0
A8	Mencret atau Kontisipasi	Ada	0.6
		Tidak	0.2
A9	Nyeri Perut	Berat	0.8
		Sedang	0.4
		Tidak	0
A10	Ptekie pada Kulit	Berat	1
		Sedang	0.7
		Tidak	0
A11	Lidah Kotor	Berat	0.85
		Sedang	0.4
		Tidak	0
A12	Bradikardi Relatif	Ya	0.75
		Tidak	0
A13	Pembesaran Hati	Ya	0.5
		Tidak	0
A14	Pembesaran Limpa	Ya	0.65

		Tidak	0
A15	Kulit Lembab / Keringat	Berat	0.85
		Sedang	0.4
		Tidak	0

4.2 Perancangan Perangkat Lunak

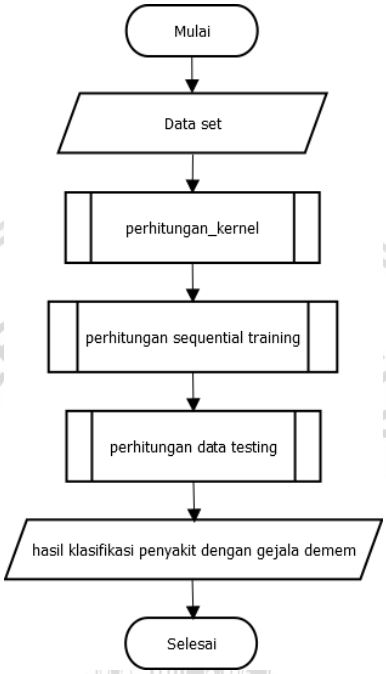
Pada perancangan perangkar lunak akan diberikan gambaran berupa diagram alir dari beberapa proses yang akan dilakukan dalam sistem ini. Secara garis besar, perancangan perangkat lunak pada sistem ini akan digambarkan pada diagram alir yang terdapat pada Gambar 4.2.



Gambar 4. 2 Diagram alir klasifikasi penyakit dengan gejala demam menggunakan SVM

4.2.1 Proses Perhitungan Algoritme SVM

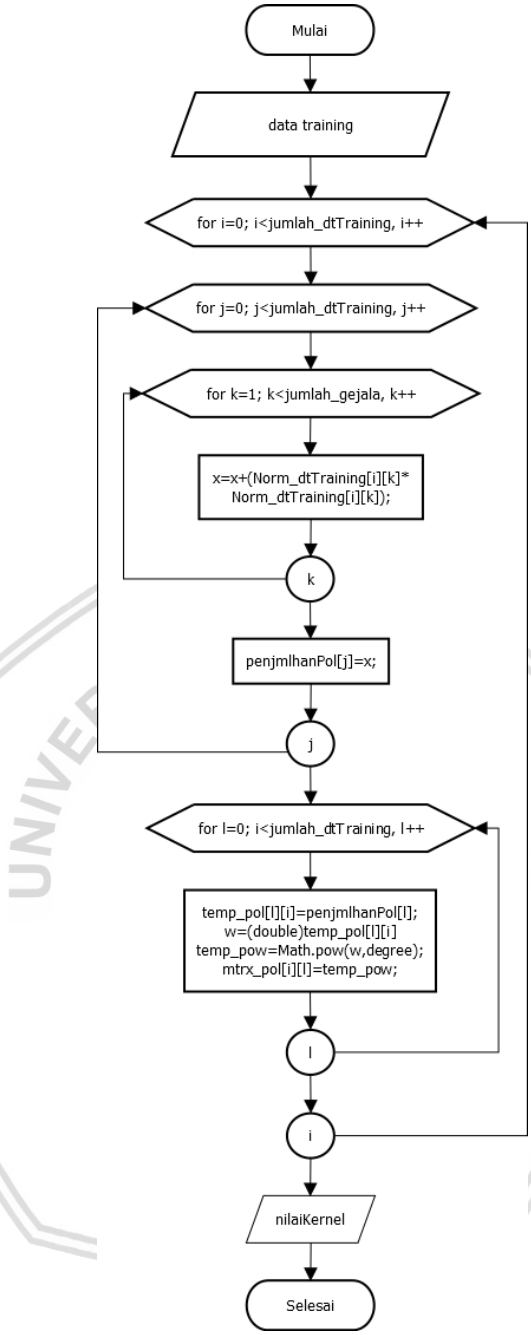
Proses perhitungan algoritme SVM, merupakan kumpulan dari beberapa bagian proses perhitungan yang terdapat di dalam algoritme SVM sendiri. Proses ini akan mengolah data set menjadi sebuah perhitungan akhir berupa prediksi data ke dalam sebuah kelas menjadi tiga klasifikasi kelas. Proses ini diawali dengan perhitungan kernel, lalu dilakukan tahap perhitungan *training* dan perhitungan *testing*. Proses perhitungan algorima SVM dapat dilihat pada Gambar 4.3.



Gambar 4. 3 Diagram alir proses perhitungan algoritme SVM

4.2.2 Proses Perhitungan Kernel

Proses perhitungan kernel dilakukan untuk melakukan pemetaan fitur yang lama pada data ke fitur yang baru. Perhitungan ini akan mengolah input berupa data set menjadi nilai kernel. Perulangan dilakukan sesuai dengan jumlah data *training* dan jumlah gejala pada data *training*, kemudian dilakukan perhitungan dengan menggunakan kernel *polynomial*. Proses perhitungan kernel yang lebih detail dapat dilihat pada Gambar 4.4.

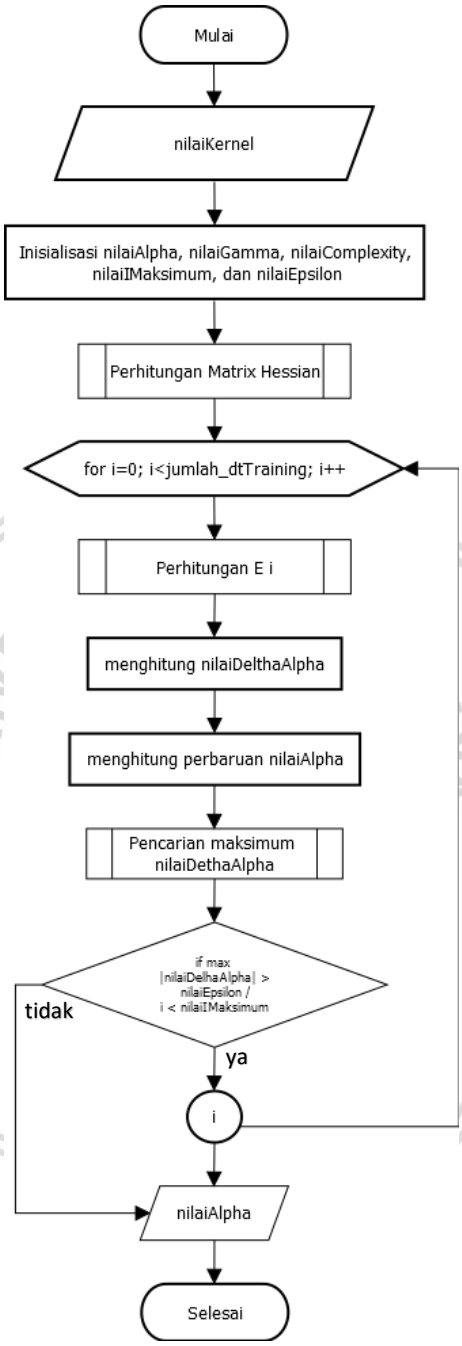


Gambar 4. 4 Diagram alir proses perhitungan kernel *polynomial*

4.2.3 Proses Perhitungan *Sequential Training*

Proses perhitungan *sequential training* dilakukan untuk mendapatkan nilai *alpha* (α) yang akan dibutuhkan dalam menentukan support vector. Proses ini diawali dengan input berupa nilai kernel. Kemudian dilanjutkan dengan inisialisasi parameter yang dibutuhkan yaitu : nilai *gamma* (γ), nilai *epsilon* (ε), nilai iterasi maksimum, nilai C (*complexity*), dan nilai *alpha* (α). Perhitungan yang dilakukan berupa perhitungan matrix Hessian, perhitungan E_i, perhitungan $\delta\alpha_i$, perhitungan

α_i dan pencarian maksimum $\delta\alpha_i$. Proses perhitungan *sequential training* secara lengkap dapat dilihat pada Gambar 4.5.

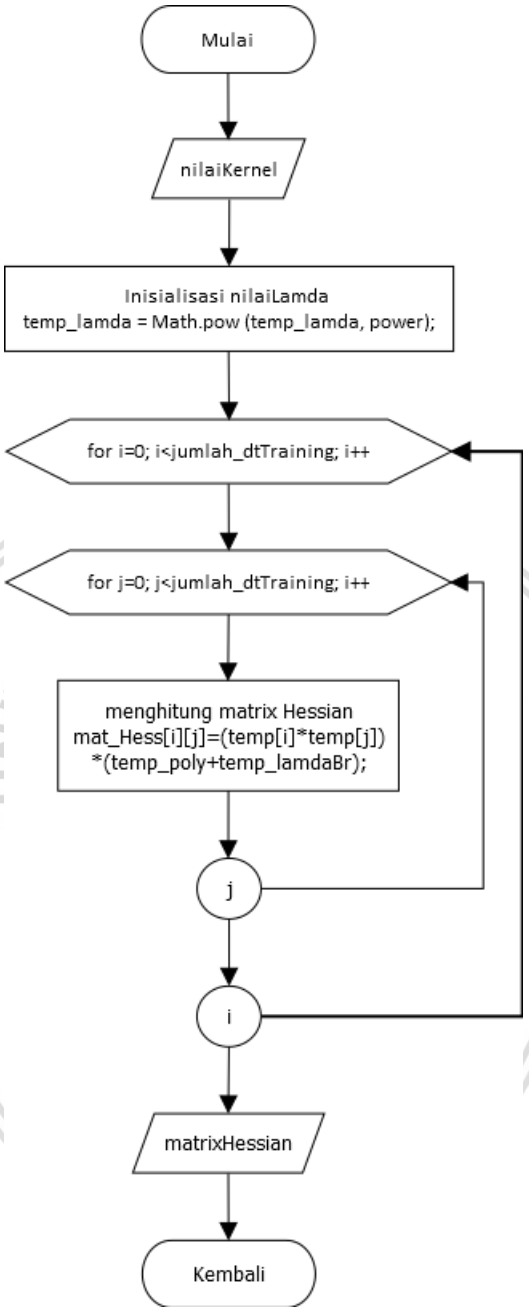


Gambar 4. 5 Diagram alir proses perhitungan *sequential training*

4.2.3.1 Proses Perhitungan Matriks Hessian

Proses perhitungan matriks *Hessian* di awali dengan masukan dari nilai kernel dan data *training*. Kemudian inisialisasi nilai *lamda* (λ) dan diteruskan dengan perhitungan matrikx Hessian sesuai dengan banyaknya data *training* yang telah ditetapkan. Hasil dari perhitungan matriks Hessian ini akan digunakan dalam

perhitungan nilai E_i. Proses perhitungan *matrix Hessian* secara lengkap dapat dilihat pada Gambar 4.6.

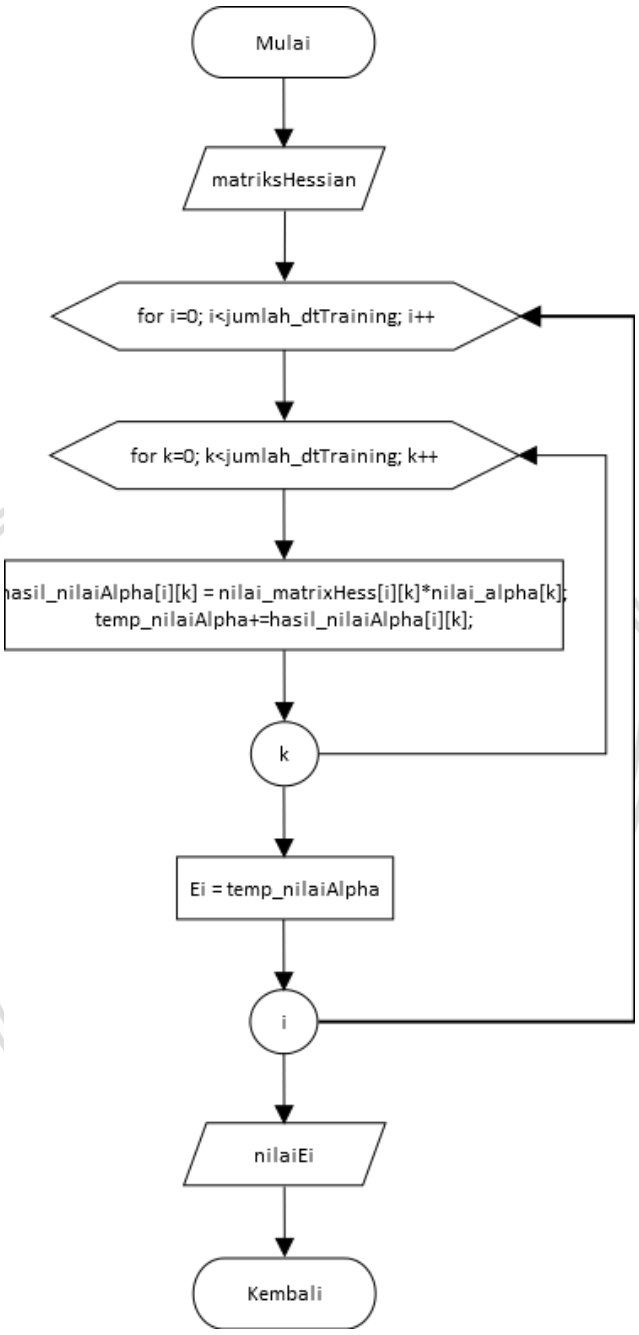


Gambar 4. 6 Diagram alir proses perhitungan *matrix Hessian*

4.2.3.2 Proses Perhitungan Nilai E_i

Proses perhitungan nilai E_i bertujuan untuk menghasilkan nilai E_i dari seluruh data training untuk digunakan dalam perhitungan nilai *delta alpha*. Proses perhitungan ini diawali dengan input berupa matrix Hessian dan inisialisasi nilai $\alpha_i = 0$, kemudian melakukan perulangan sebanyak jumlah data training untuk

menghitung nilai E_i dengan mengalikan nilai *alpha* (α_i) dan matriks Hessian. Proses perhitungan nilai E_i secara lengkap dapat dilihat pada Gambar 4.7.

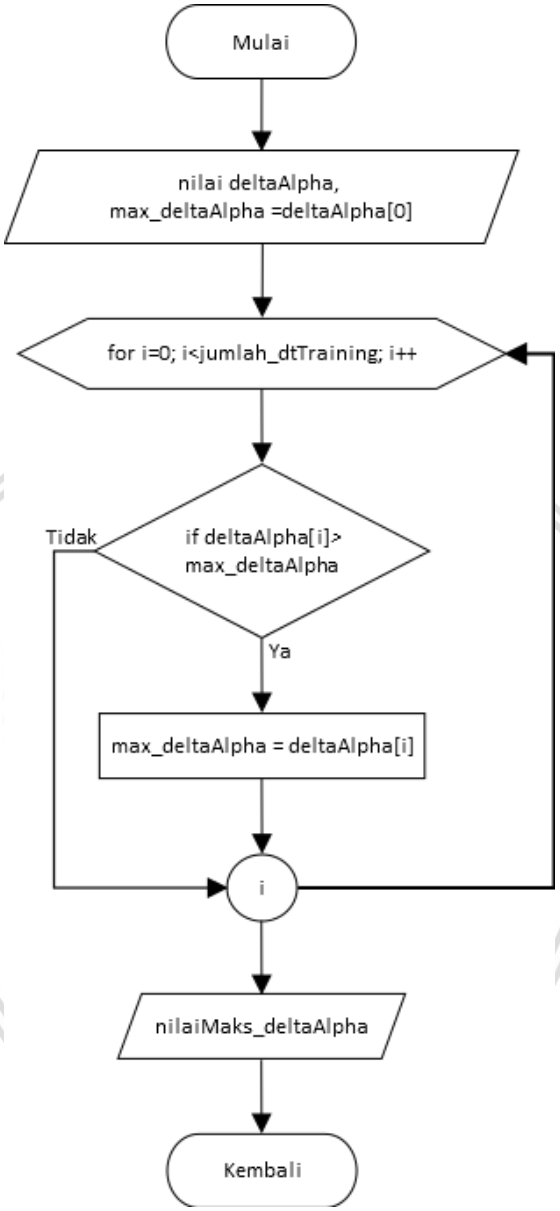


Gambar 4. 7 Diagram alir proses perhitungan nilai E_i

4.2.3.3 Proses Perhitungan Nilai Maksimum Delta Alpha

Proses perhitungan nilai Maksumum *Delta Alpha* diawali dengan inisialisasi nilai *delta alpha* dan inisialisai variabel yang berfungsi menyimpan nilai maksimum *delta alpha*. Proses ini akan terus melakukan perulangan sampai dengan jumlah

data *training* yang ditentukan. Selain itu terdapat suatu kondisi untuk setiap *array* yang menyimpan nilai *delta alpha* yang lebih besar dari sebelumnya maka nilai tersebut yang akan diambil sebagai nilai maksimum *delta alpha*. Proses pencarian nilai maksimum *delta alpha* secara lengkap dapat dilihat pada Gambar 4.8.

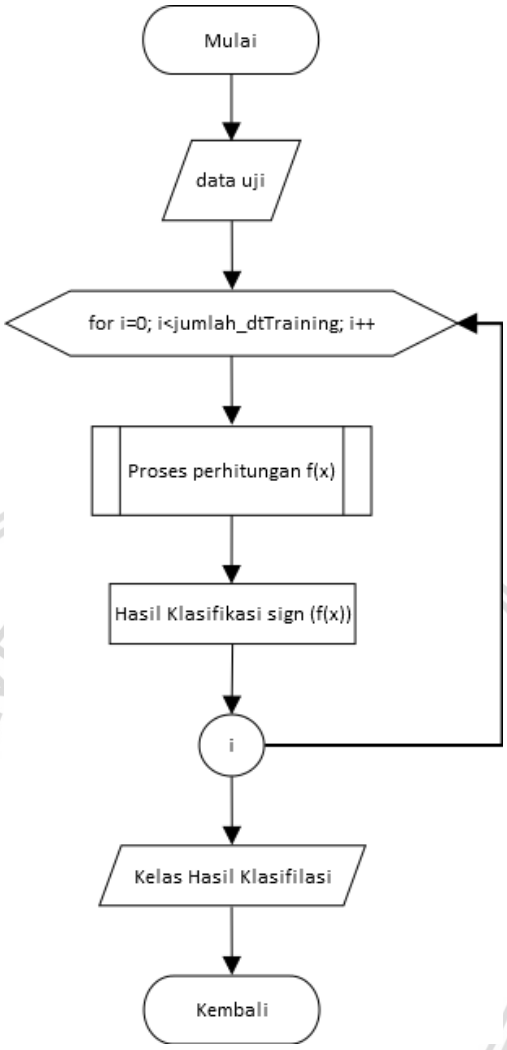


Gambar 4. 8 Diagram alir proses perhitungan nilai maksimum *Delta Alpha*

4.2.4 Proses Perhitungan *Testing*

Proses perhitungan *testing* dilakukan untuk mendapatkan klasifikasi kelas pada data uji. Proses ini diawali dengan memberi masukan, yaitu data uji yang sebelumnya telah ternormalisasi. Perhitungan yang dilakukan berupa perhitungan $f(x)$ dan perhitungan hasil klasifikasi $sign(f(x))$, yang mana pada kedua proses

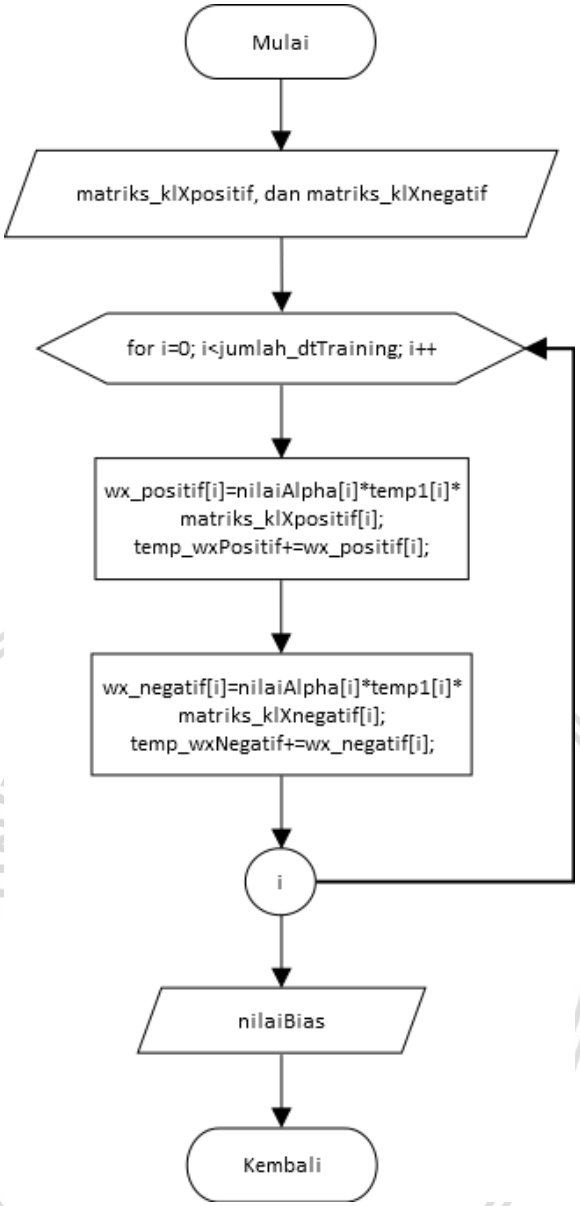
ini dilakukan sejumlah perulangan yang disesuaikan dengan data *testing*. Proses perhitungan *testing* secara lengkap dapat dilihat pada Gambar 4.9.



Gambar 4. 9 Diagram alir proses perhitungan *testing*

4.2.4.1 Proses Perhitungan Nilai Bias

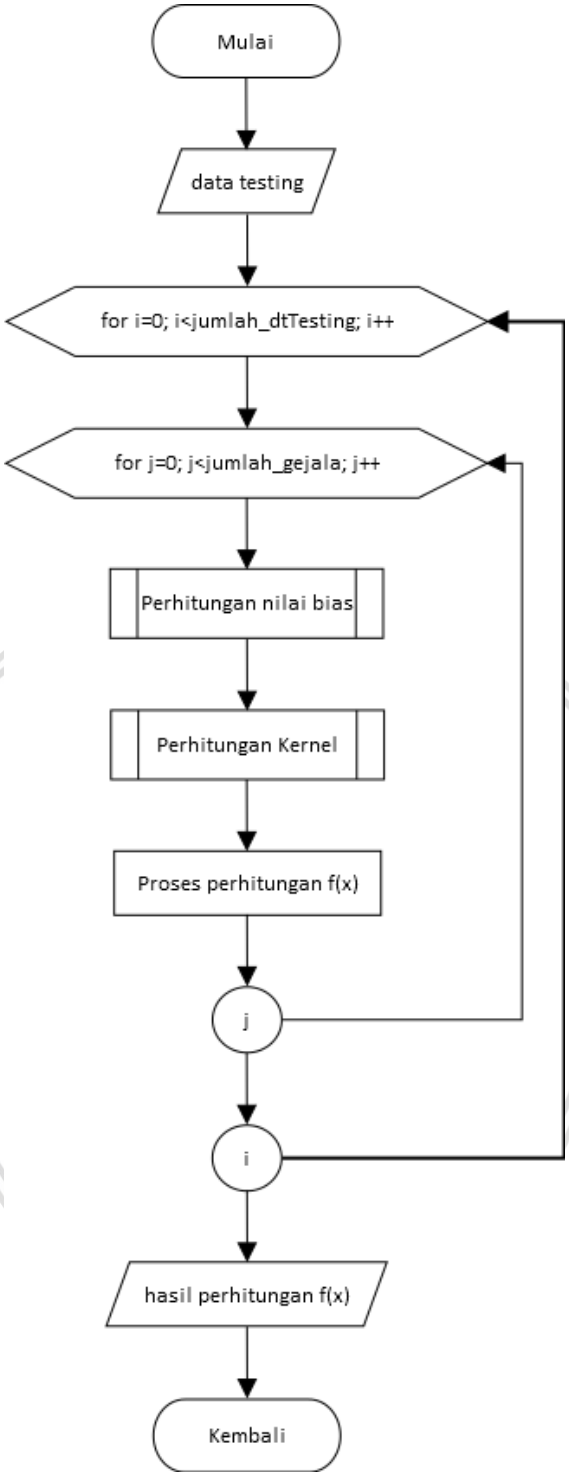
Proses perhitungan nilai bias dilakukan untuk mendapatkan nilai bias yang akan digunakan pada perhitungan fungsi $f(x)$. Proses ini diawali dengan inputan nilai matriks dari kernel x positif dan matriks dari kernel x negatif. Kemudian dilanjutkan dengan penjumlahan antara nilai $w.x^+$ dengan nilai $w.x^-$ dan hasil penjumlahan tersebut dibagi dua untuk mendapatkan nilai bias. Proses perhitungan nilai bias dapat dilihat pada Gambar 4.10.



Gambar 4. 10 Diagram alir proses perhitungan nilai bias

4.2.4.2 Proses Perhitungan Nilai f(x)

Proses perhitungan nilai $f(x)$ diawali dengan pemberian input berupa data uji yang telah ditentukan sebelumnya. Setelah itu dilakukan perhitungan nilai bias dan perhitungan kernel yang akan dibutuhkan saat proses perhitungan nilai f(x). Perhitungan nilai $f(x) = \sum_{i=1}^{m} \alpha_i y_i \mathrm{K}(\mathrm{x}_i, x) + b$, perhitungan ini dilakukan perulangan sejumlah data *testing* yang dimasukkan. Proses perhitungan nilai $f(x)$ dapat dilihat pada Gambar 4.11.

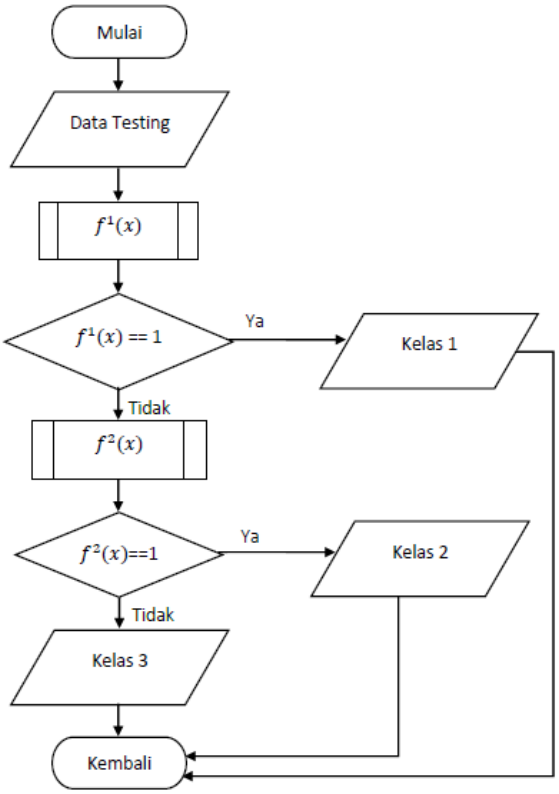


Gambar 4. 11 Diagram alir proses perhitungan nilai f(x)

4.2.4.3 Proses Perhitungan One-Against-All

Klasifikasi penyakit dengan gejala demam pada *Support Vector Machine* menggunakan metode *One against all,* karena klasifikasi penyakit dengan gejala demam ini memiliki 3 pembagian kelas. Kelas pada sistem ini dibagi menjadi kelas

demam berdarah, kelas malaria dan kelas tifoid. Sehingga dibutuhkan lebih dari satu fungsi klasifikasi. Pada sistem ini, fungsi pertama melakukan pemisahan kelas menjadi kelas dengan prediksi kelas 1 dan kelas dengan prediksi kelas selain kelas 1, fungsi kedua melakukan pemisahan kelas menjadi kelas dengan prediksi kelas 2 dan kelas dengan prediksi selain kelas 2, yaitu kelas 3. Data set akan masuk menjadi kelas 1 apabila klasifikasi dengan fungsi pertama mendapatkan hasil +1, sebaliknya data set akan masuk pada fungsi kedua apabila klasifikasi dengan fungsi pertama mendapatkan hasil -1, fungsi kedua bertujuan untuk memisahkan apakah kelas tersebut masuk pada kelas 2 atau kelas 3. Jika pada fungsi kedua, data tersebut bernilai +1 maka termasuk dalam kelas 2 sedangkan jika data tersebut bernilai -1 maka termasuk dalam kelas 3. Proses perhitungan *Support Vector Machine Multiclass* dengan metode *one against all* dapat dilihat pada Gambar 4.12.



Gambar 4. 12 Diagram alir proses perhitungan one against all

4.3 Perhitungan Manualisasi

Perhitungan manualisasi *Support Vector Machine* memiliki beberapa tahap. Tahapan tersebut dibagi menjadi 3 proses, yang mana proses perhitungan yang pertama adalah perhitungan kernel, kedua adalah perhitungan *sequential training*, dan ketiga adalah perhitungan *testing*. Sebelum ketiga proses ini diuraikan, terdapat data set yang digunakan dalam contoh perhitungan ini. Data set yang digunakan untuk mengimplementasikan perhitungan *Support Vector*

Machine menggunakan data yang dipilih secara random dari 130 data yang akan digunakan dalam implementasi sistem menjadi 21 dataset. Keduapuluh satu data ini selanjutnya akan dibagi menjadi data latih sebanyak 15 data dan data uji sebanyak 6 data. Data set yang akan digunakan pada perhitungan ini terdapat pada tabel 4.2.

Tabel 4. 2 Dataset dalam sampel Perhitungan Manualisasi

DATA SET	NO-P	GEJALA															KLS
		A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12	A13	A14	A15	
Data Latih	1	0	0.5	0.75	0.75	0.9	0.75	0.85	0.6	0.8	0	0.85	0.75	0.5	0.65	0	1
	2	0.8	0.85	0	0.5	0.9	0	0	0.2	0	0	0	0	0.5	0.65	0.85	2
	3	0.8	0.5	0.75	0.5	0	0.75	0.6	0.6	0	0	0	0.75	0.5	0.65	0	1
	4	0.8	0.85	0	0.5	0.7	0	0	0.6	0.4	0.7	0	0	0.5	0.65	0.85	2
	5	0	0.5	0.75	0.5	0.9	0	0.6	0.6	0.8	0	0.85	0.75	0.5	0.65	0	1
	6	0.8	0.5	0	0.5	0.9	0.75	0	0.2	0	0	0	0	0.5	0.65	0.85	2
	7	0	0.5	0.75	0.5	0.7	0.75	0.85	0.6	0.4	0	0.85	0	0	0.65	0	1
	8	0.8	0.85	0	0.5	0.7	0	0	0.2	0	0	0	0	0.5	0.65	0.85	2
	9	0.8	0.5	0	0.5	0.7	0	0	0.2	0.8	0	0.85	0.75	0.5	0.65	0	1
	10	0.8	0.85	0.75	0.5	0.7	0	0.6	0.2	0	0	0	0.75	0	0	0.4	2
	20	0	0.5	0.75	0.5	0.7	0.75	0.6	0.2	0.8	0.7	0.4	0	0.5	0.65	0	3
	22	0	0	0	0.2	0.9	0.9	0.85	0.2	0.4	0.7	0	0	0	0.65	0	3
	24	0	0	0	0.2	0.7	0.9	0	0.2	0.4	0.7	0	0	0	0.65	0	3
	26	0	0.5	0	0.2	0.9	0.9	0.85	0.2	0.4	0.7	0.4	0	0.5	0	0	3
	28	0	0	0	0.2	0.7	0.9	0.85	0.2	0.8	1	0	0	0.5	0	0	3
Data Uji	101	0	0	0	0.2	0.9	0.9	0.6	0.2	0.4	0.7	0	0	0.5	0.65	0	3
	102	0.8	0.85	0	0.5	0.9	0	0.6	0.2	0	0	0	0	0.5	0.65	0.85	2
	103	0	0.5	0.75	0.5	0.7	0	0.6	0.2	0	0	0	0	0.5	0	0	3
	104	0	0	0	0.5	0.9	0	0.85	0.6	0.4	0	0.4	0.75	0.5	0	0	1
	106	0.8	0	0.75	0.5	0.9	0	0.85	0.2	0.8	0	0.85	0	0.5	0.65	0.45	1
	108	0.8	0.5	0	0.5	0.7	0	0	0.2	0	0	0.4	0.75	0	0	0.4	2

Dataset di tabel 4.2 menunjukkan data pada pasien yang memiliki penyakit dengan gejala demam, klasifikasi pada data tersebut menunjukkan terdapat 15 gejala yang telah diuraikan keterangannya pada tabel 4.1. 15 gejala inilah yang akan menjadi parameter dalam perhitungan *Support Vector Machine* dengan kategori kelas yang ada pada data tersebut berjumlah 3 kelas. Kelas 1 menunjukkan bahwa pasien mengalami penyakit Demam tifoid, Kelas 2 menunjukkan bahwa pasien mengalami penyakit Demam Malaria dan Kelas 3 menunjukkan bahwa pasien mengalami penyakit Demam Brdarah.

Perhitungan manualisasi *Support Vector Machine* dibagi menjadi beberapa tahapan sebagai berikut:

- Proses Perhitungan Kernel

1. Perhitungan Kernel

Perhitungan kernel menggunakan kernel *polynomial* karena pada konfigurasi menggunakan kernel *polynomial* akurasi yang didapatkan tinggi baik saat menggunakan data latih maupun data uji. Dibandingkan dengan kernel liner, *polynomial* lebih fleksibel dan lebih kaku dibandingkan dengan kernel RBF (Munir, 2016). Penggunaan kernel ini bertujuan untuk memetakan fitur lama menjadi fitur yang baru sehingga perhitungan SVM dapat digunakan pada data yang non-linear. Perhitungan kernel *polynomial* dilakukan dengan menggunakan persamaan 2.15 dan menggunakan data yang terdapat pada tabel 4.2, dengan penentuan nilai c=1 dan d=2. Tabel 4.3 menampilkan hasil dari perhitungan kernel *polynomial*, sedangkan hasil lengkapnya dapat dilihat pada lampiran D. Contoh perhitungan kernel *polynomial* pada data di baris ke-1 dan kolam ke-1 terdapat pada perhitungan sebagai berikut:

$K(x1, y1) = (\,(0 \times 0) + (0.5 \times 0.5) + (0.75 \times 0.75) + (0.75 \times 0.75) + (0.9 \times 0.9) + (0.75 \times 0.75) + (0.85 \times 0.85) + (0.6 \times 0.6) + (0.8 \times 0.8) + (0 \times 0) + (0.85 \times 0.85) + (0.75 \times 0.75) + (0.5 \times 0.5) + (0.65 \times 0.65) + (0 \times 0) + 1)^2$ = 55.16775625

Tabel 4. 3 Hasil perhitungan kernel *polynomial*

NO-P	1	3	5	...	24	26	28
1	55.16775625	23.571025	41.796225	...	11.00580625	21.50640625	17.53515625
3	23.571025	27.2484	16.14030625	...	5.37080625	8.439025	7.049025
5	41.796225	16.14030625	38.3161	...	6.72105625	13.69	10.5625
7	35.13525625	15.34680625	25.27575625	...	8.80605625	15.98000625	11.74775625
9	24.72575625	12.215025	23.49825625	...	5.62875625	8.5849	6.3504
2	11.57700625	9.65655625	10.74200625	...	4.80705625	6.890625	4.0804
4	14.30730625	11.20575625	13.37730625	...	7.74230625	10.080625	8.8804
6	14.3641	12.215025	9.62550625	...	8.22255625	9.765625	7.263025
8	10.38450625	9.65655625	9.59450625	...	4.21275625	5.978025	3.5344
10	17.514225	15.3664	15.2881	...	2.6569	7.317025	4.5796
20	32.06390625	14.26950625	23.280625	...	12.51390625	19.580625	19.404025
22	17.8084	7.99475625	10.77480625	...	12.90605625	16.58525625	18.16890625
24	11.00580625	5.37080625	6.72105625	...	11.91975625	10.0489	11.56
26	21.50640625	8.439025	13.69	...	10.0489	22.39655625	20.36265625
28	17.53515625	7.049025	10.5625	...	11.56	35.7338	24.92505625

Setelah hasil perhitungan kernel *polynomial* didapatkan, dilanjutkan dengan proses selanjutnya yaitu perhitungan *sequential training*.

- Proses Perhitungan *Sequential Training*

2. Menghitung Matriks Hessian

Menghitung matriks Hessian merupakan tahap awal dari proses *sequential training*, yang mana perhitungan ini menggunakan inisialisasi nilai *lamda* sebesar 0.5. Perhitungan matriks Hessian dilakukan pada ke 15 data latih dengan mengalikan 3 nilai yaitu kelas dari data baris, kelas dari data kolom, dan penjumlahan nilai kernel data dan nilai *lamda* yang dikuadratkan dengan menggunakan data yang terdapat pada tabel 4.3. Perhitungan matriks Hessian dilakukan dengan menggunakan persamaan 2.1. Tabel 4.4 menampilkan hasil dari perhitungan matriks Hessian, sedangkan hasil lengkapnya ddapat dilihat pada lampiran E. Contoh perhitungan matriks Hessian pada data di baris ke-1 dan kolom ke-1 terdapat pada perhitungan sebagai berikut:

$$Matriks\ Hessian_{1,1} = 1 \times 1 \times (55.16775625 + 0.5^2) = 55.41775625$$

Tabel 4. 4 Hasil Perhitungan Matriks Hessian

NO-P	1	3	...	24	26	28
1	55.41775625	23.821025	...	-11.25580625	-21.75640625	-17.78515625
3	23.821025	27.4984	...	-5.62080625	-8.689025	-7.299025
5	42.046225	16.39030625	...	-6.97105625	-13.94	-10.8125
7	35.38525625	15.59680625	...	-9.05605625	-16.23000625	-11.99775625
9	24.97575625	12.465025	...	-5.87875625	-8.8349	-6.6004
2	-11.82700625	-9.90655625	...	5.05705625	7.140625	4.3304
4	-14.55730625	-11.45575625	...	7.99230625	10.330625	9.1304
6	-14.6141	-12.465025	...	8.47255625	10.015625	7.513025
8	-10.63450625	-9.90655625	...	4.46275625	6.228025	3.7844
10	-17.764225	-15.6164	...	2.9069	7.567025	4.8296
20	-32.31390625	-14.51950625	...	12.76390625	19.830625	19.654025
22	-18.0584	-8.24475625	...	13.15605625	16.83525625	18.41890625
24	-11.25580625	-5.62080625	...	12.16975625	10.2989	11.81
26	-21.75640625	-8.689025	...	10.2989	22.64655625	20.61265625
28	-17.78515625	-7.299025	...	11.81	20.61265625	25.17505625

Setelah hasil perhitungan matriks Hessian didapatkan, dilanjutkan dengan tahap selanjutnya yaitu menghitung nilai Ei.

3. Menghitung nilai Ei

Dalam perhitungan nilai Ei, dibutuhkan inisialisasi awal nilai *alpha*, sehingga dalam perhitungan ini ditetapkan nilai *alpha* awal yaitu 0. Perhitungan nilai Ei dilakukan dengan menggunakan persamaan 2.2 dan menggunakan data yang terdapat pada tabel 4.4. Nilai Ei didapatkan dari penjumlahan seluruh kolom pada setiap data, setiap kolom berisi hasil dari perhitungan nilai *alpha* awal pada setiap data dikali dengan nilai dari matriks Hessian. Tabel 4.5 menampilkan hasil dari perhitungan nilai Ei. Contoh perhitungan nilai Ei pada data dengan NO-P sama dengan 1 terdapat pada perhitungan sebagai berikut :

$$E_1 = \big((0 \times 554178) + (0 \times 23.8210) + (0 \times 42.0462) + (0 \times 35.3853) + (0 \times 24.9758) + (0 \times -11.8270) + (0 \times -14.5573) + (0 \times 14.6141) + (0 \times -10.6345) + (0 \times -17.7642) + (0 \times -32.3139) + (0 \times -18.0584) + (0 \times -11.2558) + (0 \times -21.75640) + (0 \times -17.7852)\big) = 0$$

Tabel 4. 5 Hasil perhitungan nilai Ei

NO-P	1	3	5	7	9	2	4	6	8	10	20	22	24	26	28	Ei
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
26	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
28	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

4. Menghitung Nilai *Delta Alpha*

Perhitungan nilai *delta alpha* dilakukan dengan menggunakan persamaan 2.3 dan menggunakan data yang terdapat pada tabel 4.5. Perhitungan nilai *delta alpha* membutuhkan nilai *gamma*, nilai *gamma* dapat diperoleh dari CLR atau konstanta *learning rate* sebesar 0.01 dibagi dengan nilai maksimum dari diagonal matriks Hessian sebesar 55.41775625. Sehingga nilai *gamma* yang didapatkan adalah 0.000180448. Pada perhitungan nilai *delta alpha* juga diinisialisasi nilai C

sebesar 1. Tabel 4.6 menampilkan hasil dari perhitungan nilai *delta alpha*. Contoh perhitungan nilai *delta alpha* pada data dengan NO-P sama dengan 1 terdapat pada perhitungan sebagai berikut:

$$\delta\alpha_1 = \min\{\max[0.000180448(1-0), 0], 1-0\} = 0.000180448$$

Tabel 4. 6 Hasil Perhitungan nilai *delta alpha*

NO-P	Δαi
1	0.000180448
3	0.000180448
5	0.000180448
7	0.000180448
9	0.000180448
2	0.000180448
4	0.000180448
6	0.000180448

NO-P	Δαi
8	0.000180448
10	0.000180448
20	0.000180448
22	0.000180448
24	0.000180448
26	0.000180448
28	0.000180448

5. Menghitung Nilai *Alpha*

Perhitungan nilai *alpha* pada tahap ini dimaksudkan untuk memperbaharui nilai *alpha*. Perhitungan ini menggunakan persamaan 2.4 dan menggunakan data yang terdapat pada tabel 4.6, yaitu dengan menambahkan nilai dari *delta alpha* yang telah didapatkan pada langkah sebelumnya dengan nilai dari *alpha* awal. Hasil dari perhitungan nilai *alpha* pada tahap ini dapat ditunjukkan pada tabel 4.7 . Contoh perhitungan nilai *alpha* yang baru pada data no-p 1 ditunjukkan pada perhitungan berikut:

$$\alpha_1 = 0 + 0.000180448 = 0.000180448$$

Tabel 4. 7 Hasil Perhitungan nilai *alpha*

NO-P	αi
1	0.000180448
3	0.000180448
5	0.000180448
7	0.000180448
9	0.000180448
2	0.000180448
4	0.000180448
6	0.000180448

NO-P	αi
8	0.000180448
10	0.000180448
20	0.000180448
22	0.000180448
24	0.000180448
26	0.000180448
28	0.000180448

6. Melakukan iterasi (3-4-5) dan mendapatkan nilai *alpha* maksimum

Pada perhitungan tahap ke 3, 4, dan 5 atau tahap perhitungan nilai Ei, *delta alpha* dan nilai *alpha* akan dilakukan iterasi sesuai dengan iterasi maksimum atau saat nilai *delta alpha* mencapai batas yang ditentukan dari nilai *epsilon* (maksimum *delta alpha* < *epsilon*). Pada perhitungan manualisasi ini telah diinisialisasikan jumlah iterasi maksimum = 2. Sehingga perhitungan pada tahap pencarian nilai Ei, *delta alpha* dan nilai *alpha* akan diulang dengan menggunakan hasil yang telah didapatkan pada perhitungan sebelumnya. Tabel 4.8 dan 4.9 merupakan hasil perhitungan nilai Ei, *delta alpha* dan nilai *alpha* yang dilakukan pada iterasi ke 2. Hasil lengkap perhitungan nilai Ei dapat dilihat pada lampiran F.

Tabel 4. 8 Hasil Perhitungan Ei iterasi

NO-P	1	3	5	7	…	28	Ei
1	0.01	0.004298	0.007587	0.006385184	…	-0.00321	0.001999
3	0.004298	0.004962	0.002958	0.002814406	…	-0.00132	-0.00143
5	0.007587	0.002958	0.006959	0.004606061	…	-0.00195	0.003631
7	0.006385	0.002814	0.004606	0.005508615	…	-0.00216	-0.00068
9	0.004507	0.002249	0.004285	0.002351345	…	-0.00119	-0.001
2	-0.00213	-0.00179	-0.00198	-0.00150823	…	0.000781	0.014457
4	-0.00263	-0.00207	-0.00246	-0.00178761	…	0.001648	0.016902
6	-0.00264	-0.00225	-0.00178	-0.00193354	…	0.001356	0.015155
8	-0.00192	-0.00179	-0.00178	-0.00136789	…	0.000683	0.013321
10	-0.00321	-0.00282	-0.0028	-0.00207926	…	0.000871	0.005711
20	-0.00583	-0.00262	-0.00425	-0.0042504	…	0.003547	0.008438
22	-0.00326	-0.00149	-0.00199	-0.00269208	…	0.003324	0.011288
24	-0.00203	-0.00101	-0.00126	-0.00163414	…	0.002131	0.009078
26	-0.00393	-0.00157	-0.00252	-0.00292867	…	0.00372	0.011198
28	-0.00321	-0.00132	-0.00195	-0.00216497	…	0.004543	0.012769

Tabel 4. 9 Hasil Perhitungan Iterasi *delta Alpha* dan perbaruan nilai *Alpha*

NO-P	δαi	αi	Kelas (y_i)	Katogori kelas
1	0.000180448	0.000360895	1	Positif
3	0.000180707	0.000361154	1	Positif
5	0.000180448	0.000360895	1	Positif
7	0.000180570	0.000361018	1	Positif
9	0.000180629	0.000361076	1	Positif
2	0.000180448	0.000360895	2	Negatif
4	0.000180448	0.000360895	2	Negatif
6	0.000180448	0.000360895	2	Negatif
8	0.000180448	0.000360895	2	Negatif
10	0.000180448	0.000360895	2	Negatif
20	0.000180448	0.000360895	3	Negatif

22	0.000180448	0.000360895	3	Negatif
24	0.000180448	0.000360895	3	Negatif
26	0.000180448	0.000360895	3	Negatif
28	0.000180448	0.000360895	3	Negatif

Selanjutnya adalah menentukan nilai *alpha* maksimum dari masing masing kategori kelas negatif dan kategori kelas positif, kategori kelas negatif berasal dari kelas 2 dan kelas 3 sedangkan kategori kelas positif berasal dari kelas 1. Dari tabel 4.9 dapat disimpulkan bahwa nilai *alpha* maksimal yang diperoleh dari kategori kelas negatif adalah 0.000360895 dari no-p 2, dan nilai *alpha* maksimum yang diperoleh dari kelas positif adalah 0.000361154 dari no-p 3. Nilai *alpha* maksimum selanjutnya akan digunakan dalam perhitungan nilai bobot yang akan di bahas pada tahap berikutnya.

7. Menghitung Nilai bobot

Sebelum menghitung bobot pada setiap data, dibutuhkan perhitungan kernel dari setiap data terhadap data dengan nilai *alpha* tertinggi pada kategori kelas positif atau data dengan no-p 3 ($K(x_i,x^+)$) dan perhitungan kernel dari setiap data terhadap data dengan nilai *alpha* tertinggi pada kategori kelas negatif atau data dengan no-p 2 ($K(x_i,x^-)$). Tabel 4.10 menampilkan hasil dari perhitungan $K(x_i,x^+)$ dan $K(x_i,x^-)$. Berikut contoh perhitungan kernel dari data dengan no-p 1 terhadap data dengan nilai *alpha* tertinggi dari kategori kelas positif dan negatif:

$$K(x_1,x^+) = (\,(0\times0.8)+(0.5\times0.5)+(0.75\times0.75)+\,(0.75\times0.5) + (0.9\times0)+(0.75\times0.75)+\,(0.85\times0.6)+(0.6\times0.6) + (0.8\times0)+\,(0\times0)+(0.85\times0)+(0.75\times0.75) + \,(0.5\times0.5)+(0.65\times0.65)+(0\times0)+1)^2 = 23.571025$$

$$K(x_1,x^-) = (\,(0\times0.8)+(0.5\times0.85)+(0.75\times0)+\,(0.75\times0.5) + (0.9\times0.9)+(0.75\times0)+\,(0.85\times0)+(0.6\times0.2) + (0.8\times0)+\,(0\times0)+(0.85\times0)+(0.75\times0)+\,(0.5\times0.5) + (0.65\times0.65)+(0\times0.85)+1)^2 = 11.57700625$$

Tabel 4. 10 Hasil dari perhitungan $K(x_i,x^+)$ dan $K(x_i,x^-)$

no-p	3	2
1	23.571025	11.57700625
3	27.2484	9.65655625
5	16.14030625	10.74200625
7	15.34680625	8.10825625
9	12.215025	13.37730625

2	9.65655625	23.59530625
4	11.20575625	22.63380625
6	12.215025	20.7936
8	9.65655625	21.87900625
10	15.3664	13.12250625
20	14.26950625	9.10530625
22	7.99475625	5.62875625
24	5.37080625	4.80705625
26	8.439025	6.890625
28	7.049025	4.0804

Selanjutnya Perhitungan nilai bobot akan dilakukan dengan menghitung masing masing bobot pada data ke 1 sampai ke 15 dengan menggunakan persamaan 2.5, yang mana untuk setiap data tersebut memiliki dua hasil perhitungan bobot. Hasil perhitungan bobot pertama ($w.x^-$) didapatkan dari perkalian antara nilai *alpha* terakhir pada data (αi), kelas data (y_i) dan hasil perhitungan kernel dari data tersebut terhadap data dengan nilai *alpha* tertinggi pada kategori kelas negatif atau data dengan no-p 2 ($K(x_i,x^-)$). Hasil perhitungan bobot kedua ($w.x^+$) didapatkan dari perkalian antara nilai *alpha* terakhir pada data (αi), kelas data (y_i) dan hasil perhitungan kernel dari data tersebut terhadap data dengan nilai *alpha* tertinggi pada kategori kelas positif atau data dengan no-p 3 ($K(x_i,x^+)$). Tabel 4.11 menampilkan hasil dari perhitungan ($w.x^+$) dan ($w.x^-$). Berikut contoh perhitungan bobot kategori kelas positif ($w.x^+$) dan bobot kategori kelas negatif ($w.x^-$) dari data dengan no-p 1:

$$w.x^+ = 0.000360895 \times 1 \times 23.571025 = 0.008506669$$

$$w.x^- = 0.000360895 \times 1 \times 11.57700625 = 0.004178086$$

Tabel 4. 11 Hasil Perhitungan Bobot ($w.x^+$) dan ($w.x^-$)

no-p	W.X+	W.X-
1	0.008506669	0.004178086
3	0.009840871	0.003487505
5	0.005824958	0.003876738
7	0.005540474	0.002927227
9	0.004410554	0.004830226
2	-0.003485004	-0.008515432
4	-0.004044103	-0.008168431
6	-0.004408343	-0.00750431
8	-0.003485004	-0.007896027
10	-0.005545659	-0.004735849
20	-0.005149796	-0.003286061

22	-0.002885269	-0.002031391
24	-0.001938298	-0.001734843
26	-0.003045603	-0.002486793
28	-0.002543959	-0.001472597
∑	-0.002407513	-0.028531953

Selanjutnya nilai bobot yang telah didapat akan digunakan dalam perhitungan bias yang akan dibahas dalam tahap selanjutnya.

8. Menghitung Nilai bias

Perhitungan nilai bias didapatkan dengan menggunakan persamaan 2.6. Untuk mendapatkan nilai tersebut, maka dilakukan penjumlahan antara seluruh nilai bobot positif dengan seluruh nilai bobot negatif, kemudian dikali dengan $-\frac{1}{2}$. Dari tabel 4.11 dapat disimpulkan bahwa jumlah seluruh nilai bobot positif adalah -0.002407513 dan jumlah seluruh nilai bobot negatif adalah -0.028531953. Berikut contoh perhitungan nilai bias pada perhitungan manualisasi ini:

$$nilai\ bias = -\frac{1}{2} \times (-0.002407513 + -0.028531953) = 0.015469733$$

Selanjutnya nilai bias akan digunakan dalam proses perhitungan *testing* yang akan dibahas pada tahap berikutnya.

- Proses Perhitungan *Testing*

9. Menghitung Nilai f(x) Level 1

Perhitungan nilai f(x) pada level pertama menggunakan perhitungan yang disebutkan pada persamaan 2.10, sebelum menghitung nilai f(x) perlu dilakukan perhitungan kernel dari setiap data latih terhadap data uji yang telah ditetapkan sebelumnya. Contoh perhitungan kernel untuk data latih no-p 1 terhadap data uji no-p 101:

$$\begin{aligned} K(x_1, x_{101}) = (\ &(0 \times 0) + (0.5 \times 0) + (0.75 \times 0) + (0.75 \times 0.2) + (0.9 \times 0.9) \\ &+ (0.75 \times 0.9) + (0.85 \times 0.6) + (0.6 \times 0.2) + (0.8 \times 0.4) \\ &+ (0 \times 0.7) + (0.85 \times 0) + (0.75 \times 0) + (0.5 \times 0.5) \\ &+ (0.65 \times 0.65) + (0 \times 0) + 1)^2 = 18.12630625 \end{aligned}$$

Tabel 4.12 menunjukkan hasil dari perhitungan kernel data latih terhadap data uji, yang mana hasil tersebut akan digunakan dalam perhitungan bobot data uji.

Tabel 4. 12 Hasil perhitungan kernel data latih terhadap data uji

no-p	101	103	102	108	104	106
1	18.12630625	13.67150625	15.30765625	10.74200625	22.4676	31.640625
3	8.57025625	7.79805625	12.02355625	7.96650625	8.59955625	14.100025
5	11.44130625	11.71350625	13.23140625	9.93825625	19.38200625	27.95765625
7	13.08630625	10.12830625	11.27280625	6.0025	11.98890625	22.5625
9	7.63140625	5.1984	13.37730625	12.76275625	12.05825625	21.114025
2	6.87750625	6.734025	23.59530625	11.055625	5.9049	14.402025
4	10.06475625	6.426225	22.63380625	10.660225	7.0225	16.120225
6	10.87350625	5.8564	20.7936	9.9225	5.9049	14.402025
8	5.96580625	6.027025	21.87900625	10.144225	5.0625	13.068225
10	4.5369	9.78125625	15.86030625	12.72705625	9.44025625	14.53515625
20	18.38265625	10.25600625	11.40750625	4.7961	10.4976	21.576025
22	18.33980625	5.1984	8.30880625	3.1329	8.48265625	11.662225
24	12.90605625	2.6569	4.80705625	2.6569	4.0401	6.31265625
26	16.8921	7.7284	9.828225	4.7524	11.03900625	12.83430625
28	18.49	5.7121	6.4009	2.6569	9.87530625	11.44130625

Setelah mendapatkan nilai kernel, maka perhitungan selanjutnya adalah, perhitungan bobot data uji. Perhitungan bobot data uji didapatkan dengan cara melakukan perkalian antara nilai *alpha* terakhir data tersebut dengan kelas data tersebut dan juga dengan kernel data latih terhadap data uji. Tabel 4.13 menunjukkan hasil dari perhitungan bobot setiap data. Contoh dari perhitungan bobot untuk data 1 terhadap data uji 101, sebagai berikut:

$$\alpha_i y_i K(\mathrm{x}_1, x_{101}) = 0.000360895 \times 1 \times 18.12630625 = 0.006535157$$

Tabel 4. 13 Hasil perhitungan bobot pada setiap data

no-p	101	103	102	108	104	106
1	0.006541696	0.00493398	0.005524459	0.003876738	0.008108448	0.011418948
3	0.003095183	0.0028163	0.004342356	0.002877136	0.003105765	0.005092282
5	0.004129112	0.004227348	0.00477515	0.003586669	0.006994872	0.010089783
7	0.004724393	0.003656502	0.004069687	0.002167011	0.004328212	0.00814547
9	0.002755518	0.001877018	0.004830226	0.004608326	0.004353948	0.00762377
2	-0.002482059	-0.002430277	-0.008515432	-0.003989922	-0.00213105	-0.005197621
4	-0.003632322	-0.002319193	-0.008168431	-0.003847224	-0.002534386	-0.005817711
6	-0.003924196	-0.002113546	-0.00750431	-0.003580982	-0.00213105	-0.005197621
8	-0.002153031	-0.002175124	-0.007896027	-0.003661002	-0.001827032	-0.004716259
10	-0.001637345	-0.003530008	-0.005723908	-0.004593133	-0.003406943	-0.005245668
20	-0.006634212	-0.003701343	-0.004116914	-0.001730889	-0.003788533	-0.007786683
22	-0.006618747	-0.001876077	-0.002998608	-0.001130648	-0.00306135	-0.004208841
24	-0.004657733	-0.000958862	-0.001734843	-0.000958862	-0.001458053	-0.002278207

26	-0.006096277	-0.002789142	-0.003546959	-0.001715118	-0.003983924	-0.004631839
28	-0.006672952	-0.002061469	-0.002310054	-0.000958862	-0.00356395	-0.004129112
∑	-0.02326297	-0.006443896	-0.028973607	-0.009050762	-0.000995024	-0.006839309

Selanjutnya jumlah dari bobot data uji pada data 101, 103, 102, 108, 104, dan 106 digunakan untuk mencari fungsi f(x), nilai f(x) didapatkan dari penjumlahan antara jumlah bobot masing-masing data uji dengan nilai bias yaitu 0.015469733, sebagaimana yang telah dituliskan pada persamaan 2.10. Hasil perhitungan dari nilai f(x) dapat dilihat pada tabel 4.14. Contoh perhitungan dalam mendapatkan nilai f(x) pada data uji 101 sebagai berikut :

$$f(x) = -0.02326297 + 0.015469733 = -0.007793237$$

$$f(x) = sign\,(-0.007793237) = -1$$

Tabel 4. 14 Hasil dari perhitungan nilai f(x)

no-p	F(X)	FUNGSI KLASIFIKASI	KELAS ASLI
101	-0.007793237	-1	3
103	0.009025838	1	3
102	-0.013503874	-1	2
108	0.006418971	1	2
104	0.014474709	1	1
106	0.008630424	1	1

Dapat diketahui dari tabel 4.14 bahwa pada fungsi klasifikasi, no-p 103, 108, 104, dan 106 mendapatkan hasil berupa nilai 1 sehingga terklasifikasi sebagai data dengan kelas 1 sedangkan 101, dan 102 mendapatkan hasil berupa nilai -1 sehingga terklasifikasi sebagai data dengan kelas selain kelas 1, yaitu kelas 2 atau kelas 3. Karena perhitungan SVM menggunakan *SVM Multiclass* maka untuk mengetahui apakah data uji dengan nilai -1 terklasifikasi sebagai kelas 2 ataupun kelas 3 harus dilakukan perhitungan kembali untuk mendapatkan nilai f(x) pada level ke 2. Perhitungan nilai f(x) pada level ke 2 akan dibahas pada tahapan berikutnya.

10. Menghitung Nilai f(x) Level 2

Perhitungan nilai f(x) di level 2 dilakukan dengan cara mengulang tahapan yang sebelumnya di lakukan pada level 1, tahapan yang sebelumnya harus dilakukan adalah : perhitungan kernel, perhitungan matriks Hessian, perhitungan nilai E_i, perhitungan nilai *delta alpha*, perhitungan nilai *alpha*, melakukan iterasi (C-D-E) dan mendapatkan nilai *alpha* maksimum, menghitung nilai bobot dan menghitung nilai bias. Setelah tahapan tersebut selesai dilakukan, perhitungan dapat dilanjutkan ke tahapan menghitung nilai f(x) level 2 dan mendapatkan hasil klasifikasi dari perhitungan manualisasi SVM ini.

A. Perhitungan kernel data latih pada level 2

Perhitungan kernel data latih pada level 2 menggunakan kernel *polynomial*, dengan menggunakan persamaan 2.15. pada perhitungan ini ditentukan nilai c=1 dan d=2. Tabel 4.15 menunjukkan hasil dari perhitungan kernel *polynomial* level 2, sedangkan hasil lengkapnya dapat dilihat pada lampiran G. Contoh perhitungannya, dengan data di baris ke-1 dan kolom ke-1 terdapat pada perhitungan sebagai berikut:

$$K(x1, y1) = (\,(0.8 \times 0.8) + (0.85 \times 0.85) + (0 \times 0) + (0.5 \times 0.5) + (0.9 \times 0.9) + (0 \times 0) + (0 \times 0) + (0.2 \times 0.2) + (0 \times 0) + (0 \times 0) + (0 \times 0) + (0 \times 0) + (0 \times 0) + (0.5 \times 0.5) + (0.65 \times 0.65) + (0.85 \times 0.85) + 1)^2 = 23.59530625$$

Tabel 4. 15 Hasil perhitungan kernel *polynomial* level 2

xi,yi	2	4	…	24	26	28
2	23.59530625	22.63380625	…	4.80705625	6.890625	4.0804
4	22.63380625	30.33255625	…	7.74230625	10.080625	8.8804
6	20.7936	19.8916	…	8.22255625	9.765625	7.263025
8	21.87900625	21.32130625	…	4.21275625	5.978025	3.5344
10	13.12250625	12.69140625	…	2.6569	7.317025	4.5796
20	9.10530625	14.19405625	…	12.51390625	19.580625	19.404025
22	5.62875625	8.54100625	…	12.90605625	16.58525625	18.16890625
24	4.80705625	7.74230625	…	11.91975625	10.0489	11.56
26	6.890625	10.080625	…	10.0489	22.39655625	20.36265625
28	4.0804	8.8804	…	11.56	20.36265625	24.92505625

Setelah hasil perhitungan kernel *polynomial* level 2 didapatkan, dilanjutkan dengan proses selanjutnya yaitu perhitungan matriks Hessian level 2.

B. Perhitungan matriks Hessian data latih pada level 2

Perhitungan pada tahap matriks Hessian level 2 menggunakan inisialisasi nilai *lamda* sebesar 0.5. Perhitungan matriks Hessian dilakukan dengan menggunakan persamaan 2.1. Tabel 4.16 menunjukkan hasil dari perhitungan matrks Hessian, sedangkan hasil lengkapnya dapat dilihat pada lampiran H. Contoh perhitungan matriks Hessian pada data di baris ke-1 dan kolom ke-1 terdapat pada perhitungan sebagai berikut:

$$Matriks\ Hessian_{1,1} = 1 \times 1 \times (23.59530625 + 0.5^2) = 23.84530625$$

Tabel 4. 16 Hasil perhitungan matriks Hessian level 2

M	2	4	…	24	26	28
2	23.84530625	22.88380625	…	-5.05705625	-7.140625	-4.3304
4	22.88380625	30.58255625	…	-7.99230625	-10.330625	-9.1304
6	21.0436	20.1416	…	-8.47255625	-10.015625	-7.513025
8	22.12900625	21.57130625	…	-4.46275625	-6.228025	-3.7844

10	13.37250625	12.94140625	⋯	-2.9069	-7.567025	-4.8296
20	-9.35530625	-14.44405625	⋯	12.76390625	19.830625	19.654025
22	-5.87875625	-8.79100625	⋯	13.15605625	16.83525625	18.41890625
24	-5.05705625	-7.99230625	⋯	12.16975625	10.2989	11.81
26	-7.140625	-10.330625	⋯	10.2989	22.64655625	20.61265625
28	-4.3304	-9.1304	⋯	11.81	20.61265625	25.17505625

Setelah hasil perhitungan matriks Hessian level 2 didapatkan, dilanjutkan dengan tahap selanjutnya yaitu menghitung nilai E_i.

C. Perhitungan nilai E_i iterasi 1 pada level 2

Dalam perhitungan nilai E_i pada level 2, diberikan inisialisasi awal nilai *alpha* senilai 0. Perhitungan nilai E_i dilakukan dengan menggunakan persamaan 2.2. Tabel 4.17 menunjukkan hasil perhitungan nilai E_i sedangkan contoh perhitungan nilai Ei pada data NO-P = 2 terdapat pada perhitungan sebagai berikut:

$$E_i = \big((0 \times 23.8453) + (0 \times 22.8838) + (0 \times 21.0436) + (0 \times 22.1290) + (0 \times 13.3725) + (0 \times -9.3553) + (0 \times -5.8788) + (0 \times -5.0571) + (0 \times -7.1406) + (0 \times -4.3304)\big) = 0$$

Tabel 4. 17 Hasil perhitungan nilai E_i pada level 2

NO-P	2	4	6	8	10	20	22	24	26	28	Ei
2	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	0
20	0	0	0	0	0	0	0	0	0	0	0
22	0	0	0	0	0	0	0	0	0	0	0
24	0	0	0	0	0	0	0	0	0	0	0
26	0	0	0	0	0	0	0	0	0	0	0
28	0	0	0	0	0	0	0	0	0	0	0

Setelah hasil perhitungan nilai E_i level 2 didapatkan, dilanjutkan dengan tahap selanjutnya yaitu menghitung nilai *delta alpha*.

D. Perhitungan nilai *delta alpha* iterasi 1 pada level 2

Perhitungan nilai *delta alpha* pada level 2 dilakukan dengan menggunakan persamaan 2.3. perhitungan nilai *delta alpha* membutuhkan nilai *gamma*, nilai *gamma* dapat diperoleh dari CLR atau konstanta learning rate sebesar 0.01 dibagi dengan nilai maksimum dari diagonal matriks Hessian sebesar 30.58255625. Sehingga nilai *gamma* yang didapatkan adalah 0.000326984. Pada perhitungan nilai *delta alpha* pada level 2 juga diinisialisasi nilai C sebesar 1. Tabel 4.18 menampilkan hasil dari perhitungan nilai *delta alpha*.

Contoh perhitungan nilai *delta alpha* pada level 2 dengan NO-P = 2 terdapat pada perhitungan sebagai berikut:

$$\delta\alpha_1 = \min\{\max[0.000326984(1-0), 0], 1-0\} = 0.000326984$$

Tabel 4. 18 Hasil Perhitungan nilai *delta alpha* iterasi 1 pada level 1

NO-P	Δαi
2	0.000326984
4	0.000326984
6	0.000326984
8	0.000326984
10	0.000326984
20	0.000326984
22	0.000326984
24	0.000326984
26	0.000326984
28	0.000326984

E. Perhitngan nilai *alpha* iterasi 1 pada level 2

Perhitungan nilai *alpha* iterasi 1 pada level 2 menggunakan persamaan 2.4. Hasil dari perhitungan nilai *alpha* level 2 ini ditunjukkan pada tabel 4.19 sedangkan contoh perhitungan nilai *alpha* yang baru pada data no-p 2 ditunjukkan pada perhitungan berikut:

$$\alpha_1 = 0 + 0.000326984 = 0.000326984$$

Tabel 4. 19 Hasil perhitungan nilai *alpha* iterasi 1 level 2

NO-P	αi
2	0.000326984
4	0.000326984
6	0.000326984
8	0.000326984
10	0.000326984
20	0.000326984
22	0.000326984
24	0.000326984
26	0.000326984
28	0.000326984

F. Melakukan iterasi 2 (C-D-E) dan mendapatkan nilai *alpha* maksimum pada level 2

Pada perhitungan tahap ke C, D, dan E atau tahap perhitungan nilai E_i, *delta alpha* dan nilai *alpha* akan dilakukan iterasi sesuai dengan iterasi maksimum atau saat nilai *delta alpha* mencapai batas yang ditentukan dari nilai *epsilon* (mamksimum *delta alpha* < *epsilon*). Pada perhitungan manualisasi ini telah diinisialisasikan jumlah iterasi maksimum = 2. Sehingga perhitungan pada tahap pencarian nilai E_i, *delta alpha* dan nilai *alpha* iterasi akan diulang dengan menggunakan hasil yang telah didapatkan pada perhitungan sebelumnya. Tabel 4.20 dan 4.21 merupakan hasil perhitungan nilai Ei, *delta alpha* dan nilai *alpha* yang dilakukan pada iterasi ke 2 di level 2 .Hasil lengkap perhitungan nilai Ei dapat dilihat pada lampiran H.

Tabel 4. 20 Hasil perhitungan nilai E_i iterasi 2 level 2

NOP	2	4	...	26	28	Ei
2	0.007797028	0.007482634	⋯	-0.002334869	-0.001415971	0.023383291
4	0.007482634	0.01	⋯	-0.003377947	-0.002985493	0.018779425
6	0.006880916	0.006585977	⋯	-0.003274947	-0.002456637	0.016110653
8	0.007235826	0.007053467	⋯	-0.002036463	-0.001237437	0.022329455
10	0.004372593	0.00423163	⋯	-0.002474294	-0.001579201	0.013858985
20	-0.003059033	-0.004722972	⋯	0.006484293	0.006426547	0.015068089
22	-0.001922258	-0.002874516	⋯	0.005504856	0.006022684	0.016982859
24	-0.001653575	-0.002613355	⋯	0.003367573	0.003861679	0.010236896
26	-0.002334869	-0.003377947	⋯	0.007405057	0.006740004	0.016003263
28	-0.001415971	-0.002985493	⋯	0.006740004	0.008231835	0.02160801

Tabel 4. 21 Hasil perhitungan dela *alpha* dan pembaharuan nilai *alpha* iterasi 2 level 2

NO-P	δαi	αi	Kelas (y_i)	Kategori Kelas
2	0.000326984	0.000653968	2	Positif
4	0.000326984	0.000653968	2	Positif
6	0.000326984	0.000653968	2	Positif
8	0.000326984	0.000653968	2	Positif
10	0.000326984	0.000653968	2	Positif
20	0.000326984	0.000653968	3	Negatif
22	0.000326984	0.000653968	3	Negatif
24	0.000326984	0.000653968	3	Negatif
26	0.000326984	0.000653968	3	Negatif
28	0.000326984	0.000653968	3	Negatif

Selanjutnya adalah menentukan nilai *alpha* maksimum dari masing masing kategori kelas negatif dan kategori kelas positif, kategori kelas negatif berasal dari kelas 3 sedangkan kategori kelas positif berasal dari kelas 2. Dari tabel 4.21 dapat disimpulkan bahwa nilai *alpha* maksimal yang diperoleh dari kategori kelas negatif adalah 0.000653968 dari no-p 20, dan nilai *alpha* maksimal yang diperoleh dari kelas positif adalah 0.000653968 dari no-p 2. Nilai *alpha* maksimal selanjutnya akan digunakan dalam perhitungan nilai bobot pada level 2, yang akan di bahas pada tahap berikutnya.

G. Perhitungan nilai bobot pada level 2

Sebelum menghitung bobot pada level 2, dibutuhkan perhitungan kernel dari setiap data terhadap data latih dengan nilai *alpha* tertinggi pada kategori kelas positif atau data dengan no-p 2 ($K(x_i,x^+)$), dan perhitungan kernel dari setiap data latih terhadap data dengan nilai *alpha* tertinggi pada kategori kelas negatif atau data dengan no-p 20 ($K(x_i,x^-)$). Tabel 4.22 menunjukkan hasil dari perhitungan $K(x_i,x^+)$ dan $K(x_i,x^-)$ pada level 2. Berikut contoh perhitungan kernel dari data dengan no-p 2 terhadap data dengan nilai *alpha* tertinggi dari kategori kelas positif dan kernel dari data dengan no p-20 terhadap nilai *alpha* tertinggi dari kategori kelas negatif pada tahap ini:

$$K(x_2, x^+) = (\ (0.8 \times 0.8) + (0.85 \times 0.85) + (0 \times 0) + (0.5 \times 0.5) + (0.9 \times 0.9) + (0 \times 0) + (0 \times 0) + (0.2 \times 0.2) + (0 \times 0) + (0 \times 0) + (0 \times 0) + (0 \times 0) + (0 \times 0) + (0.5 \times 0.5) + (0.65 \times 0.65) + (0.85 \times 0.85) + 1)^2 = 23.59530625$$

$$K(x_{20}, x^-) = (\ (0.8 \times 0) + (0.85 \times 0) + (0 \times 0.75) + (0.5 \times 0.5) + (0.9 \times 0.7) + (0 \times 0.75) + (0 \times 0.65) + (0.2 \times 0.2) + (0 \times 0.8) + (0 \times 0.7) + (0 \times 0.4) + (0 \times 0) + (0.5 \times 0.5) + (0.65 \times 0.65) + (0.85 \times 0) + 1))\text{^}2 = 9.10530625$$

Tabel 4. 22 Hasil dari perhitungan $K(x_i,x^+)$ dan $K(x_i,x^-)$ level 2

NO-P	10	24
2	23.59530625	9.10530625
4	22.63380625	14.19405625
6	20.7936	11.594025
8	21.87900625	8.28000625
10	13.12250625	9.78125625
20	9.10530625	30.00300625
22	5.62875625	17.53515625
24	4.80705625	12.51390625
26	6.890625	19.580625
28	4.0804	19.404025

Selanjutnya Perhitungan nilai bobot pada level 2 akan dilakukan dengan menghitung masing masing bobot pada data ke 1 sampai ke 10 dengan menggunakan persamaan 2.5, untuk setiap data tersebut memiliki dua hasil perhitungan bobot. Hasil perhitungan bobot pertama ($w.x^-$) didapatkan dari perkalian antara nilai *alpha* terakhir data (αi), kelas data (y_i) dan hasil perhitungan kernel dari data tersebut terhadap data dengan nilai *alpha* tertinggi pada kategori kelas negatif atau data dengan no-p 20 ($K(x_i,x^-)$). Hasil perhitungan bobot kedua ($w.x^+$) didapatkan dari perkalian antara nilai *alpha* terakhir data (αi), kelas data (y_i) dan hasil perhitungan kernel dari data tersebut terhadap data dengan nilai *alpha* tertinggi pada kategori kelas positif atau data dengan no-p 2 ($K(x_i,x^+)$). Tabel 4.23 menunjukkan hasil dari perhitungan ($w.x^+$) dan ($w.x^-$). Berikut contoh perhitungan bobot kategori kelas positif ($w.x^+$) dan bobot kategori kelas negatif ($w.x^-$) dari data dengan no-p 2:

$$w.x^+ = 0.000653968 \times 1 \times 23.59530625 = 0.015430565$$

$$w.x^- = 0.000653968 \times 1 \times 9.10530625 = 0.005954575$$

Tabel 4. 23 Hasil Perhitungan Bobot ($w.x^+$) dan ($w.x^-$) level 2

NO-P	W.X+	W.X-
2	0.015430565	0.005954575
4	0.014801775	0.009282452
6	0.01359834	0.007582116
8	0.014308161	0.005414856
10	0.008581694	0.006396624
20	-0.005954575	-0.019620993
22	-0.003681024	-0.011467424
24	-0.003143659	-0.008183689
26	-0.004506245	-0.012805094
28	-0.002668449	-0.012689603
∑	0.046766582	-0.030136179

Selanjutnya nilai bobot pada level 2 yang telah didapat akan digunakan dalam perhitungan bias pada level 2 yang akan dibahas dalam tahap selanjutnya.

H. Perhitungan nilai bias pada level 2

Perhitungan nilai bias pada level 2 didapatkan dengan menggunakan persamaan 2.6. Untuk mendapatkan nilai tersebut, maka dilakukan penjumlahan antara seluruh nilai bobot positif dengan seluruh nilai bobot negatif, kemudian dikali dengan $-\frac{1}{2}$. Dari tabel 4.23 dapat disimpulkan bahwa jumlah seluruh nilai bobot positif adalah 0.046766582 dan jumlah seluruh nilai bobot negatif adalah --0.030136179. Berikut contoh perhitungan nilai bias pada perhitungan manualisasi ini:

$$nilai\ bias = -\frac{1}{2} \times (0.046766582 + -0.030136179) = -0.008315202$$

Selanjutnya nilai bias pada level 2 akan digunakan dalam proses perhitungan nilai f(x) pada level 2 yang akan dibahas pada tahap berikutnya.

I. Perhitungan nilai f(X)

Perhitungan nilai f(x) pada level 2 menggunakan perhitungan pada persamaan 2.10, sebelum menghitung nilai f(x) perlu dilakukan perhitungan kernel dari setiap data latih pada level 2 terhadap data uji yang digunakan pada level 2. Contoh perhitungan kernel untuk data latih no-p 2 terhadap data uji no-p 101 di level 2:

$$K(x_2, x_{101}) = ((0.8 \times 0) + (0.85 \times 0) + (0 \times 0) + (0.5 \times 0.2) + (0.9 \times 0.9) + (0 \times 0.9) + (0 \times 0.6) + (0.2 \times 0.2) + (0 \times 0.4) + (0 \times 0.7) + (0 \times 0) + (0 \times 0) + (0.5 \times 0.5) + (0.65 \times 0.65) + (0.85 \times 0.85) + 1)^2 = 6.87750625$$

Tabel 4.24 menunjukkan hasil dari perhitungan kernel data latih terhadap data uji pada level 2, yang mana hasil tersebut akan digunakan dalam perhitungan bobot data uji level 2.

Tabel 4. 24 Hasil perhitungan kernel data latih terhadap data uji level 2

NO-P	101	102
	K(X_latih,X101)	K(X_latih,X102)
2	6.87750625	23.59530625
4	10.06475625	22.63380625
6	10.87350625	20.7936
8	5.96580625	21.87900625
10	4.5369	15.86030625
20	18.38265625	11.40750625
22	18.33980625	8.30880625
24	12.90605625	4.80705625
26	16.8921	9.828225
28	18.49	6.4009

Setelah mendapatkan nilai kernel, maka perhitungan selanjutnya adalah, perhitungan bobot data uji level 2. Perhitungan bobot data uji level 2 didapatkan dengan cara melakukan perkalian antara nilai *alpha* terakhir pada level 2 dari data tersebut dengan kelas data tersebut dan juga dengan kernel data latih pada level 2 terhadap data uji pada level 2. Tabel 4.25 menunjukkan hasil dari perhitungan bobot setiap data. Contoh dari perhitungan bobot untuk data 2 terhadap data uji 101 di level 2 sebagai berikut:

$$\alpha_i y_i K(\mathrm{x}_2, x_{101}) = 0.000653968 \times 1 \times 6.87750625 = 0.004497666$$

Tabel 4. 25 Hasil perhitungan penjumlahan bobot pada setiap data

NO-P	101	102
	BOBOT DATA UJI 1	BOBOT DATA UJI 2

2	0.004497666	0.015430565
4	0.006582024	0.014801775
6	0.00711092	0.01359834
8	0.003901444	0.014308161
10	0.002966985	0.010372126
20	-0.012021661	-0.007460139
22	-0.011993639	-0.00543369
24	-0.008440142	-0.003143659
26	-0.011046886	-0.00642734
28	-0.01209186	-0.004185981
∑	-0.030535148	0.041860158

Selanjutnya jumlah dari bobot data uji pada data 101, dan 102 digunakan untuk mencari fungsi f(x), nilai f(x) didapatkan dari penjumlahan antara jumlah bobot masing-masing data uji dengan nilai bias yaitu -0.008315202, sebagaimana yang telah dituliskan pada persamaan 2.10. Hasil perhitungan dari nilai f(x) level 2 dapat dilihat pada tabel 4. 26. Contoh perhitungan dalam mendapatkan nilai f(x) pada data uji 101 pada level 2 sebagai berikut :

$$f(x) = -0.030535148 + (-0.008315202) = -0.03885035$$

$$f(x) = sign\,(-0.03885035) = -1$$

Tabel 4. 26 Hasil dari perhitungan nilai f(x) pada level 2

NO-P	F(X)	FUNGSI KLASIFIKASI	KELAS ASLI
101	-0.03885035	-1	3
102	0.033544956	1	2

Dapat diketahui dari tabel 4.26 bahwa pada fungsi klasifikasi, no-p 102 mendapatkan hasil berupa nilai 1 yang berarti data tersebut terklasifikasi sebagai kelas 2. Sedangkangkan pada fungsi klasifikasi dengan no-p 101 mendapatkan hasil berupa nilai -1 yang berarti bahwa data tersebut terklasifikasi sebagai kelas selain kelas 2 atau kelas 3.

11. Mendapatkan Hasil Klasifikasi

Hasil klasifikasi dari perhitungan *Support Vector Machine* dapat dilihat dari hasil perhitungan f(x) level 1 untuk menentukan kelas yang masuk pada kelas 1 dan dapat dilihat juga dari perhitungan f(x) level 2 untuk menentukan kelas yang masuk pada kelas 2 dan 3. Dari perhitungan Support Vector Machine pada manualisasi SVM ini, telah didapatkan hasil bahwa data uji dengan no-p 103, 108, 104 dan 106 terklasifikasi ke kelas 1, data uji denga no-p 102 terklasifikasi ke kelas 2, sedangkan data uji dengan no-p 101 terklasifikasi ke kelas 3. Tabel seluruh hasil klasifikasi yang didapatkan pada perhitungan klasifikasi penyakit dengan gejala demam menggunakan Support Vector Machine dapat dilihat pada tabel 4.27.

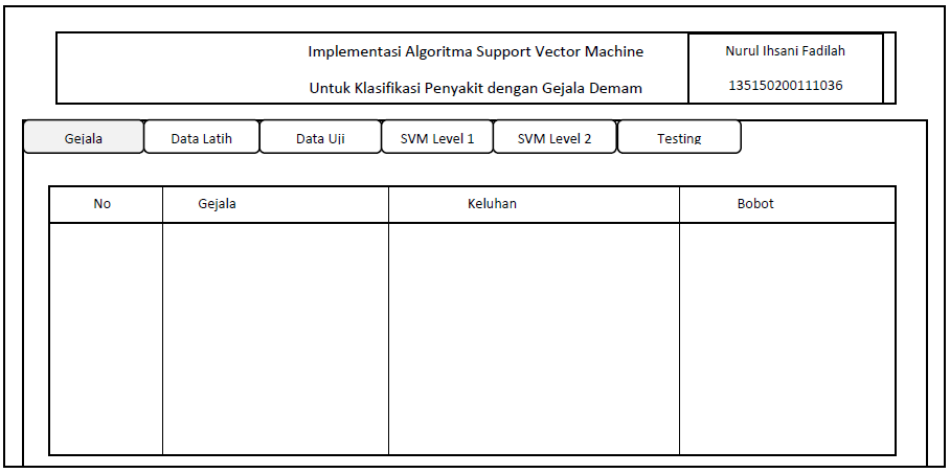
Tabel 4. 27 Hasil keseluruhan dari perhitungan SVM

DT_UJI	F(X) LEVEL 1	F(X) LEVEL 2	HASIL KLASIFIKASI	KELAS ASLI	Keterangan
101	-1	-1	3	3	Benar
103	1	-	1	3	Salah
102	-1	1	2	2	Benar
108	1	-	1	2	Salah
104	1	-	1	1	Benar
106	1	-	1	1	Benar

4.4 Perancangan Antarmuka

4.4.1 Antarmuka Halaman Gejala

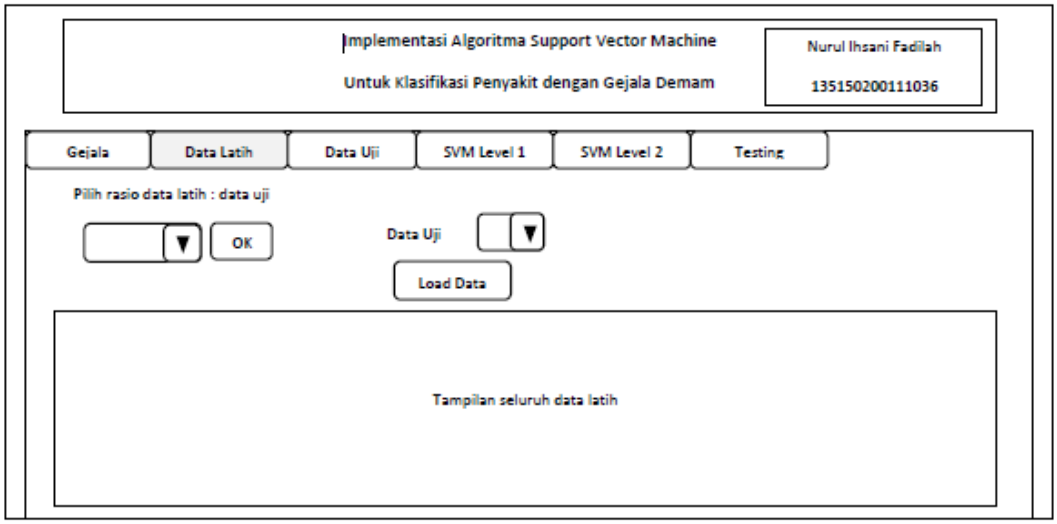
Antarmuka Halaman Gejala berisi tentang penjelasan daftar gejala penyakit yang terdapat pada program klasifikasi penyakit dengan gejala demam menggunakan algoritme *Support Vector Machine.* Perancangan antarmuka halaman gejala yang akan dibuat dapat dilihat pada Gambar 4.13



Gambar 4. 13 Rancangan Halaman Gejala

4.4.2 Antarmuka Halaman Data Latih

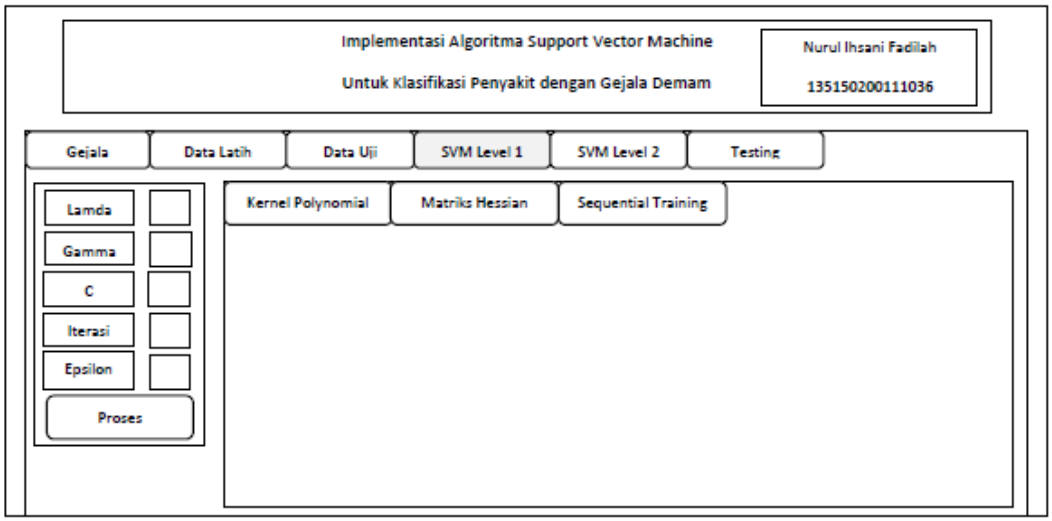
Antarmuka Halaman Data Set berisi data latih yang digunakan dalam program klasifikasi penyakit dengan gejala demam menggunakan algoritme *Support Vector Machine.* Perancangan antarmuka halaman data latih yang akan dibuat dapat dilihat pada Gambar 4.14



Gambar 4. 14 Rancangan halaman data latih

4.4.3 Antarmuka Halaman SVM Level 1

Antarmuka Halaman SVM Level 1 berisi hasil dari perhitungan SVM Level 1. Terdiri dari perhitungan kernel *polynomial*, matiks Hessian, perhitungan *sequential training* dari program klasifikasi penyakit dengan gejala demam menggunakan algoritme *Support Vector Machine.* Perancangan antarmuka halaman SVM level 1 yang akan dibuat dapat dilihat pada Gambar 4.15

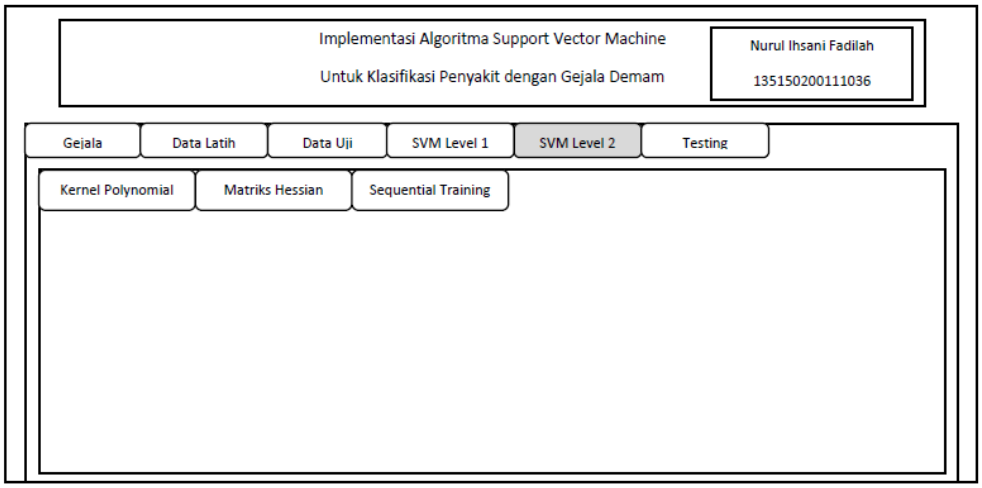


Gambar 4. 15 Rancangan halaman SVM Level 1

4.4.4 Antarmuka Halaman SVM Level 2

Antarmuka Halaman SVM Level 2 berisi hasil dari perhitungan SVM Level 2. Terdiri dari perhitungan kernel *polynomial*, matriks Hessian, perhitungan *sequential training* dari program klasifikasi penyakit dengan gejala demam

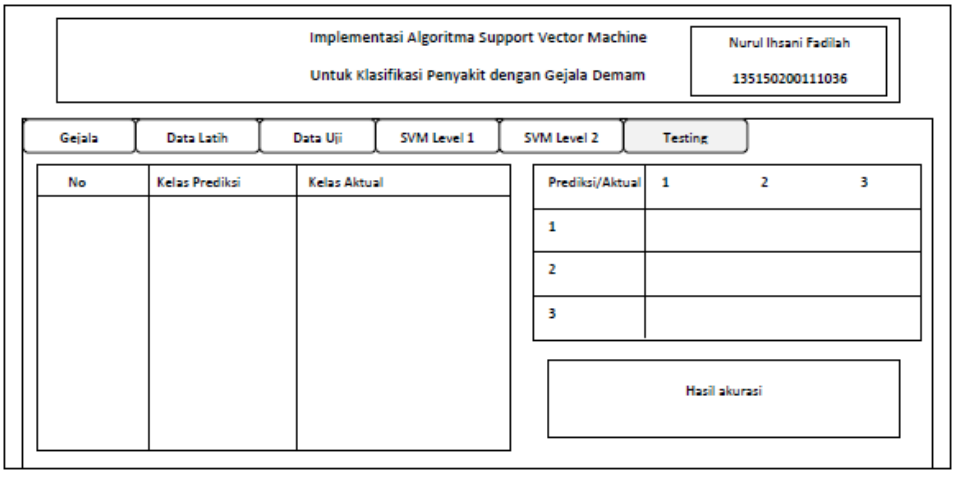
menggunakan algoritme *Support Vector Machine.* Perancangan antarmuka halaman SVM level 2 yang akan dibuat dapat dilihat pada Gambar 4.16



Gambar 4. 16 Rancangan Halaman SVM Level 2

4.4.5 Antarmuka Halaman *Testing*

Antarmuka Halaman *testing* berisi hasil akhir dari seluruh perhitungan baik hasil dari perhitungan SVM level 1 maupun hasil dari perhitungan SVM level 2. Terdapat keterangan dari hasil prediksi kelas, keterangan kelas aktualnya serta keterangan dari hasil akurasi program klasifikasi penyakit dengan gejala demam menggunakan algoritme *Support Vector Machine.* Perancangan antarmuka halaman *testing* yang akan dibuat dapat dilihat pada Gambar 4.17



Gambar 4. 17 Rancangan Halaman *Testing*

4.5 Perancangan Pengujian

Pada penelitian ini, perancangan pengujian akan dilakukan dalam rangka menampilkan akurasi sistem dalam sebuah tabel data, yang mana dilakukan beberapa skenario dalam pengujian. Pada setiap skenario penelitian, dilakukan pula percobaan yang diulang dalam beberapa perlakuan yang berbeda. Setelah dilakukan pengujian akan dianalisis bagaimana tingkat akurasi yang akan di dapatkan dalam penelitian implementasi *Support Vector Machine* dalam klasifikasi penyakit dengan gejala demam.

4.5.1 Pengujian Pengaruh Perbandingan Jumlah Data *Training* dan Data *Testing* Terhadap Nilai Akurasi

Pengujian pengaruh perbandingan jumlah data *training* dan data *testing* terhadap nilai akurasi dilakukan untuk mengetahui adakah pengaruh perubahan data *training* dan data *testing* pada percobaan yang telah dibuat, dalam perubahan nilai akurasi. Pengujian ini menggunakan *K-Fold Cross Validation* dimana setiap pembagian rasio data latih dan data uji yang berbeda menggunakan k yang berbeda pula. Pembagian data latih data uji yang digunakan adalah 90%:10% dengan k=10, 80%:20% dengan k=5, dan 50%:50% dengan k=2. Perancangan dari pengujian tersebut dapat dapat dilihat pada Tabel 4.28.

Tabel 4.28 Rancangan pengujian pengaruh perbandingan jumlah data *training* dan data *testing* terhadap nilai akurasi

Perbandingan Data Latih : Data Uji	Dataset k-*fold* ke-										Rata-Rata Akurasi
	1	2	3	4	5	6	7	8	9	10	
90% : 10%	%	%	%	%	%	%	%	%	%	%	%
Perbandingan Data Latih : Data Uji	Dataset k-*fold* ke-										Rata-Rata Akurasi
	1		2		3		4		5		
80% : 20%	%	%	%	%	%	%	%	%	%	%	%
Perbandingan Data Latih : Data Uji	Dataset k-*fold* ke-										Rata-Rata Akurasi
	1					2					
50% : 50	%					%					%

4.5.2 Pengujian Pengaruh Nilai *Lamda* (λ) Terhadap Nilai Akurasi

Pengujian pengaruh nilai *lamda* (λ) terhadap nilai akurasi dilakukan untuk mengetahui adakah pengaruh dari perubahan nilai *lamda* (λ) pada percobaan yang telah dilakukan, dalam perubahan nilai akurasi. Pengujian ini menggunakan *K-Fold Cross Validation* dimana pembagian rasio data latih dan data uji yang digunakan

adalah 90%:10% dengan k=10. Perancangan dari pengujian tersebut dapat dapat dilihat pada Tabel 4.29.

Tabel 4. 29 Rancangan pengujian pengaruh nilai *Lamda* (λ) terhadap nilai akurasi

Nilai *Lamda* (λ)	Dataset k-*fold* ke-										Rata-Rata Akurasi
	1	2	3	4	5	6	7	8	9	10	
0.5	%	%	%	%	%	%	%	%	%	%	%
1	%	%	%	%	%	%	%	%	%	%	%
1.5	%	%	%	%	%	%	%	%	%	%	%
2	%	%	%	%	%	%	%	%	%	%	%
2.5	%	%	%	%	%	%	%	%	%	%	%
3	%	%	%	%	%	%	%	%	%	%	%
5	%	%	%	%	%	%	%	%	%	%	%
10	%	%	%	%	%	%	%	%	%	%	%
15	%	%	%	%	%	%	%	%	%	%	%
20	%	%	%	%	%	%	%	%	%	%	%

4.5.3 Pengujian Pengaruh Nilai *Gamma* (*γ*) Terhadap Nilai Akurasi

Pengujian pengaruh nilai *gamma* (γ) terhadap nilai akurasi dilakukan untuk mengetahui adakah pengaruh dari perubahan nilai *gamma* (γ) pada percobaan yang telah dilakukan, dalam perubahan nilai akurasi. Pengujian ini menggunakan *K-Fold Cross Validation* dimana pembagian rasio data latih dan data uji yang digunakan adalah 90%:10% dengan k=10. Perancangan dari pengujian tersebut dapat dapat dilihat pada Tabel 4.30.

Tabel 4.30 Rancangan pengujian pengaruh nilai *Gamma* (*γ*) terhadap nilai akurasi

Nilai *Gamma* (*γ*)	Dataset k-*fold* ke-										Rata-Rata Akurasi
	1	2	3	4	5	6	7	8	9	10	
0.000000001	%	%	%	%	%	%	%	%	%	%	%
0.00000001	%	%	%	%	%	%	%	%	%	%	%
0.0000001	%	%	%	%	%	%	%	%	%	%	%
0.000001	%	%	%	%	%	%	%	%	%	%	%
0.00001	%	%	%	%	%	%	%	%	%	%	%

0.0001	%	%	%	%	%	%	%	%	%	%	%
0.001	%	%	%	%	%	%	%	%	%	%	%
0.01	%	%	%	%	%	%	%	%	%	%	%
0.1	%	%	%	%	%	%	%	%	%	%	%
1	%	%	%	%	%	%	%	%	%	%	%

4.5.4 Pengujian Pengaruh Nilai C (*Complexity*) Terhadap Akurasi

Pengujian pengaruh nilai C (*complexity*) terhadap nilai akurasi dilakukan untuk mengetahui adakah pengaruh dari perubahan nilai C (*complexity*) pada percobaan yang telah dilakukan, dalam perubahan nilai akurasi. Pengujian ini menggunakan *K-Fold Cross Validation* dimana pembagian rasio data latih dan data uji yang digunakan adalah 90%:10% dengan k=10. Perancangan dari pengujian tersebut dapat dapat dilihat pada Tabel 4.31.

Tabel 4.31 Rancangan pengujian pengaruh nilai C (Complexity) terhadap akurasi

Nilai C (*Complexity*)	Dataset k-*fold* ke-										Rata-Rata Akurasi
	1	2	3	4	5	6	7	8	9	10	
1	%	%	%	%	%	%	%	%	%	%	%
2	%	%	%	%	%	%	%	%	%	%	%
3	%	%	%	%	%	%	%	%	%	%	%
4	%	%	%	%	%	%	%	%	%	%	%
5	%	%	%	%	%	%	%	%	%	%	%
10	%	%	%	%	%	%	%	%	%	%	%
20	%	%	%	%	%	%	%	%	%	%	%
30	%	%	%	%	%	%	%	%	%	%	%
40	%	%	%	%	%	%	%	%	%	%	%
50	%	%	%	%	%	%	%	%	%	%	%

4.5.5 Pengujian Pengaruh Nilai Iterasi Maksimum Terhadap Nilai Akurasi

Pengujian pengaruh nilai iterasi maksimum terhadap nilai akurasi dilakukan untuk mengetahui adakah pengaruh dari perubahan nilai iterasi maksimum pada percobaan yang telah dilakukan, dalam perubahan nilai akurasi. Pengujian ini menggunakan *K-Fold Cross Validation* dimana pembagian rasio data latih dan data

uji yang digunakan adalah 90%:10% dengan k=10. Perancangan dari pengujian tersebut dapat dapat dilihat pada Tabel 4.32.

Tabel 4.32 Rancangan pengujian pengaruh nilai Iterasi maksimum terhadap nilai akurasi

Nilai Iterasi Maksimum	Dataset k-*fold* ke-										Rata-Rata Akurasi
	1	2	3	4	5	6	7	8	9	10	
5	%	%	%	%	%	%	%	%	%	%	%
10	%	%	%	%	%	%	%	%	%	%	%
20	%	%	%	%	%	%	%	%	%	%	%
40	%	%	%	%	%	%	%	%	%	%	%
50	%	%	%	%	%	%	%	%	%	%	%
100	%	%	%	%	%	%	%	%	%	%	%
500	%	%	%	%	%	%	%	%	%	%	%
1000	%	%	%	%	%	%	%	%	%	%	%
1500	%	%	%	%	%	%	%	%	%	%	%
2000	%	%	%	%	%	%	%	%	%	%	%

4.5.6 Pengujian Pengaruh Nilai *Epsilon* (ε) Terhadap Nilai Akurasi

Pengujian pengaruh nilai *epsilon* terhadap nilai akurasi dilakukan untuk mengetahui adakah pengaruh dari perubahan nilai *epsilon* pada percobaan yang telah dilakukan, dalam perubahan nilai akurasi. Pengujian ini menggunakan *K-Fold Cross Validation* dimana pembagian rasio data latih dan data uji yang digunakan adalah 90%:10% dengan k=10. Perancangan dari pengujian tersebut dapat dapat dilihat pada Tabel 4.33.

Tabel 4. 33 Rancangan pengujian pengaruh nilai *epsilon* terhadap nilai akurasi

Nilai *Epsilon*	Dataset k-*fold* ke-										Rata-Rata Akurasi
	1	2	3	4	5	6	7	8	9	10	
0.000000001	%	%	%	%	%	%	%	%	%	%	%
0.00000001	%	%	%	%	%	%	%	%	%	%	%
0.0000001	%	%	%	%	%	%	%	%	%	%	%
0.000001	%	%	%	%	%	%	%	%	%	%	%
0.00001	%	%	%	%	%	%	%	%	%	%	%
0.0001	%	%	%	%	%	%	%	%	%	%	%

0.001	%	%	%	%	%	%	%	%	%	%	%
0.01	%	%	%	%	%	%	%	%	%	%	%
0.1	%	%	%	%	%	%	%	%	%	%	%
1	%	%	%	%	%	%	%	%	%	%	%

4.5.7 Pengujian *K-fold Cross Validation*

Pengujian *K-fold Cross Validation* dilakukan dengan menggunakan pembagian rasio data latih dan data uji yaitu 90%:10% dengan k=10, kemudian digunakan parameter terbaik yang telah didapatkan pada pengujian sebelumnya dari pengujian nilai *lamda*, nilai *gamma*, nilai *epsilon*, iterasi maksimum, dan nilai C. Pengujian ini dilakukan untuk mengetahui bagaimana perbandingan akurasi pada setiap *fold* dalam 10-*fold cross validation* sehingga dapat dilihat bagaimana kestabilan akurasi yang didapatkan. Perancangan dari pengujian tersebut dapat dilihat pada Tabel 4.34

Tabel 4. 34 Rancangan pengujian *k-fold cross validation*

Dataset k-*fold* ke-										Rata-Rata Akurasi
1	2	3	4	5	6	7	8	9	10	
%	%	%	%	%	%	%	%	%	%	%

BAB 5 IMPLEMENTASI

Pada bab ini berisi penjabaran tentang bagaimana mengimplementasikan sebuah sistem yang didasarkan dari kebutuhan serta proses perancangan yang telah dijabarkan pada bab sebelumnya. Pada bab ini akan dijelaskan tentang implementasi algotitma *Support Vector Machine* dalam mengklasifikasi penyakit dengan gejala demam.

5.1 Spesifikasi Sistem

Spesifikasi sistem akan diuraikan menjadi dua spesifikasi. Pertama adalah spesifikasi terkait perangkat keras yang digunakan dalam membangun penelitian ini, yang kedua adalah spesifikasi terkait perangkat lunak yang digunakan dalam membangun penelitian ini.

5.1.1 Spesifikasi Perangkat Keras

Spesifikasi pada lingkup implementasi perangkat keras yang digunakan dalam membangun penelitian ini terdapat pada tabel yang ditunjukkan pada Tabel 5.1.

Tabel 5. 1 Spesifikasi Perangkat Keras

Komponen	Spesifikasi
Processor	Intel (R) Core (TM) i5-4200U @ 1.60GHz (4CPUs), ~2.3GHz
Memory	4096MB RAM
Harddisk	700 GB

5.1.2 Spesifikasi Perangkat Lunak

Spesifikasi pada lingkup implementasi perangkat lunak yang digunakan dalam membangun penelitian ini terdapat pada tabel yang ditunjukkan pada Tabel 5.2.

Tabel 5. 2 Spesifikasi Perangkat Lunak

Tipe Perangkat Lunak	Spesifikasi
Sistem Operasi	Microsoft Windows 10 Pro 64-*bit*
Bahasa Pemrograman	JAVA
Tools Pemrograman	Netbeans IDE 8.0.2

5.2 Batasan Implementasi

Batasan implementasi yang terdapat dalam membangun sistem penelitian ini adalah :

1. Penelitian ini menggunakan kriteria sejumlah 15 kriteria yang akan digunakan dalam perhitungan SVM
2. Data yang digunakan dalam pengembangan penelitian ini adalah data dari penelitian sebelumnya sebanyak 130 data
3. Implementasi SVM menggunakan *kernel polynomial* degree dengan nilai d=2

5.3 Implementasi Algoritme *Suport Vector Machine*

Pada sub bab ini akan di isi dengan hasil dari implementasi program dari algoritme *Support Vector Machine*, dimana implementasi tersebut terdiri dari implementasi berbagai macam perhitungan yang terdapat pada perhitungan *Support Vector Machine*, diantaranya adalah : implementasi perhitungan kernel *polynomial*, implementasi perhitungan matriks Hessian, implementasi perhitungan Ei, implementasi perhitungan *delta alpha*, implementasi perhitungan *alpha*, implementasi perhitunga bobot $w.x^+$ dan $w.x^-$, implementasi perhitungan nilai bias, serta implementasi dari perhitungan nilai f(x).

5.3.1 Implementasi Perhitungan Kernel *Polynomial*

Implementasi perhitungan kernel *polynomial* terdapat pada Kode Program 5.1. Perhitungan ini merupakan perhitungan awal yang dilakukan pada implementasi program dalam mengklasifikasi penyakit dengan gejala demam menggunakan algorita *Suport Vector Machine*. Perhitungan kernel *polynomial* merupakan perkalian *dot product* dari dua data *training* yang berbeda dengan menghasilkan matriks n x n, dimana n merupakan jumlah data *training* yang digunakan.

```
1     Public double[][] kernelPolinomial(double[][] data1,
2      double[][] data2) {
3         System.out.println("Kernel");
4         double[][] hasilKernel = new
5          double[data1.length][data2.length];
6         for (int i = 0; i < data1.length; i++) {
7             for (int j = 0; j < data2.length; j++) {
8                 double total = 0;
9                 for (int panjang = 0; panjang < data1[0].length;
10                 panjang++) {
11                    total = total + data1[i][panjang] *
12                     data2[j][panjang];
13                }
14                hasilKernel[i][j] = Math.pow(total + c, degree);
15                System.out.print(hasilKernel[i][j] + "\t");
16            }
17            System.out.println("");
18        }
19        return hasilKernel;
20    }
```

Kode Program 5. 1 Implementasi perhitungan kernel *polynomial*

Penjelasan dari Kode Program 5.1 mengenai perhitungan kernel *polynomial* sebagai berikut :

1. Baris ke 4-5 digunakan untuk inisialisasi variable.

2. Baris ke 6-10 digunakan untuk perulangan mengambil nilai data yang akan dihitung.
3. Baris ke 11-20 digunakan untuk menghitung rumus kernel *polynomial* pada data kemudian mencetak hasil dari kernel *polynomial*.

5.3.2 Implementasi Perhitungan Mariks Hessian

Implementasi perhitungan matriks Hessian terdapat pada Kode Program 5.2. Perhitungan ini diproses setelah mendapatkan hasil dari perhitungan kernel *polynomial*. Perhitungan matriks Hessian dilakukan dengan melakukan perkalian antara kelas dari dua data *training* dan kernel *polynomial* di tambah dengan nilai *lamda* yang dikuadratkan.

```
1     Double[][] Hessian(double[] kelas, double[][] hasilKernel) {
2         System.out.println("Hessian");
3         double[][] matriksHessian = new
4          double[hasilKernel.length][hasilKernel[0].length];
5         for (int i = 0; i < hasilKernel.length; i++) {
6             for (int j = 0; j < hasilKernel[0].length; j++) {
7                 matriksHessian[i][j] = kelas[i] * kelas[j] *
8                   (hasilKernel[i][j] + Math.pow(lamda, 2));
9                 System.out.print(matriksHessian[i][j] + "\t");
10            }
11            System.out.println("");
12        }
13        return matriksHessian;
14    }
```

Kode Program 5. 2 Implementasi perhitungan matriks Hessian

Penjelasan dari Kode Program 5.2 mengenai perhitungan matriks Hessian sebagai berikut :

1. Baris ke 3-4 digunakan untuk inisialisasi variable.
2. Baris ke 5-6 digunakan untuk perulangan mengambil nilai data yang akan dihitung.
3. Baris ke 7-14 digunakan untuk menghitung rumus matriks Hessian pada data kemudian mencetak hasil dari matriks Hessian.

5.3.3 Implementasi Perhitungan Ei

Implementasi perhitungan Ei terdapat pada Kode Program 5.3. Perhitungan ini dilakukan setelah mendapatkan hasil dari perhitungan matriks Hessian. Perhitungan Ei pada setiap kolom dilakukan dengan melakukan penjumlahan nilai *alpha* dan matriks Hessian kemudian di total pada setiap baris.

```
1     double[] hitungError(double alpha[], double
2      matriksHessian[][]) {
3         double error[] = new double[matriksHessian.length];
4         System.out.println("Error");
5         for (int i = 0; i < matriksHessian.length; i++) {
6             double total = 0;
```

```
7               for (int j = 0; j < matriksHessian.length; j++) {
8                   total = total + alpha[i] * matriksHessian[i][j];
9               }
10              error[i] = total;
11              System.out.println(error[i]);
12          }
13          return error;
14      }
```

Kode Program 5. 3 Implementasi perhitungan Ei

Penjelasan dari Kode Program 5.3 mengenai perhitungan Ei sebagai berikut :

1. Baris ke 3 digunakan untuk inisialisasi variable.
2. Baris ke 5-7 digunakan untuk perulangan mengambil nilai data yang akan dihitung.
3. Baris ke 8-14 digunakan untuk menghitung rumus penjumlahan seluruh nilai Ei pada setiap data kemudian mencetak hasil dari Ei.

5.3.4 Implementasi Perhitungan *Delta Alpha*

Implementasi perhitungan *delta alpha* terdapat pada Kode Program 5.4. Perhitungan ini dilakukan setelah mendapatkan hasil dari perhitungan nilai Ei pada setiap data. Perhitungan *delta alpha* pada setiap data dilakukan dengan mencari nilai minimum antara pre*deltaAlpha* dengan C dikurang *alpha* setiap data, yang mana pre*deltaAlpha* merupakan pencarian nilai maksimum antara perkalian *gamma* dengan hasil dari 1 dikurang nilai Ei yang dibandingkan dengan nilai *alpha* dari setiap data.

```
1       public double[] hitung DeltaAlpha(double error[], double
2        alpha[], double matriksHessian[][]) {
3           double[] deltaA = new double[error.length];
4           double terbesar = 0;
5           System.out.println("cekkk");
6           for (int i = 0; i < matriksHessian.length; i++) {
7               System.out.println(matriksHessian[i][i]);
8               if (matriksHessian[i][i] > terbesar) {
9                   terbesar = matriksHessian[i][i];
10              }
11          }
12          System.out.println("terbesar " + terbesar);
13          double gamma = clr / terbesar;
14          for (int i = 0; i < error.length; i++) {
15              double predeltaAlpha = Math.max(gamma * (1 -
16               error[i]), alpha[i]);
17              deltaA[i] = Math.min(predeltaAlpha, C - alpha[i]);
18          }
19          System.out.println("akhir delta alpha");
20          return deltaA;
21      }
```

Kode Program 5. 4 Implementasi perhitungan *delta alpha*

Penjelasan dari Kode Program 5.4 mengenai perhitungan *delta alpha* sebagai berikut :

1. Baris ke 3-4 digunakan untuk inisialisasi variable.

2. Baris ke 6-9 digunakan untuk perulangan mengambil nilai terbesar dari diagonal matriks Hessian yang akan digunakan untuk menentukan nilai *gamma*.
3. Baris ke 13 digunakan untuk menghitung rumus *gamma* (nilai *gamma* ditentukan dari CLR dibagi nilai terbesar dari diagonal matriks Hessian).
4. Baris ke 14 digunakan untuk perulangan mengambil nilai data yang akan dihitung.
5. Baris ke 15-20 digunakan untuk menghitung rumus *delta alpha* pada data kemudian mencetak hasil dari *delta alpha*.

5.3.5 Implementasi Perhitungan *Alpha*

Implementasi perhitungan nilai *alpha* terdapat pada Kode Program 5.5. Perhitungan ini dilakukan setelah mendapatkan hasil dari perhitungan nilai *delta alpha* pada setiap data. Perhitungan nilai *alpha* didapatkan dengan menjumlahkan nilai *alpha* awal atau nilai *delta* alpa sebelumnya dengan hasil dari perhitungan *delta alpha*.

```
1     double[] hitungAlpha(double deltaAlpha[], double alpha[]) {
2         double alpha[] = new double[dataLatih.length];
3         for (int i = 0; i < dataLatih.length; i++) {
4             alpha[i] = 0;
5         }
6 //update alpha
7         double alphaBaru[] = new double[deltaAlpha.length];
8         for (int i = 0; i < deltaAlpha.length; i++) {
9             alphaBaru[i] = alpha[i] + deltaAlpha[i];
10        }
11        return alphaBaru;
12    }
```

Kode Program 5. 5 Implementasi perhitungan *alpha*

Penjelasan dari Kode Program 5.5 mengenai perhitungan *alpha* sebagai berikut :

1. Baris ke 2 dan 7 digunakan untuk inisialisasi variable.
2. Baris ke 3-5 digunakan untuk perulangan menghitung nilai *alpha* pertama.
3. Baris ke 8-12 digunakan untuk menghitung rumus *alpha* terupdate setelah perulangan ke-sekian.

5.3.6 Implementasi Perhitungan Bobot $w.x^+$ dan $w.x^-$ serta Nilai Bias

Implementasi perhitungan bobot $w.x^+$ dan $w.x^-$ serta nilai bias terdapat pada Kode Program 5.6. Perhitungan ini dilakukan setelah mendapatkan hasil dari perhitungan nilai *alpha*. Perhitungan bobot dimulai dengan mencari hasil kernel dari data yang memiliki nilai *alpha* maksimal pada kelas positif dan kernel dari data yang memiliki nilai *alpha* maksimal pada kelas negatif. Perhitungan bobot $w.x^+$

didapatkan dari nilai kernel data dengan *alpha* maksimal pada kelas positif di kalikan dengan kelas dari data tersebut dan nilai *alpha* dari data tersebut, sedangkan perhitungan bobot $w.x^-$ didapatkan dari nilai kernel data dengan *alpha* maksimal pada kelas negatif di kalikan dengan kelas dari data tersebut dan nilai *alpha* dari data tersebut. Selanjutnya total dari $w.x^+$ dan $w.x^-$ dijumlahkan dan dikali - 0.5 untuk mendapatkan nilai bias.

```
1     Public double hitungBias(double kelasLevel1[], double
2      alpha[], double kernelPoli[][]) {
3         totalBobotPositif = 0;
4         totalBobotNegatif = 0;
5         //cari alpha tertinggi dari kelas positif dan negatif
6         double tertinggiPositif = 0, tertinggiNegatif = 0;
7         int indeksPositif = 0, indeksNegatif = 0;
8         for (int i = 0; i < kelasLevel1.length; i++) {
9             if (kelasLevel1[i] > 0) {
10                if (alpha[i] > tertinggiPositif) {
11                    tertinggiPositif = alpha[i];
12                    indeksPositif = i;
13                }
14            } else {
15                if (alpha[i] > tertinggiNegatif) {
16                    tertinggiNegatif = alpha[i];
17                    indeksNegatif = i;
18                }
19            }
20        }
21        System.out.println("index positif = " + indeksPositif);
22        System.out.println("tertinggi positif = " +
23         tertinggiPositif);
24        System.out.println("index negatif = " + indeksNegatif);
25        System.out.println("tertinggi negatif = " +
26         tertinggiNegatif);
27
28        System.out.println("alpha pos-neg");
29        for (int i = 0; i < kelasLevel1.length; i++) {
30
31System.out.println(df.format(kernelPoli[indeksPositif][i]) +"\t"
32 + df.format(kernelPoli[indeksNegatif][i]) + "\t" + alpha[i] +
33 "\t" + kelasLevel1[i]);
34        }
35        //hitung bobot positif dan negatif
36        bobotPositif = new double[kelasLevel1.length];
37        bobotNegatif = new double[kelasLevel1.length];
38        for (int i = 0; i < kelasLevel1.length; i++) {
39
40            bobotPositif[i] = alpha[i] * kelasLevel1[i] *
41              kernelPoli[i][indeksPositif];
42            bobotNegatif[i] = alpha[i] * kelasLevel1[i] *
43              kernelPoli[i][indeksNegatif];
44            System.out.println(bobotPositif[i] + "\t" +
45              bobotNegatif[i]);
46            totalBobotPositif = totalBobotPositif +
47               bobotPositif[i];
48            totalBobotNegatif += bobotNegatif[i];
49
50
51        }
52        System.out.println("***");
53        System.out.println("totalBobotPositif" +
54         totalBobotPositif);
55        System.out.println("totalBobotNegatif" +
56         totalBobotNegatif);
```

```
57
58  //bias
59          double bias = -0.5 * (totalBobotPositif +
60           totalBobotNegatif);
61          System.out.println("bias " + bias);
62          return bias;
63      }
```

Kode Program 5. 6 Implementasi perhitungan bobot $w.x^+$ dan $w.x^-$ dan nilai bias

Penjelasan dari Kode Program 5.6 mengenai perhitungan bobot $w.x^+$ dan $w.x^-$ dan nilai bias sebagai berikut :

1. Baris ke 3 dan 4 digunakan untuk inisialisasi variable.
2. Baris ke 6-26 digunakan untuk mendapatkan nilai *alpha* tertinggi beserta indeksnya dari masing-masing kelas positif dan negatef kemudian mencetak hasil dari nilai tersebut.
3. Baris ke 40-51 digunakan untuk menghitung rumus dari pencarian bobot serta menjumlahkan total seluruh bobot positif dan negatif.
4. Baris ke 53-57 digunakan untuk mencetak nilai dari total bobot positif dan total bobot negatif.
5. Baris ke 59-63 digunakan untuk menghitung rumus dari nilai bias dan mencatak hasil dari nilai bias tersebut.

5.3.7 Implementasi Perhitungan nilai f(x)

Implementasi perhitungan nilai f(x) terdapat pada Kode Program 5.7. Perhitungan ini dilakukan setelah mendapatkan hasil dari perhitungan nilai bias. Sebelum mencari nilai f(x) terlebih dahulu menghitung nilai bobot dari seluruh data uji. Bobot dari seluruh data uji didapatkan dari nilai kernel setiap data latih terhadap data uji yang dikalikan dengan nilai *alpha* terakhir setiap data dan kelas pada data tersebut. Kemudian nilai f(x) didapatkan dari penjumlahan antara jumlah bobot masing-masing data uji dengan nilai bias.

```
1         public double[] hitungFx(double dataLatih[][], double
2          dataUji[][], double alpha[], double kelasLevel1[],
3           double bias) {
4          double[] fxfix = new double[dataUji.length];
5          double[][] kernelDataLatihUji = new
6           double[dataLatih.length][dataUji.length];
7          System.out.println("Kernel Latih-Uji");
8          for (int i = 0; i < kernelDataLatihUji.length; i++) {
9              for (int j = 0; j < kernelDataLatihUji[0].length;
10              j++) {
11                 double total = 0;
12                 for (int panjang = 0; panjang <
13                  dataLatih[0].length; panjang++) {
14                     total = total + dataLatih[i][panjang] *
15                      dataUji[j][panjang];
16                 }
17
```

```
18                  kernelDataLatihUji[i][j] = Math.pow(total + c,
19                   degree);
20
21  System.out.print(df.format(kernelDataLatihUji[i][j]) + "\t");
22              }
23              System.out.println("");
24          }
25
26          //cari bobot data uji = alpha*kelas*kernelDataLatihUji
27          double[][] bobotDataUji = new
28           double[dataLatih.length][dataUji.length];
29          System.out.println("Bobot Data Uji");
30          for (int i = 0; i < kernelDataLatihUji.length; i++) {
31              for (int j = 0; j < kernelDataLatihUji[0].length;
32               j++) {
33                  bobotDataUji[i][j] = alpha[i] * kelasLevel1[i] *
34                   kernelDataLatihUji[i][j];
35                  System.out.print(bobotDataUji[i][j] + "\t");
36              }
37              System.out.println("");
38          }
39          double[] totalBobotDataUji = new double[dataUji.length];
40          System.out.println("Total bobot");
41          for (int j = 0; j < dataUji.length; j++) {
42              double total = 0;
43              for (int i = 0; i < dataLatih.length; i++) {
44                  total = total + bobotDataUji[i][j];
45              }
46              totalBobotDataUji[j] = total;
47              System.out.println(totalBobotDataUji[j]);
48          }
49
50          //fx
51          double[] fx = new double[dataUji.length];
52          signfx = new double[dataUji.length];
53          for (int i = 0; i < dataUji.length; i++) {
54              fx[i] = bias + totalBobotDataUji[i];
55              signfx[i] = Math.signum(fx[i]);
56              //ditambah karena ingin memanggil nilai fx saat di
57                run di class svm
58              fxfix[i] = fx[i];
59          }
60          return fxfix;
        }
```

Kode Program 5. 7 Implementasi perhitungan nilai f(x)

Penjelasan dari Kode Program 5.7 mengenai implementasi perhitungan nilai f(x) sebagai berikut :

1. Baris ke 4-6 digunakan untuk inisialisasi variable.
2. Baris ke 8-22 digunakan untuk menghitung nilai dari perhitungan kernel setiap data latih terhadap data uji kemudian mencatak hasil dari perhitungan tersebut.
3. Baris ke 40-51 digunakan untuk menghitung rumus dari pencarian bobot serta menjumlahkan total seluruh bobot positif dan negatif.
4. Baris ke 53-57 digunakan untuk mencetak nilai dari total bobot positif dan total bobot negatif.

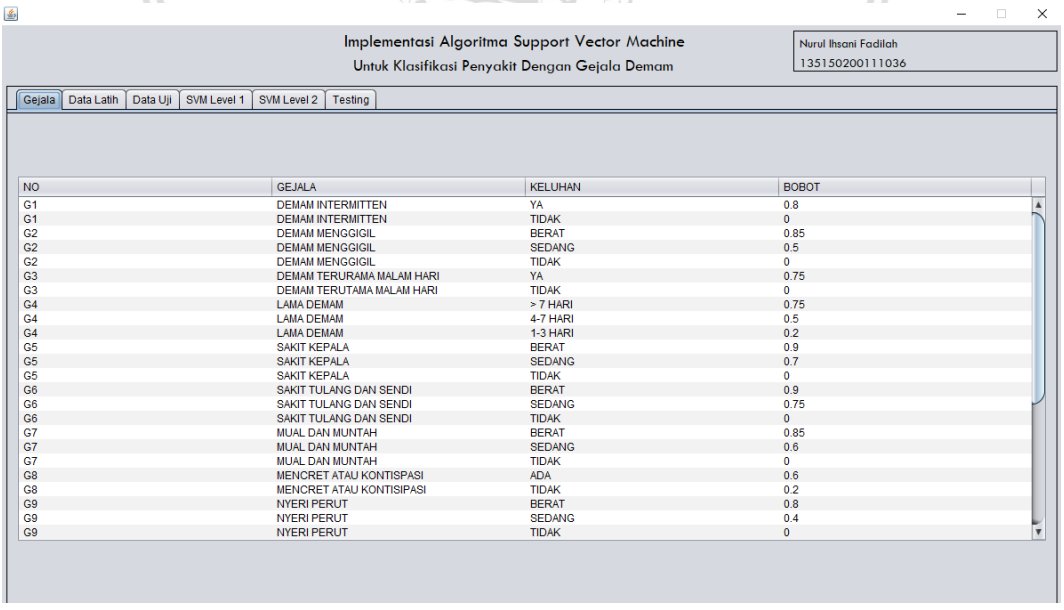
5. Baris ke 59-62 digunakan untuk menghitung rumus dari nilai bias dan mencatak hasil dari nilai bias tersebut.

5.4 Implementasi Antarmuka

Penjelasan implementasi antarmuka pada program klasifikasi penyakit dengan gejala demam menggunakan metode *Support Vector Machine* terdapat beberapa bagian yaitu implementasi antarmuka halaman gejela, antarmuka halaman data latih, antarmuka halaman SVM level 1, antarmuka halaman level 2, dan halaman *testing.*

5.4.1 Implementasi Antarmuka Halaman Gejala

Implementasi antarmuka halaman gejala berisi keterangan dari gejala yang digunakan sebagai parameter pada penelitian implementasi *Support Vector Machine* untuk mengklasifikasi penyakit dengan gejala demam. Terdapat tabel yang berisi nomor serta gejala yang dijadikan parameter. Gejala-gejala yang dimaksud adalah pembesaran hati, pembesaran limpa dan kulit lembab/keringat, demam intermitten, demam menggigil, demam terutama malam hari, demam lebih 1 minggu, sakit kepala, nyeri perut, bintik merah pada kulit, sakit tulang dan sendi, mual dan muntah, mencret dan susah BAB, lidah kotor, bradikardi relatif. Setiap gejala memiliki keterangan bobot berdasarkan keluhan yang dirasakan oleh pasien. Nilai dari bobot yang di ambil berdasarkan keluhan di setiap gejala inilah yang digunakan dalam perhitungan klasifikasi algoritme *Support Vector Machine.* Implementasi antarmuka halaman gejala ditunjukkan pada Gambar 5.1.

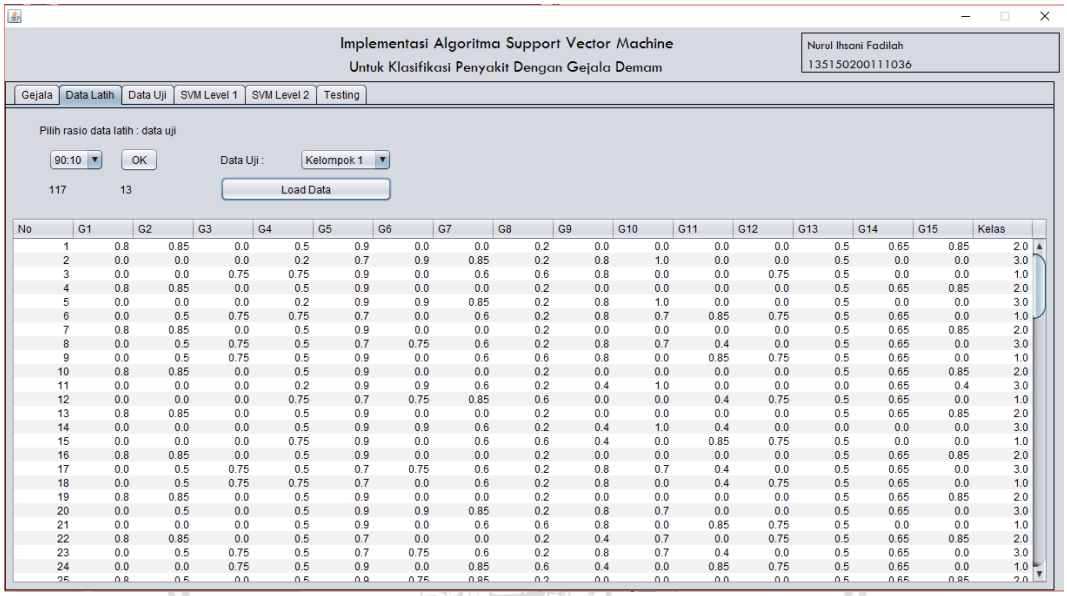


NO	GEJALA	KELUHAN	BOBOT
G1	DEMAM INTERMITTEN	YA	0.8
G1	DEMAM INTERMITTEN	TIDAK	0
G2	DEMAM MENGGIGIL	BERAT	0.85
G2	DEMAM MENGGIGIL	SEDANG	0.5
G2	DEMAM MENGGIGIL	TIDAK	0
G3	DEMAM TERURAMA MALAM HARI	YA	0.75
G3	DEMAM TERUTAMA MALAM HARI	TIDAK	0
G4	LAMA DEMAM	> 7 HARI	0.75
G4	LAMA DEMAM	4-7 HARI	0.5
G4	LAMA DEMAM	1-3 HARI	0.2
G5	SAKIT KEPALA	BERAT	0.9
G5	SAKIT KEPALA	SEDANG	0.7
G5	SAKIT KEPALA	TIDAK	0
G6	SAKIT TULANG DAN SENDI	BERAT	0.9
G6	SAKIT TULANG DAN SENDI	SEDANG	0.75
G6	SAKIT TULANG DAN SENDI	TIDAK	0
G7	MUAL DAN MUNTAH	BERAT	0.85
G7	MUAL DAN MUNTAH	SEDANG	0.6
G7	MUAL DAN MUNTAH	TIDAK	0
G8	MENCRET ATAU KONTISPASI	ADA	0.6
G8	MENCRET ATAU KONTISIPASI	TIDAK	0.2
G9	NYERI PERUT	BERAT	0.8
G9	NYERI PERUT	SEDANG	0.4
G9	NYERI PERUT	TIDAK	0

Gambar 5. 1 Implementasi Halaman Gejala

5.4.2 Implementasi Antarmuka Halaman Data Latih

Implementasi antarmuka halaman data latih berfungsi untuk menampilkan data latih yang digunakan pada penelitian implementasi *Support Vector Machine* untuk mengklasifikasi penyakit dengan gejala demam. Pada halaman ini, terdapat pilihan untuk memilih pembagian rasio data latih dan data uji yang akan digunakan, dan terdapat pula pilihan kelompok data yang digunakan menggunakan *K-Fold Cross Validation*. Untuk menampilkan data yang telah dipilih, terdapat tombol Load Data yang harus di pilih terlebih dahulu untuk menampilkan seluruh data latih dan data uji. Implementasi antarmuka halaman gejala ditunjukkan pada Gambar 5.2.

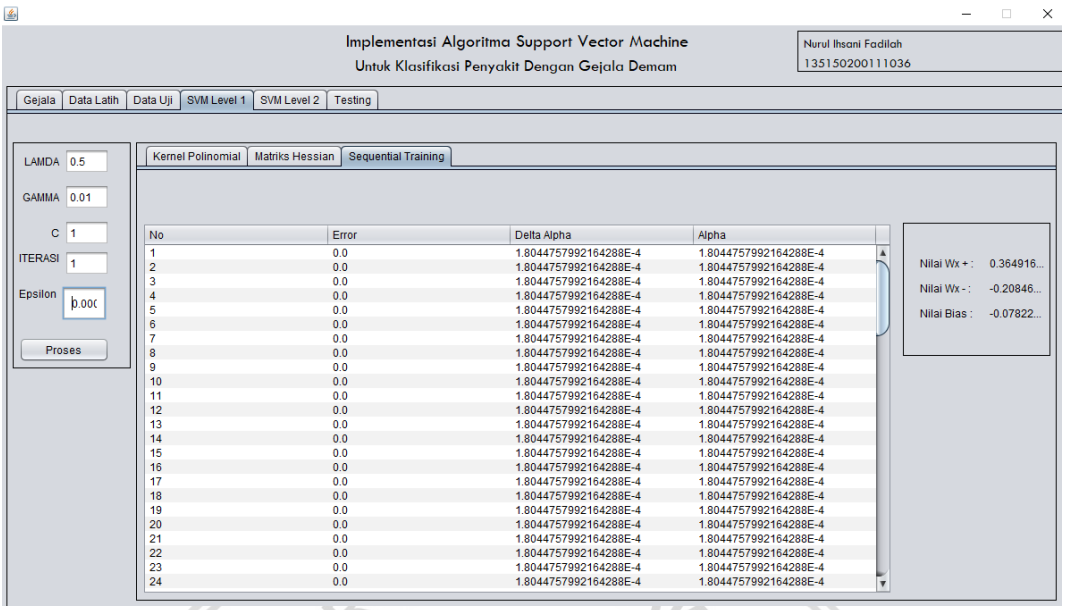


Gambar 5. 2 Implementasi halaman data latih

5.4.3 Implementasi Antarmuka Halaman SVM Level 1

Implementasi antarmuka halaman SVM level 1 berisi hasil dari setiap bagian perhitungan yang dilakukan program implementasi *Support Vector Machine* untuk klasifikasi penyakit dengan gejala demam pada level 1. Pada halaman ini, terdapat *text field* untuk memberikan masukan terhadap parameter yang harus diinputkan sebelum memulai proses perhitungan. Parameter yang harus diinputkan adalah nilai *lamda*, *gamma*, C, iterasi maksimal dan *epsilon*. Terdapat tombol proses yang berfungsi untuk memulai perhitungan dari program ini. Halaman SVM level 1 memilik 3 sub menu yaitu kernel *polynomial*, matriks Hessian dan *sequential training*. Sub menu kernel *polynomial* berisi tabel dari hasil perhitungan kernel *polynomial* pada level 1, sub menu matriks Hessian berisi tabel dari hasil perhitungan matriks Hessian pada level 1, sub menu *sequential training* berisi hasil nilai error, nilai *delta alpha* dan nilai *alpha* terakhir. Pada sub menu sequential training ditampilkan juga hasil dari nilai wx^+, wx^- dan nilai bias dari perhitungan

SVM level 1 . Implementasi antarmuka halaman SVM level 1 ditunjukkan pada Gambar 5.3.



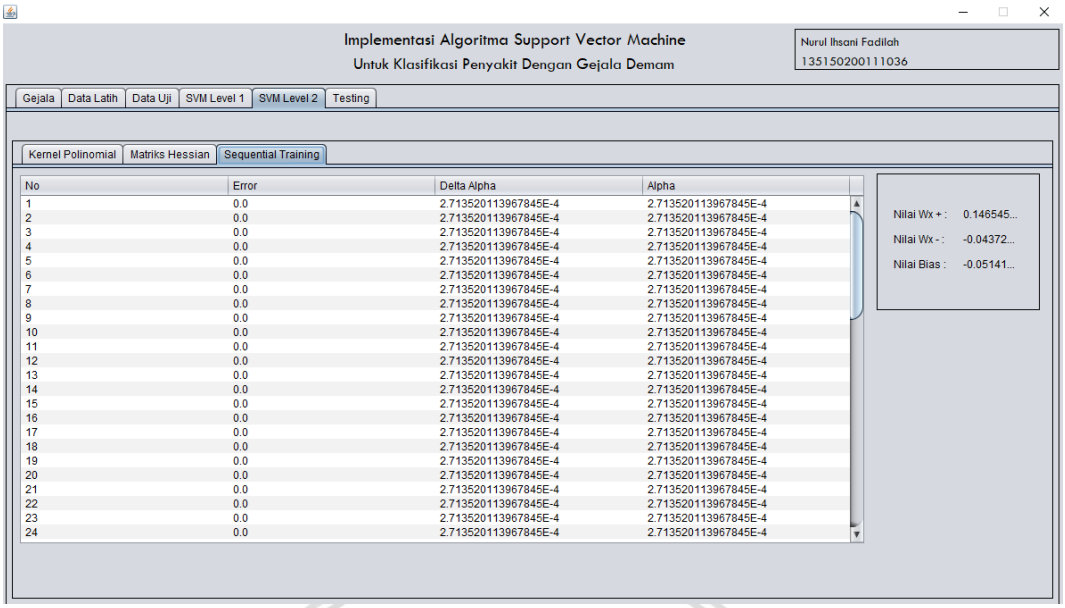
Gambar 5. 3 Implementasi Halaman SVM Level 1

5.4.4 Implementasi Antarmuka Halaman SVM Level 2

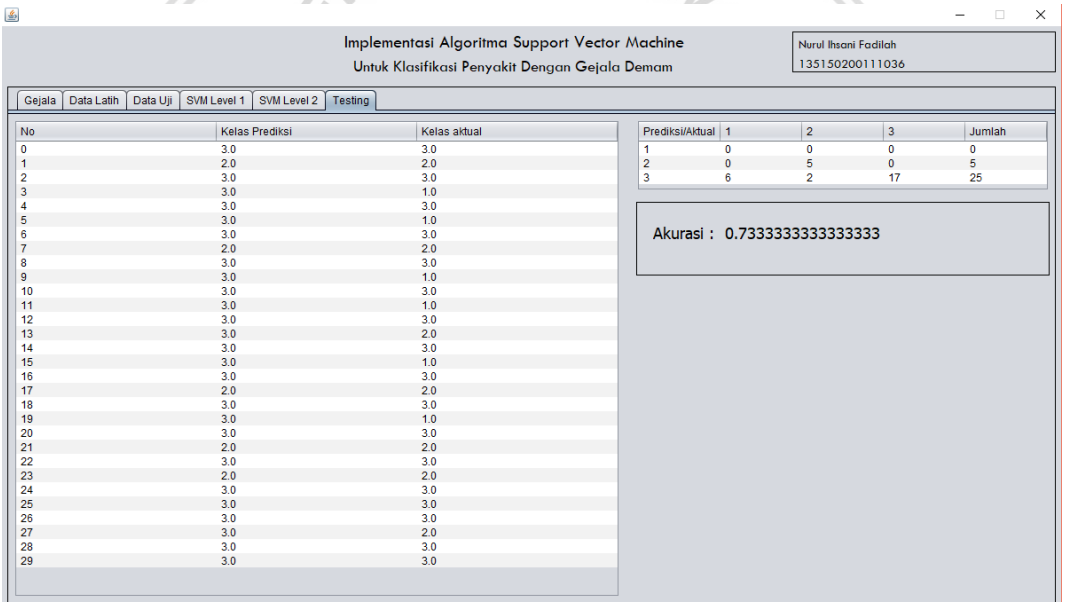
Implementasi antarmuka halaman SVM level 2 berisi hasil dari setiap bagian perhitungan yang dilakukan program implementasi *Support Vector Machine* untuk klasifikasi penyakit dengan gejala demam pada level 2. Halaman SVM level 2 memiliki 3 sub menu yaitu kernel *polynomial*, matriks Hessian dan *sequential training*. Sub menu kernel *polynomial* berisi tabel dari hasil perhitungan kernel *polynomial* pada level 2, sub menu matriks Hessian berisi tabel dari hasil perhitungan matriks Hessian pada level 2, sub menu *sequential training* berisi hasil nilai error, nilai *delta alpha* dan nilai *alpha* terakhir. Pada sub menu sequential *training* ditampilkan juga hasil dari nilai wx^+, wx^- dan nilai bias dari perhitungan SVM level 2. Implementasi antarmuka halaman SVM level 2 ditunjukkan pada Gambar 5.4.

5.4.5 Implementasi Antarmuka Halaman *Testing*

Implementasi antarmuka halaman *testing* berisi hasil prediksi akhir dari perhitungan seluruh proses baik pada proses perhitungan SVM di level 1 maupun pada proses perhitungan SVM di level 2. Pada halaman ini menunjukkan kelas prediksi yang dihasilkan oleh sistem dan dibandingkan dengan kelas aktual atau kelas asli pada data uji yang digunakan. Pada bagian kanan halaman terdapat nilai akumulasi dari akurasi yang didapatkan dari klasifikasi yang berhasil di prediksi dengan baik. Implementasi dari halaman testing ditunjukkan pada Gambar 5.5.



Gambar 5. 4 Implementasi Halaman SVM Level 2



Gambar 5. 5 Implementasi Halaman *Testing*

BAB 6 PENGUJIAN DAN ANALISIS

Pada bab ini berisi penjabaran tentang bagaimana hasil dari pegujian yang dilakukan pada implementasi *Support Vector Machine* untuk klasifikasi penyakit dengan gejala demam. Selain itu dijabarkan juga bagaimana analisis dari hasil pengujian yang telah dilakukan.

6.1 Sistematika Pengujian

Pengujian pada implementasi Support Vector Machine untuk klasifikasi penyakit dengan gejala demam dilakukan sesuai skenario yang telah dirancang pada bab sebelumnya. Pengujian yang dilakukan sesuai dengan skenario yang telah dirancang adalah pengujian pengaruh perbandingan jumlah data *training* dan data *testing* terhadap nilai akurasi, pengujian pengaruh nilai *Lamda* (λ) terhadap nilai akurasi, pengujian pengaruh nilai *Gamma* (γ) terhadap nilai akurasi, pengujian pengaruh nilai C (Complexity) terhadap akurasi, pengujian pengaruh nilai Iterasi maksimum terhadap nilai akurasi, pengujian pengaruh nilai *epsilon* (ε) terhadap nilai akurasi. Pengujian ini menggunakan *K-Fold Cross Validation*, jadi setiap pengujian yang dilakukan, jumlah percobaan disesuaikan dengan K. Selanjutnya hasil akurasi seluruh percobaan disetiap pengujian di analisis dengan menggunakan hasil dari rata-rata akurasinya.

6.2 Hasil dan Analisis Pembahasan

Hasil dan analisis pembahasan pad sub bab ini akan menguraikan bagaimana hasil dari pengujian implementasi Support Vector Machine untuk klasifikasi penyakit dengan gejala demam. Pembahasanan ini akan menjelaskan bagaimana skenario yang telah dirancang, serta menjelaskan hasil dan analisis dari pengujian yang sesuai dengan skenario perancangan pengujian.

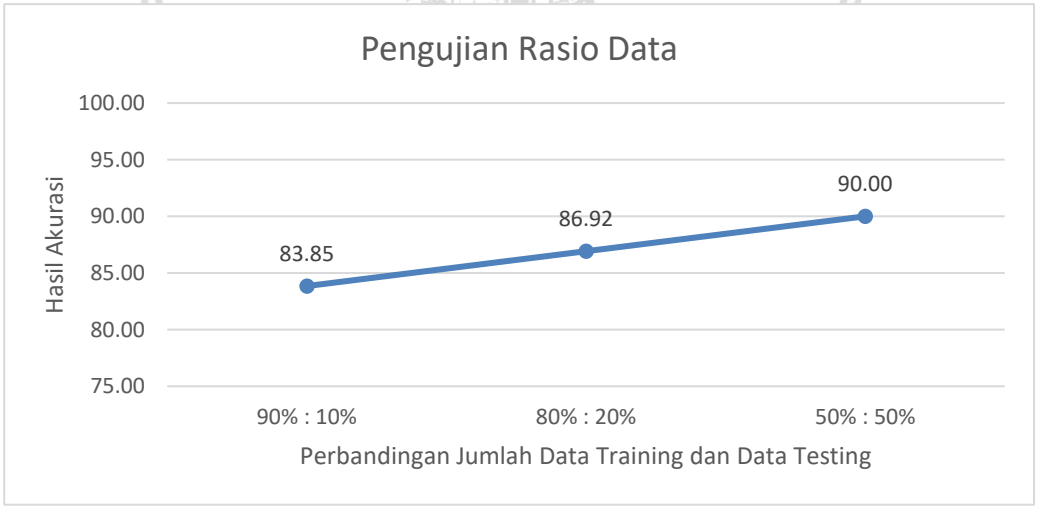
6.2.1 Pengujian Pengaruh Perbandingan Jumlah Data *Training* dan Data *Testing* Terhadap Nilai Akurasi

Pengujian pengaruh perbandingan jumlah data *training* dan data *testing* terhadap nilai akurasi dilakukan dengan membagi dataset sebanyak 130 data menjadi data *training* dan data *testing*. Pembagian data *training* dan data *testing* sesuai dengan rasio perbandingan yang telah ditetapkan pada perancangan pengujian yaitu 90%:10%, 80%:20%, dan 50%:50%. Pengujian ini menggunakan *K-Fold Cross Validation* dimana setiap pembagian rasio data latih dan data uji yang berbeda menggunakan k yang berbeda pula. Pembagian data latih data uji yang digunakan adalah 90%:10% dengan k=10, 80%:20% dengan k=5, dan 50%:50% dengan k=2.. Tabel 6.1 menunjukkan hasil dari pengujian yang dilakukan. Pengujian dilakukan dengan menggunakan parameter λ=0.5, γ=0.01, C=1, iterasi maksimum=2, ε=0.0001.

Tabel 6. 1 Hasil pengujian pengaruh perbandingan jumlah data *training* dan data *testing* terhadap nilai akurasi

Data Latih : Data Uji	Dataset K-*Fold* ke- (%)										Rata-Rata Akurasi
90%:10% (k=10)	1	2	3	4	5	6	7	8	9	10	
	92.31	100.00	76.92	92.31	76.92	69.23	92.31	100.00	76.92	61.54	83.85
80%:20% (k=5)	1		2		3		4		5		
	96.15		96.15		76.92		96.15		69.23		86.92
50%:50% (k=2)	1					2					
	98.46					81.54					90.00

Hasil pengujian menunjukkan bahwa perbedaan pada perbandingan jumlah data-latih dan data uji memiliki pengaruh pada nilai akurasi. Pada perbandingan dengan rasio 90%:10%, didapatkan rata-rata akurasi sebesar 83.85%. Pada perbandingan dengan rasio 80%:20%, didapatkan rata-rata akurasi sebesar 86.92%. Pada perbandingan dengan rasio 50%:50%, didapatkan rata-rata akurasi sebesar 90.00%. Pengujian rasio data ini menunjukkan semakin besar k yang digunakan maka semakin banyak data latih yang digunakan. Selain itu perbedaan komposisi kelas pada jumlah data latih dan data uji mempengaruhi hasil akurasi. Hasil dari rata-rata akurasi pengujian perbandingan rasio dapat dilihat dalam bentuk grafik pada Gambar 6.1.



Gambar 6. 1 Grafik hasil pengujian pengaruh perbandingan jumlah data *training* dan data *testing* terhadap nilai akurasi

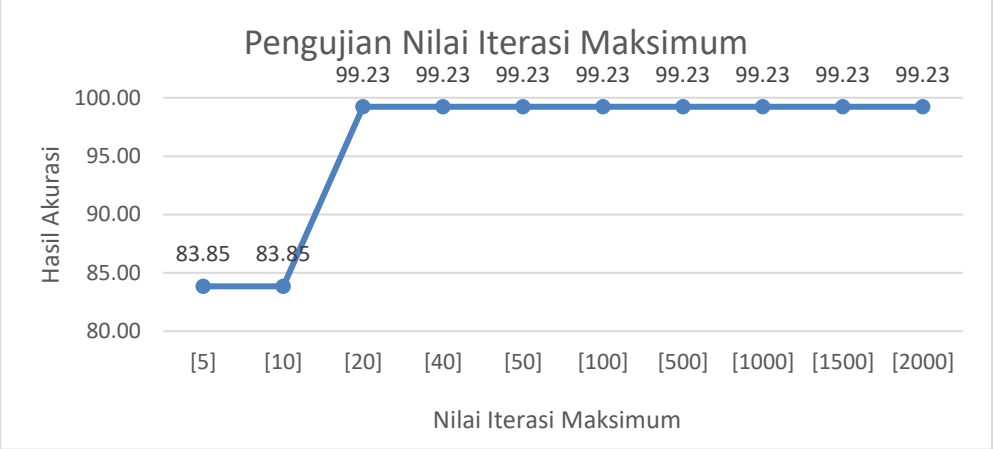
6.2.2 Pengujian Pengaruh Nilai Iterasi Maksimum Terhadap Nilai Akurasi

Pengujian pengaruh nilai iterasi maksimum terhadap nilai akurasi dilakukan untuk mengetahui adakah pengaruh dari perubahan nilai iterasi maksimum pada percobaan yang telah dilakukan, dalam perubahan nilai akurasi. Pengujian ini menggunakan *K-Fold Cross Validation* dimana pembagian rasio data latih dan data uji yang digunakan adalah 90%:10% dengan k=10. Tabel 6.2 menunjukkan hasil dari pengujian yang dilakukan. Pengujian dilakukan dengan menggunakan nilai parameter λ=0.5, γ=0.01, C=1, ε=0.0001. Nilai iterasi maksimum yang akan diujikan yaitu 5, 10, 20, 40, 50, 100, 500, 1000, 1500, 2000.

Tabel 6. 2 Hasil pengujian pengaruh nilai Iterasi maksimum terhadap nilai akurasi

Nilai Iterasi Maksimum	Dataset k-*fold* ke-										Rata-Rata Akurasi
	1	2	3	4	5	6	7	8	9	10	
5	92.31	100.00	76.92	92.31	76.92	69.23	92.31	100.00	76.92	61.54	83.85
10	92.31	100.00	76.92	92.31	76.92	69.23	92.31	100.00	76.92	61.54	83.85
20	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	92.31	99.23
40	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	92.31	99.23
50	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	92.31	99.23
100	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	92.31	99.23
500	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	92.31	99.23
1000	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	92.31	99.23
1500	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	92.31	99.23
2000	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	92.31	99.23

Hasil pengujian menunjukkan bahwa perbedaan nilai pada iterasi maksimum memiliki pengaruh pada nilai akurasi. Rata-rata akurasi tertinggi didapatkan pada pengujian dengan nilai iterasi maksimum = 20 sampai 2000 dengan akurasi sebesar 99.23%. Hasil dari rata-rata akurasi pengujian pengaruh nilai iterasi maksimum terhadap nilai akurasi dapat dilihat dalam bentuk grafik pada Gambar 6.2. Grafik tersebut menunjukkan bahwa akurasi bernilai rendah ketika nilai iterasi maksimum berada pada angka 5 dan 10 sedangkan akurasi meningkat dan menunjukkan nilai akurasi menjadi konvergen dimulai dari iterasi maksimum bernilai 20. Hal ini disebabkan karena semakin besar iterasi membuat proses pembelajaran semakin baik, sehingga hasil pengujian yang didapat juga semakin baik.



Gambar 6. 2 Grafik hasil pengujian pengaruh nilai Iterasi maksimum terhadap nilai akurasi

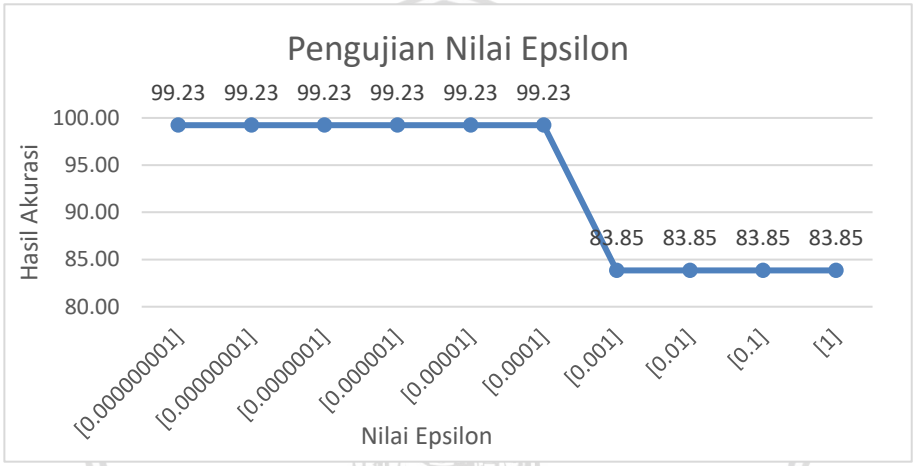
6.2.3 Pengujian Pengaruh Nilai *Epsilon* (ε) Terhadap Nilai Akurasi

Pengujian pengaruh nilai *epsilon* terhadap nilai akurasi dilakukan untuk mengetahui adakah pengaruh dari perubahan nilai *epsilon* pada percobaan yang telah dilakukan, dalam perubahan nilai akurasi. Pengujian ini menggunakan *K-Fold Cross Validation* dimana pembagian rasio data latih dan data uji yang digunakan adalah 90%:10% dengan k=10. Tabel 6.3 menunjukkan hasil dari pengujian yang dilakukan. Pengujian dilakukan dengan menggunakan nilai parameter terbaik yang sebelumnya telah diujikan. Sehingga nilai parameter yang digunakan untuk iterasi maksimum=20, kemudian parameter lain yang digunakan adalah λ=0.5, γ=0.01, C=1. Nilai *epsilon*(ε) yang telah dipilih yaitu 0.000000001, 0.00000001, 0.0000001, 0.000001, 0.00001, 0.0001, 0.001, 0.01, 0.1, 1.

Tabel 6. 3 Hasil pengujian pengaruh nilai *epsilon* (ε) terhadap nilai akurasi

Nilai *Epsilon*	Dataset k-*fold* ke-										Rata-Rata Akurasi
	1	2	3	4	5	6	7	8	9	10	
0.000000001	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	92.31	99.23
0.00000001	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	92.31	99.23
0.0000001	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	92.31	99.23
0.000001	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	92.31	99.23
0.00001	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	92.31	99.23
0.0001	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	92.31	99.23
0.001	92.31	100.00	76.92	92.31	76.92	69.23	92.31	100.00	76.92	61.54	83.85
0.01	92.31	100.00	76.92	92.31	76.92	69.23	92.31	100.00	76.92	61.54	83.85
0.1	92.31	100.00	76.92	92.31	76.92	69.23	92.31	100.00	76.92	61.54	83.85
1	92.31	100.00	76.92	92.31	76.92	69.23	92.31	100.00	76.92	61.54	83.85

Hasil pengujian menunjukkan bahwa perbedaan nilai pada *epsilon* memiliki pengaruh pada nilai akurasi. Rata-rata akurasi tertinggi didapatkan pada pengujian dengan nilai *epsilon* = 0.000000001 sampai dengan 0,0001 dengan akurasi sebesar 99.23%. Hasil dari rata-rata akurasi pengujian pengaruh nilai *epsilon* terhadap nilai akurasi dapat dilihat dalam bentuk grafik pada Gambar 6.3. Grafik tersebut menunjukkan bahwa rata-rata akurasi menurun menjadi 83.85% ketika nilai *epsilon* berada pada angka 0.001 sampai dengan 1. Hal ini disebabkan karena semakin besar nilai *epsilon* maka semakin cepat memenuhi kondisi syarat berhentinya iterasi maksimum, yaitu nilai maksimum *delta alpha* kurang dari nilai *epsilon*. Jadi semakin besar nilai *epsilon* semakin sedikit iterasi yang dilakukan dan semakin rendah pula hasil dari proses *training* yang dilakukan. Sebaliknya, semakin kecil nilai *epsilon* maka semakin banyak iterasi yang dapat dilakukan dan semakin tinggi nilai akurasi yang didapatkan.



Gambar 6. 3 Grafik hasil pengujian pengaruh nilai epsilon (ε) terhadap nilai akurasi

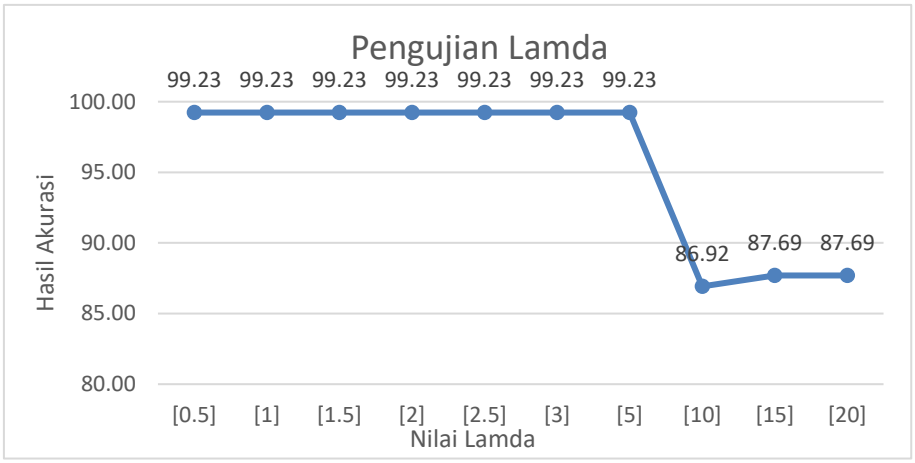
6.2.4 Pengujian Pengaruh Nilai *Lamda* (λ) Terhadap Nilai Akurasi

Pengujian pengaruh nilai *lamda* (λ) terhadap nilai akurasi dilakukan untuk mengetahui adakah pengaruh dari perubahan nilai *lamda* (λ) pada percobaan yang telah dilakukan, dalam perubahan nilai akurasi. Pengujian ini menggunakan *K-Fold Cross Validation* dimana pembagian rasio data latih dan data uji yang digunakan adalah 90%:10% dengan k=10. Tabel 6.4 menunjukkan hasil dari pengujian yang dilakukan. Pengujian dilakukan dengan menggunakan nilai parameter terbaik yang sebelumnya telah diujikan. Sehingga nilai parameter yang digunakan untuk iterasi maksimum=20, dan ε=0.0001, kemudian parameter lain yang digunakan adalah γ=0.01, C=1. Nilai *lamda* yang akan diujikan yaitu 0.5, 1, 1.5, 2, 2.5, 3, 5, 10, 15, dan 20.

Tabel 6. 4 Hasil pengujian pengaruh nilai *Lamda* (λ) terhadap nilai akurasi

Nilai *Lamda* (λ)	Dataset k-*fold* ke-										Rata-Rata Akurasi
	1	2	3	4	5	6	7	8	9	10	
0.5	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	92.31	99.23
1	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	92.31	99.23
1.5	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	92.31	99.23
2	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	92.31	99.23
2.5	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	92.31	99.23
3	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	92.31	99.23
5	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	92.31	99.23
10	92.31	100.00	76.92	100.00	84.62	76.92	92.31	92.31	92.31	61.54	86.92
15	92.31	100.00	76.92	100.00	84.62	76.92	92.31	92.31	100.00	61.54	87.69
20	92.31	100.00	76.92	100.00	84.62	76.92	92.31	92.31	100.00	61.54	87.69

Hasil pengujian menunjukkan bahwa perbedaan nilai pada *lamda* memiliki pengaruh pada nilai akurasi. Rata-rata akurasi tertinggi didapatkan pada pengujian dengan nilai *lamda* = 0.5 sampai dengan 5 dengan akurasi sebesar 99.23%. Hasil dari rata-rata akurasi pengujian pengaruh nilai *lamda* terhadap nilai akurasi dapat dilihat dalam bentuk grafik pada Gambar 6.4. Grafik tersebut menunjukkan semakin tinggi nilai *lamda* yang diberikan maka nilai akurasi yang didapatkan akan semakin rendah. Terbukti ketika nilai *lamda* bernilai 10, 15 dan 20, rata-rata akurasi menurun menjadi 86.92% dan 87.69%. Nilai *lamda* digunakan dalam perhitungan matriks Hessian, sehingga semakin besar nilai *lamda* yang diberikan akan berpengaruh terhadap kecepatan komputasi program dalam menyelesaikan perhitungan, terutama perhitungan matriks Hessian. Diagonal dari matriks Hessian juga mempengaruhi nilai *gamma* yang digunakan pada perhitungan maksimum *delta alpha*. Semakin besar nilai *lamda* maka semakin kecil nilai dari *delta alpha* yang didapat. Semakin kecil *delta alpha* maka akan semakin besar kemungkinan bertemu dengan kondisi syarat iterasi berhenti yaitu maksimum *delta alpha* lebih kecil dari nilai *epsilon*.



Gambar 6. 4 Grafik hasil pengujian pengaruh nilai Lamda (λ) terhadap nilai akurasi

6.2.5 Pengujian Pengaruh Nilai *Gamma* (*γ*) Terhadap Nilai Akurasi

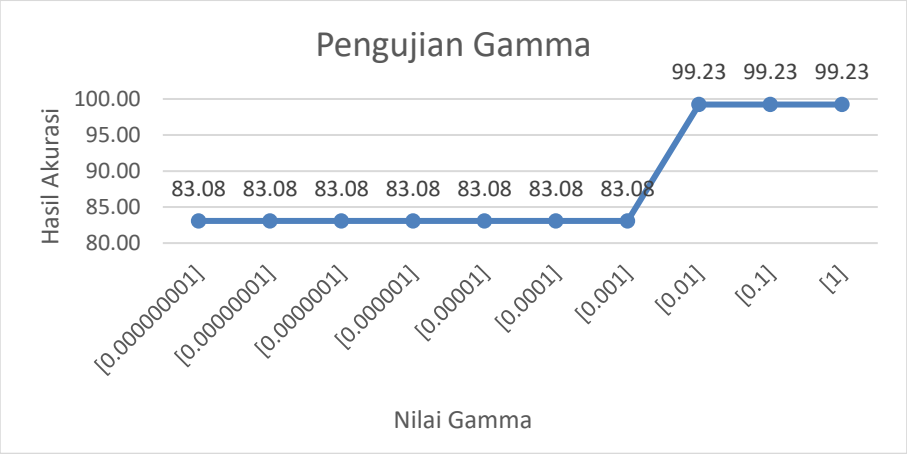
Pengujian pengaruh nilai *gamma* (γ) terhadap nilai akurasi dilakukan untuk mengetahui adakah pengaruh dari perubahan nilai *gamma* (γ) pada percobaan yang telah dilakukan, dalam perubahan nilai akurasi. Pengujian ini menggunakan *K-Fold Cross Validation* dimana pembagian rasio data latih dan data uji yang digunakan adalah 90%:10% dengan k=10. Tabel 6.5 menunjukkan hasil dari pengujian yang dilakukan. Pengujian dilakukan dengan menggunakan nilai parameter terbaik yang sebelumnya telah diujikan. Sehingga nilai parameter yang digunakan untuk iterasi maksimum=20, ε=0.0001, dan γ=0.5, kemudian parameter lain yang digunakan adalah C=1. Nilai *gamma* yang akan diujikan yaitu 0.000000001, 0.00000001, 0.0000001, 0.000001, 0.00001, 0.0001, 0.001, 0.01, 0.1, 1.

Tabel 6. 5 Hasil pengujian pengaruh nilai *Gamma* (*γ*) terhadap nilai akurasi

Nilai *Gamma* (*γ*)	Dataset k-*fold* ke-										Rata-Rata Akurasi
	1	2	3	4	5	6	7	8	9	10	
0.000000001	92.31	100.00	76.92	92.31	76.92	69.23	84.62	100.00	76.92	61.54	83.08
0.00000001	92.31	100.00	76.92	92.31	76.92	69.23	84.62	100.00	76.92	61.54	83.08
0.0000001	92.31	100.00	76.92	92.31	76.92	69.23	84.62	100.00	76.92	61.54	83.08
0.000001	92.31	100.00	76.92	92.31	76.92	69.23	84.62	100.00	76.92	61.54	83.08
0.00001	92.31	100.00	76.92	92.31	76.92	69.23	84.62	100.00	76.92	61.54	83.08
0.0001	92.31	100.00	76.92	92.31	76.92	69.23	84.62	100.00	76.92	61.54	83.08
0.001	92.31	100.00	76.92	92.31	76.92	69.23	84.62	100.00	76.92	61.54	83.08
0.01	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	92.31	99.23
0.1	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	92.31	99.23
1	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	92.31	99.23

Hasil pengujian menunjukkan bahwa perbedaan nilai pada *gamma* memiliki pengaruh pada nilai akurasi. Rata-rata akurasi tertinggi didapatkan pada pengujian

dengan nilai *lamda* = 0.01 sampai dengan 1 dengan akurasi sebesar 99.23%. Hasil dari rata-rata akurasi pengujian pengaruh nilai *gamma* terhadap nilai akurasi dapat dilihat dalam bentuk grafik pada Gambar 6.5. Grafik tersebut menunjukkan semakin tinggi nilai *gamma* yang diberikan maka nilai akurasi yang didapatkan akan semakin tinggi pula. Sedangkan sebaliknya, semakin rendah nilai *gamma* akan membuat akurasi juga semakin rendah. Hal ini disebabkan karena nilai *gamma* memiliki pengaruh pada pencarian nilai *delta alpha*. Semakin tinggi *gamma* maka semakin menyebabkan nilai *delta alpha* menjadi tinggi sehingga maksimum *delta alpha* bisa lebih banyak melakukan iterasi sampai batas maksimum *delta alpha* lebih besar dari *epsilon*.



Gambar 6.5 Grafik hasil pengujian pengaruh nilai *Gamma* (*γ*) terhadap nilai akurasi

6.2.6 Pengujian Pengaruh Nilai C (*Complexity*) Terhadap Akurasi

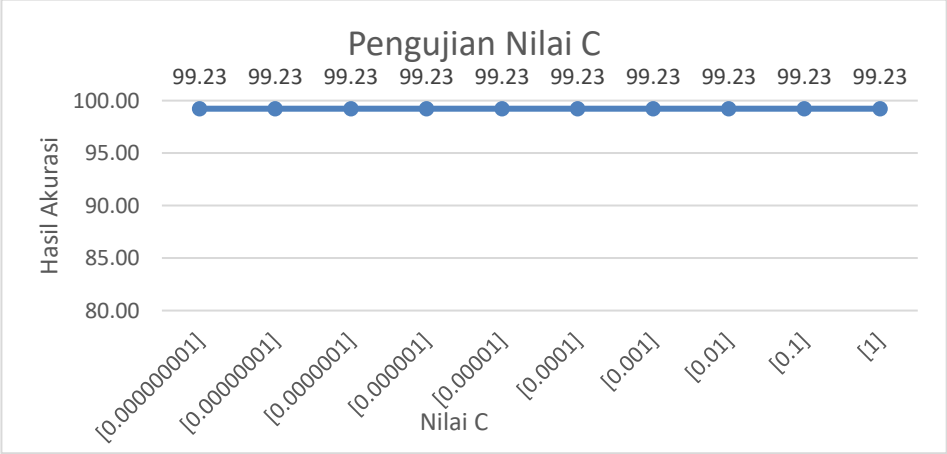
Pengujian pengaruh nilai C (*complexity*) terhadap nilai akurasi dilakukan untuk mengetahui adakah pengaruh dari perubahan nilai C (*complexity*) pada percobaan yang telah dilakukan, dalam perubahan nilai akurasi. Pengujian ini menggunakan *K-Fold Cross Validation* dimana pembagian rasio data latih dan data uji yang digunakan adalah 90%:10% dengan k=10. Tabel 6.6 menunjukkan hasil dari pengujian yang dilakukan. Pengujian dilakukan dengan menggunakan nilai parameter terbaik yang sebelumnya telah diujikan. Sehingga nilai parameter yang digunakan untuk λ=0.5 dan γ=0.01, maksimum=20, dan ε=0.0001. Nilai C (*complexity*) yang akan diujikan yaitu 0.000000001, 0.00000001, 0.0000001, 0.000001, 0.00001, 0.0001, 0.001, 0.01, 0.1, 1.

Tabel 6. 6 Hasil pengujian pengaruh nilai C (Complexity) terhadap akurasi

Nilai C (*Complexity*)	Dataset k-*fold* ke-										Rata-Rata Akurasi
	1	2	3	4	5	6	7	8	9	10	
0.000000001	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	92.31	99.23
0.00000001	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	92.31	99.23
0.0000001	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	92.31	99.23

0.000001	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	92.31	99.23
0.00001	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	92.31	99.23
0.0001	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	92.31	99.23
0.001	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	92.31	99.23
0.01	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	92.31	99.23
0.1	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	92.31	99.23
1	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	92.31	99.23

Hasil pengujian menunjukkan bahwa perbedaan nilai pada C (*complexity*) memiliki pengaruh yang sama pada nilai akurasi, karena rata-rata akurasi yang didapatkan pada seluruh nilai C (*complexity*) yang digunakan pada pengujian (dari 0.000000001 sampai 1) bernilai sama yaitu sebesar 99.23%. Hasil dari rata-rata akurasi pengujian pengaruh nilai C (*complexity*) terhadap nilai akurasi dapat dilihat dalam bentuk grafik pada Gambar 6.6. Grafik tersebut menunjukkan bahwa dengan nilai C yang berbeda, akurasi yang didapatkan sama dan cukup tinggi, hal ini disebabkan karena adanya pengaruh dari nilai parameter lain, yaitu dari jumlah iterasi, nilai *epsilon*, *lamda* dan *gamma* yang digunakan pada pengujian, yang merupakan parameter terbaik pada pengujian sebelumnya. Walaupun seharusnya semakin kecil nilai C (*complexity*) maka semakin tidak toleran terhadap kesalahan sehingga akurasi yang didapatkan semakin baik.



Gambar 6. 6 Grafik hasil pengujian pengaruh nilai C (Complexity) terhadap akurasi

6.2.7 Pengujian *K-fold Cross Validation*

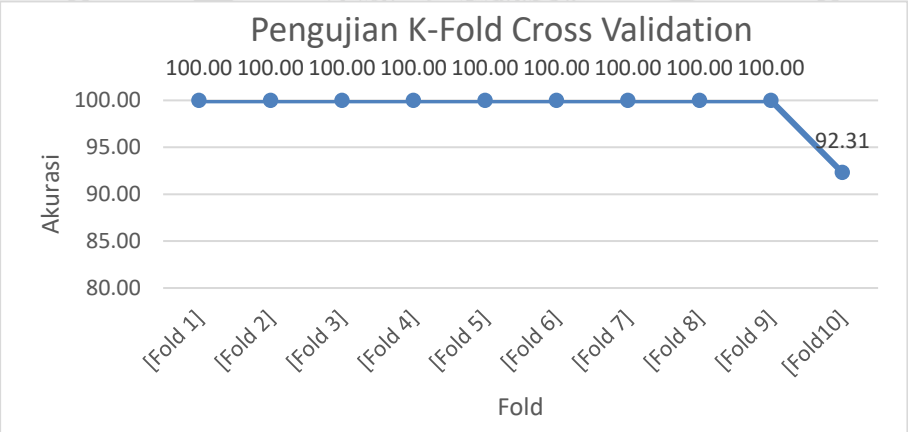
Pengujian ini dilakukan untuk mengetahui bagaimana perbandingan akurasi pada setiap *fold* dari 10-*fold cross validation* sehingga dapat dilihat bagaimana kestabilan akurasi yang didapatkan. Pengujian *K-fold Cross Validation* dilakukan dengan menggunakan pembagian rasio data latih dan data uji yaitu 90%:10%, dengan dataset sejumlah 130 sehingga dalam setiap *fold* terdapat 13 dataset yang dibagi secara acak dengan terdapat pembagian kelas yang berbeda . Parameter yang digunakan adalah parameter yang terbaik pada setiap pengujiannya, yaitu

lamda = 0.5, *gamma* = 0.01, C (*complexity*) = 1, *epsilon* = 0.0001, iterasi maksimum = 20. Tabel 6.7 menunjukkan hasil dari pengujian yang dilakukan.

Tabel 6. 7 Hasil pengujian *k-fold coss validation*

Dataset k-*fold* ke-										Rata-Rata Akurasi
1	2	3	4	5	6	7	8	9	10	
100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	92.31	99.23

Hasil pengujian *k-fold cross validation* dengan k=10 menunjukkan bahwa setiap *fold* mendapatkan nilai akurasi terbaik karena parameter yang digunakan adalah parameter yang terbaik pada setiap pengujiannya. Hasil dari rata-rata akurasi pengujian *k-fold cross validation* dapat dilihat dalam bentuk grafik pada Gambar 6.7. Pada *fold* 1 sampai dengan *fold* 9 akurasi yang didapatkan 100% sedangkan *fold* 10 mendapatkan akurasi yang lebih rendah dari yang lainnya. Hal ini disebabkan karena pada data uji yang terdapat pada *fold* 10 hanya mewakili 2 kelas dari 3 kelas yang ada sehingga pada proses *testing*, ditemukan data yang tidak dapat terklasifikasikan dengan baik. Sedangkan data uji pada *fold* 1 sampai 9 terdapat perwakilan dataset dari seluruh kelas yaitu kelas demam berdarah, kelas malaria dan kelas tifoid. Sehingga hasil dari proses *testing* yang dilakukan, seluruh data dapat terklasifikasi dengan baik.



Gambar 6. 7 Grafik hasil pengujian *k-fold cross validation*

BAB 7 PENUTUP

Pada bab ini berisi penjabaran tentang kesimpulan dari hasil penelitian terkait implementasi algoritme *Support Vector Machine* (SVM) untuk klasifikasi penyakit dengan gejala demam disertai saran untuk pengembangan penelitian lebih lanjut.

7.1 Kesimpulan

Berdasarkan dari hasil penelitian tentang implementasi algoritme *Support Vector Machine* (SVM) untuk klasifikasi penyakit dengan gejala demam, dapat disimpulkan bahwa:

1. Dalam menyelesaikan masalah klasifikasi penyakit dengan gejala demam, algoritme Support Vector Machine dapat diterapkan dengan baik. Implementasi dari algoritme Support Vector Machine untuk klasifikasi penyakit dengan gejala demam ini menggunakan 130 dataset yang memiliki 15 parameter dan dibagi dalam 3 kelas yaitu kelas demem berdarah, kelas malaria, dan kelas tifoid. Dataset dibagi menjadi data latih dan data uji dengan menggunakan metode *K-Fold Cross Validation,* dengan k=10. Sehingga data dibagi menjadi 10 bagian dan dipilih secara acak namun pada setiap bagiannya terdapat kelas yang berbeda. Setelah pembagian data latih dan data uji, untuk mendapatkan hasil klasifikasi untuk penyakit dengan gejala demam, dilakukan proses *training* dan *testing* dengan melakukan proses perhitungan yaitu perhitungan kernel dengan menggunakan kernel *polynomial,* perhitungan matriks Hessian, perhitungan nilai Ei, perhitungan nilai *delta alpha*, perhitungan nilai *alpha*, perhitungan nilai bobot, perhitungan bias, perhitungan nilai f(x) level 1, dan perhitungan nilai f(x) level 2. Hasil akhir dari implementasi peyakit dengan gajala demam adalah akurasi dari ketepatan sistem dalam membagi kelas demam berdarah, kelas malaria dan kelas tifoid.
2. Pada pengujian implementasi algoritme *Support Vector Machine* dalam menyelesaikan penyakit dengan gejala demam, didapatkan hasil akurasi terbaik dengan menggunakan metode *k-fold cross validation*, dengan k=10, ketika parameter yang digunakan adalah pembagian rasio data = 90%:10%, *lamda* = 0.5, *gamma* = 0.01, C (*complexity*) = 1 , *epsilon* = 0.0001, iterasi maksimum = 20, sehingga rata-rata akurasi yang didapatkan cukup tinggi yaitu 99.23%.

7.2 Saran

Penelitian ini dapat dikembangkan kembali dengan mengkombinasikan algoritme Support Vector Machine dengan algoritme yang dapat mengoptimasi parameter SVM. Selain itu dalam penelitian berikutnya diharapkan untuk melakukan validasi terhadap dataset terlebih dahulu, karena kevalidan data menentukan akurasi yang di dapatkan benar-benar sesuai dengan perhitungan, bukan karena kesalahan dari data yang digunakan.

DAFTAR PUSTAKA

Ai, X.X., Jia, H., Xin, L., 2016. SVM-based cancer incidence foresting of patients. *International Symposium on Computational Intelligence and Design,* [online] Tersedia di: < http://ieeexplore.ieee.org/document/7830843/> [Diakses 15 Maret 2017]

Arsin, A.A., 2012. *Malaria di Indonesia : Tinjauan aspek epidemiologi*. Makassar: Masagena press.

Beriajaya. 2009. Peranan vektor sebagai penular penyakit zoonosis. Proc. Lokarkarya Nasional Penyakit Zoonosis. Hal. 257-288.

Chandra B., 2006. *Ilmu kedokteran pencegahan dan komunitas.* Jakarta: EGC.

Cita Y.P., 2011. Bakteri Salmonella typi dan demam Tifoid. *Jurnal Kesehatan Masyarakat*, [online] Tersedia di: < http://jurnal.fkm.unand.ac.id/index.php/jkma/article/view/87.> [Diakses 20 Juni 2017]

Darsyah, M.Y., 2013. Menakar tingkat akurasi Support Vector Machine pada study kasus kanker payudara. *ResearchGate Article,* [online] Tersedia di: < https://www.researchgate.net/publication/303804988_Menakar_Tingkat_Akurasi_Support_Vector_Machine_pada_Study_Kasus_Kanker_Payudara> [Diakses 6 Juni 2017]

Liwan A.S., 2015. Diagnosis dan penatalaksanaan Malaria tanpa komplikasi pada anak. *CDK-299*, [online] Tersedia di: < http://www.kalbemed.com/Portals/6/08_229Diagnosis%20dan%20Penatalaksanaan%20Malaria%20tanpa%20Komplikasi%20pada%20Anak.pdf.> [Diakses 20 Juni 2017]

Lukman, 2016. Penerapan algoritme Support Vector Machine dalam pemilihan beasiswa : studi kasus SMK Yapimda. *Faktor Exacta*, [online] Tersedia melalui : e-journal Universitas PGRI < http://journal.lppmunindra.ac.id > [Diakses 15 Maret 2017]

Mohanty, S., Bebartta, H.N.D., 2011. Performance comparison of SVM and K-NN for Oriya character recognation. *International Journal of Advanced Computer Science and Applications (IJACSA): Specia Issue on image Processing and Analysis,* [online] Tersedia di: < http://thesai.org/Publications/ViewPaper?Volume=1&Issue=1&Code=SpecialIssue&SerialNo=16> [Diakses 15 Maret 2017]

Muis, I.A., Affandes, M., 2015. Penerapan metode (Support Vector Machine (SVM) menggunakan Kernel Radial Basis Function (RBF) pada klasifikasi *Tweet*. *Jurnal Sains, Teknologi dan Industri,* [online] Tersedia di: < http://ejournal.uin-suska.ac.id/index.php/sitekin/article/view/1010> [Diakses 15 Mei 2017]

Munir, Rinaldi., Henry, Marcelinus,. 2016. Blind steganalysis pada citra digital dengan metode Support Vector machine. [online] Tersedia di: < http://informatika.stei.itb.ac.id/~rinaldi.munir/TA/Makalah_TA_Marcelinus_Henry.pdf> [Diakses 14 Desember 2017]

Nelwan, R.H.H., 2012. Tata laksana terkini demam Tifoid. *CDK-192,* [online] Tersedia di: < http://www.kalbemed.com/Portals/6/05_192CME_1%20Tata%20Laksana%20Terkini%20Demam%20Tifoid.pdf. > [Diakses 20 Juni 2017]

Nugrahaeni, R.A., Mutijarsa, K., 2016. Comparative analysis of machine learning KKN, SVM, and Random Forest Algorithm for facial expression classification. *International Seminar on Application for Technology of Information and Communication,* [online] Tersedia di: < http://ieeexplore.ieee.org/document/7873831/> [Diakses 20 April 2017]

Paputungan, W., Rombot, D., dan Akili, R.H., 2016. Hubungan antara perilaku hidup bersih dan sehat dengan kejadian demam tifoid di wilayah kerja puskesmas Upai kota Kotamobagu tahun 2015. *Jurnal Ilmiah Informasi,* [online] Tersedia di: < https://ejournal.unsrat.ac.id/index.php/pharmacon/article/view/12215> [Diakses 6 Juni 2017]

Prasetyo, E., 2014. Data Mining *: Mengolah data menjadi informasi menggunakan matlab*. Yogyakarta: Penerbit Andi.

Purba, I.E., Wandra, T., Nugrahini, N., Nawawi, S., Kandun, N., 2016. Program pengendalian demam tifoid di Indonesia : tantangan dan peluang. *ResearchGate Article,* [online] Tersedia di: < https://www.researchgate.net/publication/313680646> [Diakses 6 Juni 2017]

Pusat Data dan Informasi Kementerian Kesehatan RI, 2016. *Malaria*. [pdf] Kementerian Kesehatan Republik Indonesia. Tersedia di: <http://www.depkes.go.id/download.php?file=download/pusdatin/infodatin/InfoDatin-Malaria-2016.pdf> [Diakses 3 Juni 2017]

Pusat Data dan Informasi Kementerian Kesehatan RI, 2016. *Profil Kesehatan Indonesia 2015*. [pdf] Kementerian Kesehatan Republik Indonesia. Tersedia di: < http://www.depkes.go.id/resources/download/pusdatin/infodatin/infodatin-dbd-2016.pdf> [Diakses 3 Juni 2017]

Putra, T.R.I., 2011. Malaria dan permasalahannya. *Jurnal Kedokteran Syiah Kuala,* [online] Tersedia di: < http://jurnal.unsyiah.ac.id/JKS/article/viewFile/3469/3231> [Diakses 20 Juni 2017]

Ramadhani, P., Wardhani, D.K., dan Nugroho, E.S., 2012. Sistem pakar diagnosa infeksi penyakit tropis berbasis web. *Jurnal Teknik Informatika,* [online] Tersedia di: <

https://aksara.pcr.ac.id/page/read_pdf.php?name=JurnalPurnamaFix1.pdf&id=21> [Diakses 9 Juni 2017]

Shofia, E.N., Putri, R.R.M., dan Arwan, A., 2017. Sistem pakar diagnosis penyakit demam : DBD, Malaria, Tifoid menggunakan metode K-Nearest Neighbor-Certainty Factor. *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer,* [online] Tersedia di: <http://j-ptiik.ub.ac.id/index.php/j-ptiik/article/download/122/62> [Diakses 9 Juni 2017]

Suhardiono, 2005. Sebuah analisis faktor risiko perilaku masyarakat terhadap kejadian Demam Berdarah Dague (DBD) di kelurahan Helvetia Tengah, Medan, tahun 2005. *Jurnal Mutiara Kesehatan Indonesia,* [online] Tersedia di: < http://repository.usu.ac.id/bitstream/handle/123456789/15364/mki-des2005-%20%287%29.pdf?sequence=1&isAllowed=y> [Diakses 15 Juni 2017]

Utama, S., Merati, T.P., 2015. Acute Febrile Illness. *Bali Infectious Disease Symposium – 7 (BIDs-7), Bali Scientific Meeting For HIV & Opportunistic Infection-3 (BAMHOI-3): Recent Diagnostic & Treatment of Common Infectious Disease & Opportunistic Infection,* [online] Tersedia di: < http://erepo.unud.ac.id/5145/1/f26193619a8cfdc4e132f332d380d69c.pdf> [Diakses 3 Juni 2017]

Wafiyah, F., Hidayat, N., dan Perdana, R.S., 2017. Implementasi algoritme Modified-K-Nearest Neighbor (MKNN). *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer,* 1 (1), p. x-x

WHO Library Cataloguing-in-Publication Data, 2014. World Malaria Report 2014. [pdf] World Health Organization. Tersedia di: <http:// http://www.who.int/malaria/publications/world_malaria_report_2014/wmr-2014-no-profiles.pdf> [Diakses 3 Juni 2017]