KLASIFIKASI PENYAKIT KULIT KUCING MENGGUNAKAN METODE *SUPPORT VECTOR MACHINE*

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:
Yuwilda Wilantikasari
NIM: 135150201111177

PROGRAM STUDI TEKNIK INFORMATIKA
JURUSAN TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2019

PENGESAHAN

KLASIFIKASI PENYAKIT KULIT KUCING MENGGUNAKAN METODE *SUPPORT VECTOR MACHINE*

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh :
Yuwilda Wilantikasari
NIM: 135150201111177

Skripsi ini telah diujii dan dinyatakan lulus pada
12 Maret 2019

Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I	Dosen Pembimbing II
Imam Cholissodin, S.Si, M.Kom	Edy Santoso, S.Si, M.Kom
NIK: 201201 8507191 1 001	NIP: 19740414 200312 1 004

Mengetahui
Ketua Jurusan Teknik Informatika

Tri Atoto 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, 23 Juli 2018

Yuwilda Wilantikasari

NIM: 135150201111177

KATA PENGANTAR

Puji dan syukur penulis panjatkan kehadirat Allah SWT atas anugerah serta limpahan rahmat-Nya sehingga penulis dapat menyelesaikan skripsi dengan judul **"**Klasifikasi Penyakit Kuit Kucing Menggunakan Metode *Support Vector Machine***".** Skripsi ini disusun sebagai syarat memperoleh gelar sarjana pada Program Studi Informatika/ Ilmu Komputer, Fakultas Ilmu Komputer Universitas Brawijaya.

Dalam proses penyelesaian skripsi ini penulis mendapatkan banyak bantuan, baik bantuan moral maupun materiil dari berbagai pihak. Oleh karena itu, penulis mengucapkan banyak terimakasih kepada:

1. Imam Cholissodin, S.Si, M.Kom I selaku Pembimbing yang telah memberikan bimbingan, arahan, ilmu, dan masukan dalam penyelesaian skripsi ini, sekaligus sebagai Dosen Pembimbing Akademik yang telah memberi dukungan moril maupun masukan selama masa perkuliahan.
2. Edy Santoso, S.Si, M.Kom selaku Pembimbing II yang juga telah memberikan bimbingan, arahan, ilmu, dan masukan dalam penyelesaian skripsi ini.
3. Bapak dan Ibu dosen yang telah mendidik dan memberikan ilmu selama Penulis menempuh pendidikan di Fakultas Ilmu Komputer Universitas Brawijaya.
4. Kedua Orang Tua dan keluarga besar yang telah mendukung penulis dengan segala usahanya, mulai dari doa, materi, dukungan moral, semangat hidup, dan tauladan yang semata-mata untuk keberhasilan penulis.
5. Kepada teman terdeket saya yang telah memberikan banyak dukungan dalam penulisan peneltian ini yaitu Muhammad Murtadho.
6. Kepada teman-teman 2011 informatika yang telah memberikan banyak dukungan dan selalu memberikan saya semangat dan harapan yaitu Muhammad Murtadho, Ersania Susena, Rohbi Harris, Muhammad Lazuardy, Sandy Yahya, Dzulfikar Muhammad dll terimakasih banyak untuk kalian.
7. Teman-teman terdekat di Malang yaitu Afrida Djulia, Tiara Erlinda, Nimas Raya, Rayindita Siwie, Chusnah Putri,
8. Firly Wahyudi, Muhammad Fachri dll yang selalu memberi semangat selama saya kuliah di Malang.
9. Kepada teman-teman 2012 informatika yang telah memberikan semangat dan dukungan yaitu Hastian Bayu dan Muhammad Ali Fachmi.

Penulis menyadari bahwa skripsi ini tidak lepas dari kesalahan dan kekurangan. Oleh karena itu, Penulis bersedia menerima kritik dan saran yang membangun untuk memperbaiki diri. Penulis berharap semoga skripsi ini dapat memberi manfaat

Malang, Juli 2018

Penulis
Wilantikasari22@gmail.com

ABSTRAK

Kucing merupakan salah satu hewan peliharaan terpopuler di dunia. Akan tetapi, kesehatan merupakan hal yang perlu diperhatikan dalam pemeliharaanya, karena di Indonesia memiliki kelembapan udara yang tinggi, sehingga parasit dan jamur dapat berkembang biak dan menyebar sehingga dapat menyebabkan penyakit kulit. Keterbatasan pengetahuan pemilik kucing tentang penyakit kulit kucing dan beberapa gejala yang memiliki kemiripan berbagai macam jenis penyakit kulit kucing yang sulit untuk diidentifikasi. Dengan adanya permasalahan tersebut, maka dibuatlah sistem cerdas yang mampu mengklasifikasikan penyakit kulit kucing berdasarkan gejalanya. Sistem cerdas ini juga bertujuan untuk membantu tim medis khususnya bidang kedokteran hewan dalam memberikan diagnosis penyakit kulit kucing. Metode *Support Vector Machine* diterapkan dengan menggunakan dataset yang masih terbatas yakni sebesar 240 dengan jumlah parameter sebanyak 14. Pada penelitian ini menggunakan lima kelas yaitu kelas scabies, cat flea, abses, dermatitis, dan jamur. Kinerja SVM ini memberikan akurasi sebesar rata – rata 98.745% menggunakan nilai parameter pada *sequential training* SVM dengan nilai $\lambda = 0.00001$ *y* = 0.01 *C* = 10 ε = 0.01 iterasi = 100 dan rasio data 90%:10%.

ABSTRACT

Cats are one of the most popular pets in the world. However, health is a matter of concern in the nurturing of cats, Indonesia has high humidity of air, hence parasites and fungi can multiply and spread which could cause skin diseases. Limited knowledge of cat owners about cat skin disease and some symptoms that have similarities to various types of cat skin disease are difficult to identify. With these problems, an intelligent system that can classify cat skin diseases based on symptoms is created. This intelligent system also aims to help medical teams especially in the field of veterinary medicine in providing a diagnosis of cat skin diseases. *Support Vector Machine* method can be applied to dental and oral disease classification problems using a limited dataset of 240 with 14 parameter. This study uses five classes of class scabies, cat flea, abscesses, dermatitis, and fungi. SVM performances provides the best result with an accuracy of 98.745% with parameter value on sequential training SVM, value nilai $\lambda = 0.00001$ y = 0.01 C = 10 ε = 0.01 iteration= 100 and the ratio of data 90% : 10%.

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR LAMPIRAN

BAB 1 PENDAHULUAN

1.1 Latar belakang

Saat ini binatang yang sering dipelihara oleh masyarakat adalah kucing, karena kucing mudah beradaptasi dan dapat dijadikan teman yang baik. Kucing memiliki bulu yang tebal dan memiliki jenis ras (Kusumaningrum, 2012). Karena indonesia memiliki kelembapan udara yang tinggi maka kesehatan dalam memelihara kucing perlu diperhatikan, agar parasit dan jamur tidak terdapat dan berkembang biak sehingga menyebabkan penyakit kulit (Hakim, 2015).

Kesehatan kucing dapat dilihat dari kondisi kulit kucing apakah terdapat penyakit atau tidak. Kondisi tersebut merupakan refleksi kesehatan kucing secara umum dan merupakan indikator terhadap adanya penyakit dalam tubuh kucing tersebut. Apabila penyakit kulit sudah menginveksi melebihi 40% area tubuh kucing maka kucing tersebut berpotensi terjangkit infeksi sekunder sehingga menyebabkan kematian. Salah satu penyakit kulit yang dapat menyebabkan kematian pada kucing adalah *scabies.* Gejala penyakit ini ditandai dengan munculnya gatal pada area kulit, sehingga kucing mengalami penurunan daya tubuh karena kehilangan nafsu makan sehingga kucing tersebut akan mati. Menurut Drh. Naumi D.R.P *scabies* merupakan penyakit yang dapat menulat kepada manusia. Walaupun penyakit ini tidak menyebabkan kematian pada manusia tetapi, penyakit ini dapat menimbukan rasa gatal yang dapat mengganggu aktivitas seseorang yang terkena penyakit ini (Palguna, 2014).

Oleh karena itu Keterbatasan pengetahuan pemilik kucing tentang penyakit kulit kucing dan beberapa gejala yang memiliki kemiripan berbagai macam jenis penyakit kulit kucing yang sulit untuk diidentifikasi seperti bulu rontok dan seringnya kucing menggaruk (Kurniati, 2017). Terdapat lima jenis penyakit kulit kucing yang memiliki gejala yang mirip, oleh karena itu dibuatlah sistem cerdas untuk mengklasifikasikan penyakit kulit kucing. Metode klasifikasi sendiri telah berkembang di dunia teknologi. Metode klasifikasi diantaranya adalah K-nearest Neighbor, Bayesian Network, dan *Support Vector Machine*. Pengertian dari klasifikasi ialah suatu proses pengelompokan data dengan tujuan untuk mengelompokan data yang dimana data mempunyai kelas label. Klasifikasi terbaik dengan menggunakan algoritme SVM (Long Wan & Wenxing Bao, 2009). Algoritme SVM digunakan pada penelitian ini untuk membuat model sistem diagnosis penyakit hewan ternak. Dalam penelitian ini menggunakan metode satu lawan satu. Pengolahan model menggunakan contoh penyakit yang diberikan oleh ahli. Setiap fungsi keputusan dilatih oleh dua kategori sampel yang sesuai. Strategi voting Max-won diadopsi ketika sampel penyakit x diklasifikasikan. Dalam percobaannya menggunakan beberapa data penyakit sapi untuk diuji. Penyakit sapi gejalanya diklasifikasikan sebagai 26 kategori, sesuai dengan 26 komponen mesin vektor pendukung. Dalam tes tersebut, terdapat lima jenis penyakit sapi. Setiap penyakit memiliki 35 kasus, 20 kasus di antaranya untuk dilatih, dan 15

kasus lainnya diuji. Hasil percobaan menunjukkan bahwa model diagnosis penyakit hewan dapat dilakukan dengan lebih akurat, cepat pada kondisi sampel kecil. Hasilnya menunjukkan bahwa sistem pakar diagnosis penyakit hewan berdasarkan mesin vektor pendukung merupakan aplikasi yang baik di bidang diagnosis penyakit hewan.

Penelitian sebelumnya yang dilakukan oleh (Reza M. 2014) memanfaatkan algoritme *Support Vector Machine* menggunakan fungsi kernel RBF dan fungsi kernel linear dalam mendiagnosis hepatitis, karena kedua fungsi tersebut memiliki parameter tertentu dan rentang nilai yang kecil yaitu [0,1]. Data yang digunakan berukuran dengan menggunakan metode sekuantial untuk menentukan nilai $\alpha = 0$, $C = 1$, $\text{epsilon} = 0{,}001$, $\text{gamma} = 0{,}5$, $\text{lambda} = 0{,}5$. Data yang digunakan berjumlah 579 dataset tes fungsi hati dibagi dua untuk data training dan data testing yang masing – masing berjumlah 100 data. Analisis kemampuan metode diketahui dengan uji coba menggunakan data testing dengan kedua kernel dengan data training 100 data positif dan 100 data negatif. Hasil uji coba dengan menggunakan fungsi kernel linier mendapatkan hasil persentase benar 68-83 % dan fungsi kernel RBF 70-96 %.

Menurut penelitian Ana Mariam Puspitasari (2018) dengan menggunakan kemampuan *One-Againts-All* dan menggunakan kernel RBF untuk pengklasifikasian data. Menggunakan 122 dataset gejala penyakit dan dengan parameter yang sudah ditentukan lambda = 0.1, gamma = 0.1, C = 1, epsilon = 1.10^{-10}, itermax = 50, dan rasio data 80%:20%. Dari hasil penelitian tersebut didapat nilai akurasi rata – rata 93.32%.

Kriteria gejala beberapa penyakit yang meliputi scabies, abses, dermatitis, jamur, dan cat flea akan digunakan dalam perhitungan dengan menggunakan metode *Support Vector Machine*. Metode *Support Vector Machine* ini digunakan untuk pengklasifikasian jenis penyakit kulit kucing. Metode tersebut terbukti dalam penelitian sebelumnya dapat memberikan nilai akurasi yang cukup tinggi, dengan menggunakan algoritme SVM diharapkan dapat membantu masyarakat awam dengan mendiagnosis jenis penyakit kulit pada kucing secara akurat dan meminimalisir tingkat kematian pada kucing.

1.2 Rumusan masalah

Dari latar belakang yang telah dijelaskan diatas, rumusan masalah didapatkan sebagai berikut:

1. Bagaimana perancangan metode *Support Vector Machine* pada klasifikasi penyakit kulit kucing dengan gejala atau parameter yang ada.
2. Bagaimana hasil akurasi yang dihasilkan dengan menggunakan metode *Support Vector Machine* pada klasifikasi penyakit kucing.
3. Bagaimana pengaruh kernel pada klasifikasi penyakit kulit kucing menggunakan metode *Support Vector Machine.*

1.3 Tujuan

Tujuan dari penulisan penelitian ini adalah:

1. Membuat klasifikasi penyakit kulit kucing pada kucing menggunakan metode *Support Vector Machine* pada sistem.
2. Mengetahui tingkat keakurasian dari metode *Support Vector Machine* pada pengujian terhadap penyakit kulit pada kucing.

1.4 Manfaat

Manfaat penelitian yang dari penelitian ini antara lain:

1. Mengimplementasikan ilmu pengetahuan teknologi dalam bidang Artificial Intelegent
2. Menambah wawasan bagaimana penerapan metode *Support Vector Machine* untuk klasifikasi penyakit kulit kucing.

1.5 Batasan masalah

Batasan masalah dibuat sehingga penelitian tidak keluar dari topik diberikan batasan masalah diantaranya:

1. Untuk penelitian ini digunakan metode *Support Vector Machine* untuk membuat sistem klasifikasi penyakit kulit kucing.
2. Data yang digunakan didapatkan dari Klinik Hewan Purple Petshop kota Malang
3. Data yang digunakan sebanyak 240 data dengan 14 gejala penyakit kulit kucing.
4. Terdapat 5 jenis penyakit diantaranya cat flea, scabies, abses, jamur, dan dermatitis.

BAB 2 LANDASAN KEPUSTAKAAN

Pada bab 2 yang membahas landasan kepustakaan terdiri dari kajian pustaka yang mendukung dalam proses penulisan penelitian ini. Landasan kepustkaan ini menjadi pendukung dimana kajian pustaka tersebut dijadikan pembanding dan mencari kelebihan ataupun kekurangan dari kajian pustaka yang sudah ada. Landasan kepustakaan yang diperlukan untuk proses penelitian ini ialah yang berkaitan dengan algoritme *Support Vector Machine (SVM).*

2.1 Kajian Pustaka

Berikut merupakan penelitian sebelumnya teori yang berkaitan dengan algoritme *Support Vector Machine* yakni penelitian tentang "Klasifikasi Tuberkulosis Dengan Pendekatan Metode *Support Vector Machine* (SVM)" (Moh.Yamin Darsyah, 2014). Penelitian yang dilakukan di Provinsi Papua Barat untuk mengindetifikasi apakah pasien terinfeksi tuberkulosis ; atau tidak. Penelitian ini menggunakan faktor penyebab terinfeksinya tuberculosis selain bakteri basil tahan asam (BTA) yaitu faktor diantaranya pendidikan (X1), jenis pekerjaan (X2), status sosial ekonomi (X3), kebiasaan merokok (X4), dan kebiasaan konsumsi alkohol (X5). Dengan metode ini disimpulkan bahwa faktor – faktor lain yang berpengaruh secara signifikan terhadap pasien sebesar (<5%) jenis pekerjaan (X2), kebiasan merokok (X4) dan kebiasaan konsumsi alkohol (X5). Sedangkan parameter pendidikan (X1) dan status sosial ekonomi (X3) tidak berpengaruh secara signifikan (>5%). Dengan metode SVM dapat disimpulkan bahwa nilai keakurasian diperoleh sebesar 98%. Dengan kesimpulan yang didapatkan, tujuan dari penelitian yang diharapkan adalah meminimalisir penyakit tuberkulosis.

Selanjutnya penelitian yang dilakukan oleh Puspita Sari (2018) dengan judul "Klasifikasi Kualitas Susu Sapi Menggunakan Metode *Support Vector Machine* (SVM)". Penelitian tersebut berisis tentang bagaimana mengklasifikasi kualitas susu sapi yang baik. Penelitian ini menggunakan metode *Support Vector Machine*. Penelitian ini menggunakan kernel RBF dimana parameter C = 0.01, gamma = 0,00001 , σ kernel rbf = 2, lambda = 0,001 , itermax = 30 dari parameter – parameter tersebut didapatkan rata – rata sebesar 92.82% dan didapatkan akurasi sebesar 94,02%.

Selanjutnya penelitian dari (Shar dan Alaa, 2013) dengan judul "Predicting The Severity Of Breast Masses With Data Mining Method" (Shar dan Alaa, 2013). Penelitian tersebut menjelaskan prediksi tingkat keparahan kanker payudara menggunakan tiga metode yaitu Decision Tree (DT), *Support Vector Machine* (SVM), dan Artificial Neural Network (ANN). Dari ketiga metode tersebut metode *Support Vector Machine* yang memiliki tingkat keakurasian paling tinggi dan menghasilkan akurasi yang terbaik (Mokhtar & Elsayad, 2013).

Penelitian sebelumnya yang dilakukan oleh Achmad Affan Suprayogi Nugraha (2018). Metode certainty factor bertujuan untuk mencari nilai kepercayaan, sedangkan metode naive bayes bertujuan untuk mencari nilai peluang

kemunculan penyakit kucing. Dilakukan pengujian dari 25 data kasus yang menghasilkan keakurasian sebesar 80%.

Penelitian yang terakhir oleh Made Bela Pramesthi Putri (2017) menggunakan metode Modified K- Nearest Neighbor pada pengklasifikasian jenis penyakit yaitu dengan memberi nilai k = 1. Penelitian ini memiliki dataset 240 data penyakit kulit kucing yang didalamnya terdapat 5 jenis penyakit dengan 14 gejala penyakit kulit kucing. Meskipun penelitian ini menghasilkan akurasi tertinggi yaitu 100% dan akurasi terendah sebesar 89.668%. Tetapi metode ini memiliki kekurangan di mana metode Modified k-Nearest Neighbor perlu menentukan nilai dari parameter K serta membutuhkan waktu dan biaya komputasi yang cukup tinggi karena diperlukan perhitungan jarak dari tiap sample uji pada keseluruhan sample latih.

Dari kajian pustaka yang sudah dijabarkan , maka penelitian ini menggunakan metode *Support Vector Machine* (SVM) untuk mengklasifikasi objek dari penelitian ini yaitu mengklasifikasi jenis penyakit kulit kucing. Landasan kepustakan mengenai teori *Support Vector Machine* (SVM) dan penyakit kulit kucing dapat dilihat seperti Tabel 2.1.

Tabel 2.1 Landasan Kepustakaan

No	Judul	Objek	Metode	Hasil
1	(Moh.Yamin Darsyah, 2014)	Tuberkulosis	*Support Vector Machine* (SVM)	Menghasilkan keakurasian sebesar 98% dengan menggunakan metode SVM
2	(Puspita Sari 2018)	Kualitas susu sapi	*Support Vector Machine* (SVM)	Hasil rata-ratwa akurasi menggunakan metode SVM yang diperoleh dari penelitian tersebut yakni 92.82% dan akurasi tertinggi sebesar 94,02%.
3	(Shar dan Alaa, 2013)	Kanker payudara	SVM, DT, ANN	Metode SVM memiliki nilai keakurasian paling tinggi dibanding metode DT dan ANN

Tabel 2.1 Landasan Kepustakaan (lanjutan)

No	Judul	objek	metode	Hasil
4	(Achmad Affan Suprayogi Nugraha, 2018)	Penyakit kucing	Naive Bayes, Certainty Factor	pengujian dari 25 data kasus yang menghasilkan keakurasian sistem pakar diagnosis penyakit kucing sebesar 80% dengan menggunakan metode naive bayes.
5	(Made Bela Pramesthi Putri 2017)	Penyakit kulit kucing	Modified K-Nearest Neighbor	Penelitian ini memiliki dataset sebanyak 240 dengan data penyakit 14 parameter dan 5 jenis penyakit kulit yang berbeda. menghasilkan akurasi tertinggi yaitu 100% dan akurasi terendah sebesar 89.668%.

2.2 Penyakit Kulit Kucing

2.2.1 Scabies

Salah satu penyakit yang sering terjadi pada kucing adalah *scabies*, *scabies* sendiri merupakan penyakit kulit kucing yang sering disebabkan karena infestasi parasit sejenis tungau yang hidup pada folikel kulit dan hanya dapat dilihat oleh mikroskop. Penyakit ini menyerang pada kucing dengan ditandai kucing gatal gatal. Penyakit menyebakan kerusakan jaringan kulit, dan apabila tidak dilakukan pengobatan maka akan menyebakan kerusakan sekunder atau pembusukan pada kulit kucing. Penyakit scabies merupakan penyakit bersifat zoonosis yang artinya dapat menular kepada manusia. Manusia yang tertular akan mengalami gejala gatal – gatal dan kulit akan mengalami peradangan (drh. Yusni, 2015).

Drh. Naumi D.R.P menyebutkan bahwa penyakit Scabies dapat ditularkan dari kucing kepada pemilik kucing, penyakit ini akan mengganggu aktivitas pemilik kucing. Scabies pada manusia tidak akan menyebabkan kematian, oleh sebab itu penyakit scabies harus ditangani agar tidak menyebabkan kematian pada kucing atau infeksi tidak menyebar kepada pemilik kucing (Pitaloka,2014).

Gambar 2.1 Scabies (Effendi, 2012)

2.2.2 Abses

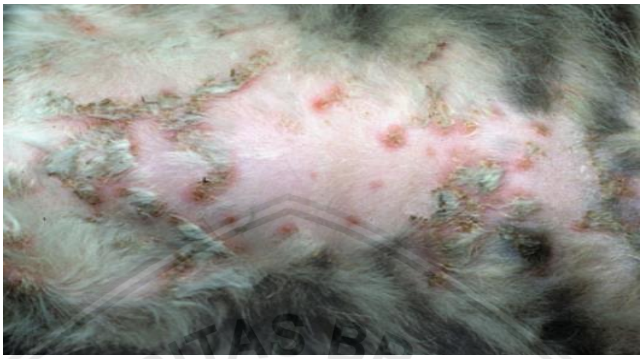
Abses yaitu penyakit kulit pada kucing berupa benjolan atau pembengkakan yang terdapat pada dibawah kulit dan menyebabkan rasa sakit. Penyakit kulit ini disebabkan karena luka tusuk benda tajam dan gigitan kucing lain. Luka tusuk biasanya disebabkan serpihan kaca, kawat yang menempel pada lapisan kulit, dan peluru senapan. Oleh sebab itu benda yang menempel merupakan sumber iritasi yang menyebabkan infeksi dan nanah pada kucing. Apabila dibiarkan maka akan berdampak buruk dan kucing akan mati (Aris Baskoro, 2014).

Gambar 2.2 Abses (Way, 2014)

2.2.3 Jamur

Penyakit jamur yang sering dijumpai pada kucing disebabkan karena Indonesia sendiri merupakan negara tropis dan memiliki kelembapan yang tinggi, penyakit jamur mudah berkembang dan tumbuh pada kucing. Jamur juga disebut sebagai *dermatofitosis* atau *ringworm* merupakan jenis penyakit kurap kucing dan bukan disebabkan oleh cacing. *Trychophyton tonsuranse* merupakan salah satu penyakit jamur sejenis *ringworm*. Jamur *trychophyton tonsurans* berkembang pada sekitar kulit kucing dan menyebabkan meluas ke dalam folikel rambut tetapi tidak sampai ke akar. Jamur pada kucing biasanya memiliki ciri-ciri dimana terdapat adanya lesi melingkar dan dapat ditemukan pada seluruh bagian kulit, penyakit ini sering

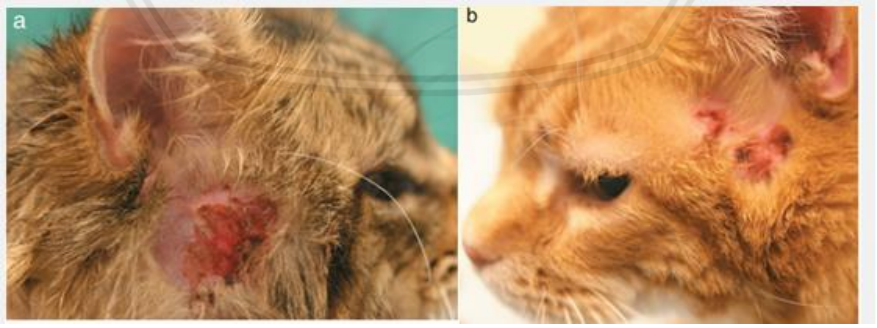
terdapat pada kulit kaki dan bagian yang sering bersentuhan dengan obyek yang terkontaminasi oleh parasit. Apabila penyakit ini tidak ditangani makan menyebabkan bagian kulit menjadi rontok dan mengalami kebotakan. Apabila dibiarkan maka penyakit ini akan menjadi infeksi tingkat lanjut dan ruam merah pada kulit kucing. Kucing yang mengidap penyakit ini dapat menularkan penyakit ini kepada manusia apabila manusia tersebut melakukan kontak fisik, oleh karena itu penyakit jamur pada kucing harus ditangani secara tepat (Brumley,1921).



Gambar 2.3 Jamur (Kusumaningrum, 2013)

2.2.4 Dermatitis

Penyakit dermatitis biasanya disebabkan oleh kucing yang mengalami alergi makanan, alergi pada air liur kutu, dan gangguan lingkungan. Jika hewan yang alergi terhadap kutu akan menggaruk seluruh tubuh. Hal yang harus dilakukan dalam penanganan ini diantaranya pertama yakni mencari tahu penyebab dari penyakit dermatitis, setelah itu dilanjutkan dengan manifestasinya. Penyakit menyabkan gatal-gatal karena alergi pada kucing sehingga penanganan berikutnya yakni menghilangkan gatal- gatal yang disebabkan alergi dengan membersihkan kullit yang terkontaminasi dan memberikan antiseptik dengan rutin. Untuk mengatasi kucing dapat dimandikan dengan air karbonat untuk menghilangkan gatal – gatal pada kucing. (Brumley, 1921).



Gambar 2.4 Dermatitis (Kusumaningrum, 2013)

2.2.5 Pinjal (*Cat Flea*)

Penyakit pinjal atau cat flea sering disebut sebagai parasit, parasit tersebut dapat dilihat dengan mata telanjang pada kulit kucing, parasit tersebut tidak memiliki sayap yang berukuran yakni 2mm sampai 5 mm. Disebut *ordo siphonoptera* memiliki karakteristik diantaranya berbentuk pipih dan tidak

memiliki sayap. Parasit ini bermetamorfosis sempurna, bermula dari telur berkembang jadi larva, menjadi pupa dan akhirnya menjadi imago. Parasit ini memiliki alat mulut penusuk atau penghisap dan bermata tunggal (Pratiwi, 2007). Selain menyababkan gatal – gatal pada kulit kucing dapat terlihat fases atau kotoran dari pinjal yang berwarna merah.

Gambar 2.5 Pinjal (Effendi, 2012)

2.2.6 Gejala Penyakit Kulit Kucing

Beberapa gejala yang dialami pada kucing ditunjukan oleh Tabel 2.2:

Tabel 2.2 Gejala Penyakit Kulit Kucing

No	Nama Gejala
1	Krusta
2	Kutuan
3	Gatal pada Kulit
4	Kulit Keropeng
5	Bulu Rontok
6	Kulit Kemerahan
7	Pioderma
8	Cairan Kuning (nanah)
9	Luka
10	Bulatan cincin pada kulit
11	Ketombe
12	Benjolan Lunak
13	Kulit Bersisik
14	Kotoran berwarna merah kecoklatan pada kulit

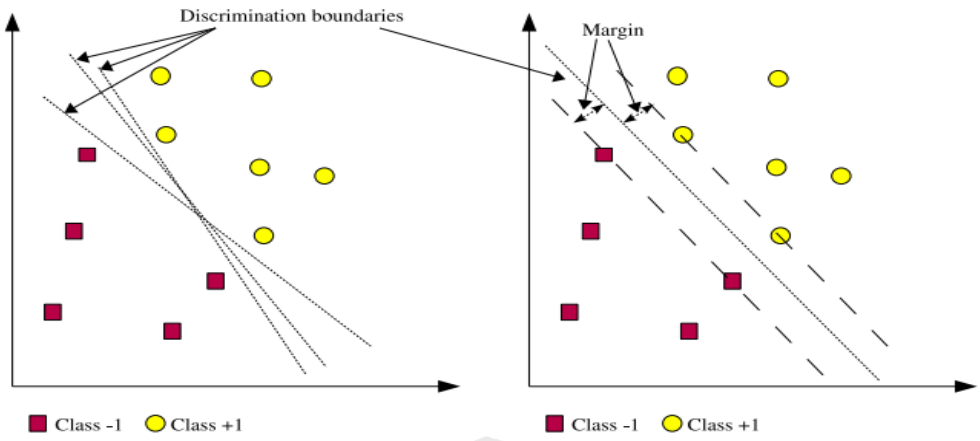
2.3 Klasifikasi

KIasifikasi merupakan pekerjaan untuk melakukan pengelompokan pelatihan pada fungsi target *f ,* fungsi tersebut mempunyai kegunaan untuk memetakan suatu set kedalam suatu kelas, yang dimana kelas tersebut sudah ditentukan ketegorinya. Atribut – atribut yang memiliki kesamaan dari suatu objek akan dikelompokan pada sebuah model yang disebut klasifikasi. Klasifikasi membutuhkan data training dan data testing, data training digunakan sebagai tahapan pengelompokan dan data testing digunakan sebagai pengujian. (Winarko2 & Suwanto, Oktober 2014). KIasifikasi merupakan algoritme yang terbagi menjadi dua macam diantaranya adalah *lazy learner* dan *eager learner*. *Eager learner* ialah sebuah pelatihan dimana proses perhitungan pada peIatihannya membutuhkan waktu yang Iebih Iama tetapi mendapatkan proses prediksi yang reIatif cepat. *Lazy I*earner pada prosesnya pelatihannya sangat efektif, tetapi kelemahannya ialah pada proses prediksinya membutuhkan waktu yang Iama. *K-Nearest Neighbor* (K-NN) merupakan algoritme dalam pengklasifikasian tak hanya itu ada beberapa algoritme diantara *Fuzzy K-Nearest Neighbor* (FK-NN)*,* dan *Decision Tree.*

2.4 *Support Vector Machine* (SVM)

Algoritme *Support Vector Machine* (SVM) dikembangkan oleh penelIti yang bernama Boser, Guyon, Vapnik, dan penelitian ini pertama kali diumumkan pada tahun 1992 di Annual Workshop on Computational Learning Theory. Algoritme SVM sangat baik dalam mengklasifikasikan data latih untuk proses prediksi, karena pada sejumlah data latih, hanya sebagian kecil dari data yang digunakan untuk melakukan proses prediksi. Oleh sebab itu pada saat proses perhitungan perulangan SVM, tidak seluruh data latih akan digunakan. Hal itu yang membedakan algoritme SVM dengan SVM lainnya. SVM alah algoritme yang dapat mengolah data set yang mempunyai dimensi yang tinggi. Algoritme SVM bertujuan untuk memaksimalkan batas nilai *hyperplane.* Algoritme SVM juga mencari nilai maksimal dari margin, agar generalisasi pada klasifikasi lebih baik. Inti dari SVM adalah mencari *hyperplane*, yakni dengan cara mencari jarak atau margin yang maksimal diantara data yang paling dekat dengan garis *hyperplane*. (Anto Satrio Nugroho, 2007).

Algoritme SVM memiliki kemampuan yang pada penerapannya dikatakan baik dalam proses mengolah data. SVM menggunakan *hyperplane*, *hyperplane* linear digunakan untuk memisahkaan data dengan garis linear yang tidak dapat dipisahkan. Untuk mengklasifikasi teknik SVM dapat memaksimalkan nilai margin dan memaksimaIkan batas nilai *hyperplane*. Pada SVM nonlinear prosesnya membutuhkan fungsi yaitu *kernel trick*, karena data tidak dapat dipisahkan hanya dengan garis linear. SVM nonlinear dapat mengolah data dari data berdimensi rendah menjadi data dengan dimensi yang lebih tinggi.



Gambar 2.6 Pemisah *Hyperplane* dari kedua kelas yaitu -1 dan +1

Sumber : (Nugroho, et al., 2003)

Dari gambar diatas sebelah kiri terdapat dua buah data yang ditunjukan oleh pattern, pattern tersebut merupakan perwakilan kelas, dimana terdpat dua kelas yakni kelas -1 dan +1. Pada gambar pertama terdapat beberapa garis linear yang memisahkan dua kelas tersebut, tetapi fungsi SVM sendiri adalah mencari garis *hyperplane* yang memiliki nilai margin paling besar. Pada gambar sebelah kanan terdapat satu garis yang dapat memisahkan dua buah kelas dengan nilai margin paling besar. Data yang berada pada bidang pembatas disebut *support vector*. Intinya bidang pembatas pertama yang memisahkan kelas pertama dan bidang pembatas kedua memisahkan kelas kedua sehingga diperoleh:

$$(w.x_i + b) = 0 \quad (2.1)$$

$$(w.x_i + b) \geq -1 \quad (2.2)$$

$$(w.x_i + b) \geq +1 \quad (2.3)$$

W adalah normal bidang atau bobot *vector* terhadap *hyperplane*. Pattern (x_i) merupakan data i yang keberapa dan b merupakan posisi bidang yang relatif terhadap koordinat. Nilai margin didapatkan dari $\frac{1-b-(-1-b)}{w}$ = $\frac{2}{w}$ nilai margin dipenuhi dengan mengalikan b dan w dengan konstanta. Untuk mendapat nilai margin terbesar yaitu dengan melihat posisi margin terbesar yaitu dengan dengan cara memaksimalkan nilai titik terdekatnya dari jarak *hyperplane*, yaitu $1/\| \varpi \|$. Dari uraian tersebut didapatkan rumus *Quadratic Programming (QP) Problem,* yaitu untuk mendapatkan titik dengan nilai terendah menggunakan persamaan (2.4) dengan tetap memperhitungkan batasan pada persamaan (2.5)

$$\min \tau(w) = \frac{1}{2} \| w \|^2 \quad (2.4)$$

$$y_i (x_i . w + b) - 1 \geq 0, \forall i \quad (2.5)$$

langkah selanjutnya pada SVM sebelum dapat mendapatkan hasil klasifikasi dari tiap kelas akan dilakukan inisiasi data yang dinyatakan x_i, y_i yang mana i merupakan

1,2... N dan x_i meruoakan *x1, x2,..... xik* merupakan parameter pada data latih untuk $y_i \in \{-1, +1\}$ yang artinya perwakilan dari suatu kelas.

2.4.1 Sequential Training

*Quadratic Programming (*QP) adalah fungsi yang dimana didalamnya terdapat cara untuk mendapatkan *hyperplane* yang maksimal yaitu menggunakan SVM dengan cara analisis numerik. Pada SVM terdapat margin geometris dimana akan memberikan dampak kesalahan minimal pada proses klasifikasi dan mendapatkan hasil yang maksimal. Tetapi pada prosesnya memiliki proses yang kompleks dan rumit sehingga waktu yang dibutuhkan lebih lama oleh sebab itu prosesnya membutuhan proses *Sequential Training* seperti yang telah dikembangkan oleh (Vijayakumar S, 1999) seperti dibawah ini:

1. Langkah pertama ialah $\alpha_1 = 0$. Yaitu menghitung perkalian kernel gaussian menggunakan nilai Y maka sehingga mendapatkan nilai matriks *Hessian.* α_1 dicari agar mendapatkan nilai dari perhitungan *support vector*. Untuk proses perhitungan tiap data dari i sampai dengan data j, didapatkan dengan memakai persamaan Matriks *Hessian* yang akan ditunjukkan seperti dibawah ini :

$$D_{ij} = y_i y_j (K (x_i, x_j) + \lambda^2) \quad (2.6)$$

2. Selanjutnya dilakukan perhitungan dengan persamaan dibawah ini :

$$E_i = \sum_{j=1}^{i} \alpha_j D_{ij} \quad (2.7)$$

$$\delta\alpha_i = \min \{\max[y (1 - E_i) - \alpha_i] C - \alpha_i \} \quad (2.8)$$

$$\alpha_i = \alpha_i + \delta\alpha_i \quad (2.9)$$

Keterangan :

α_i = alfa ke-*i*

D_{ij} = matriks *Hessian*

E_i = *error rate*

C = konstanta C

$\delta\alpha_i$ = delta alfa ke-*i*

Selanjutnya kembali ke langkah 2 sampai nilai α_i untuk mendapatkan konvergen. Jika terdapat perubahan pada nilai α_i maka kita mendapatkan konvergensi. Setelah itu mencari nilai w dan b dengan menggunakan rumus seperti dibawah ini:

$$w = \alpha_i y_i y_j \phi (x_i) \quad (2.10)$$

$$b = -\frac{1}{2} (w. x^+ + w. x^-) \quad (2.11)$$

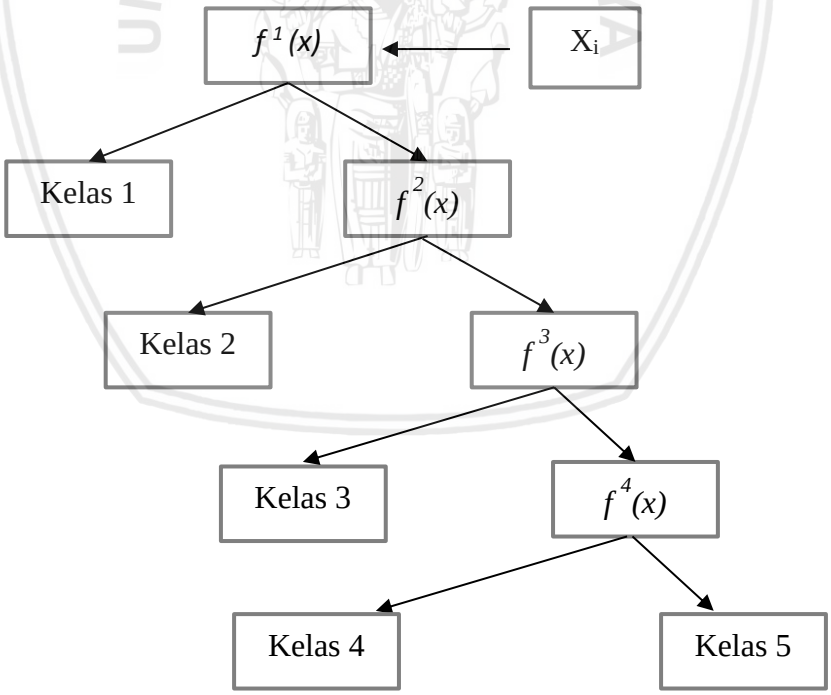
2.4.2 *One-Againts-All*

Proses selanjutnya ialah menggunakan metode *One-Againts-All* yaitu strategi dengan cara membangun sejumlah nilai *k* dan dimasukan pada model SVM. Pada prosesnya terdapat pelatihan klasifikasi ke-i dilatih menggunakan semua seluruh data. Jika terdapat kasus terdapat klasifikasi dengan 5 buah kelas maka pada hanya 5 buah SVM biner pada proses pelatihannya (Sembiring, September 2007). Berikut persamaan metode *One-Againts-All* ditunjukan oleh Tabel 2.3

Tabel 2.3 Contoh 5 SVM dengan metode *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	Bukan kelas 2	$f^2(x) = (w^2)\,x + b^2$
Kelas 3	Bukan kelas 3	$f^3(x) = (w^3)\,x + b^3$
Kelas 4	Bukan kelas 4	$f^4(x) = (w^4)\,x + b^4$
Kelas 5	Bukan kelas 5	$f^5(x) = (w^4)\,x + b^5$

Sumber : (Sembiring, September 2007)



Gambar 2.7 Gambar klasifikasi dengan metode One-Againts-All

Sumber : **(Sembiring, September 2007)**

Pada perhitungan SVM non-linear menggunakan fungsi kernel yang dibutuhkan untuk pemetaan data yang lama dapat pindah ke fitur yang baru. Fungsi kernel ini disebut dengan *kernel trick*. *Kernel trick* digunakan pada proses

perhitungan SVM *non-linear* berikut fungsi kernel *non-linear* ditunjukan pada Tabel 2.4:

Tabel 2.4 Fungsi Kernel *non-linear*

Jenis Kernel	Definisi
Polynomial	$\kappa(\vec{x}_i, \vec{x}_j) = (\vec{x}_i, \vec{x}_j + c)^p$
RBF	$\kappa(\vec{x}_i, \vec{x}_j) = \exp\left(-\frac{\|\vec{x}i - \vec{x}j\|}{2\sigma}\right)$
Sigmoid	$\kappa(\vec{x}_i, \vec{x}_j) = \tanh(\alpha\vec{x}_i, \vec{x}_j + \beta)$

2.4.3 Kelebihan dan Kekurangan *Support Vector Machine* (SVM)

SVM sendiri memiliki beberapa kelebihan dibandingkan dengan metode lainnya, akan tetapi SVM juga terdapat kekurangan. Berikut kelebihan (Anto, et al., 2003):

- Kelebihan metode SVM:
    a. Generalisasi adalah kelebihan metode ini dimana pada proses klasifikasi suatu pattern tetapi pattern tidak masuk kedalam data yang dipakai pada prosesnya.
    b. *Curse of dimensionality* adalah pengenalan metode pattern agar dapat mengestimasikan parameter.
    c. *Feasibility* adalah proses klasisfikasi yang menggunakan QP problem pada SVM.
- Kelemahan pada metode SVM yakni sulit untuk digunakan pada kasus yang berskla besar.

BAB 3 METODOLOGI

Pada bab metodologi yang dibahas adalah metode dan langkah – langkah dalam pembuatan sistem untuk mengklasifikasi penyakit kulit kucing dengan menggunakan metode SVM. Hasil yang dikeluarkan pada proses klasifikasi menjadi lima kelas yaitu diantaranya *pinjal (Cat Flea), abses, dermatitis, scabies, jamur.*

3.1 Tipe Penelitian

Tipe penelitian Klasifikasi kulit kucing menggunakan algoritme *Supoort Vevtor Machine (SVM)* merupakan tipe penelitian non implementatif karena tujuan penelitian ini dilakukan hanya menghasilkan suatu deskripsi. Pendekatan yang dilkukan pada penelitian ini merupakan pendekatan pada deskripsi dari kegiatan penelitian non-implementatif.

3.2 Strategi Penelitian

Strategi penelitian ini bertujuan untuk mengklasifikasi penyakit kulit kucing menggunakan algoritme *Supoort Vevtor Machine (SVM).* Strategi yang dilakukan pada penelitian ini dikontrol oleh peneliti dan pengembang yang bertanggung jawab. Metode yang digunakan adalah metode algoritme *Supoort Vevtor Machine (SVM)* dengan menngunakan *One Against All* yang memiliki kelas lebih dari dua kelas.

3.3 Teknik Pengumpulan Data

Pada bab metodologi langkah pertama yang dilakukan adalah mengumpulkan data yang dibutuhkan dalam pembuatan penelitian ini. Penelitian ini membutuhkan data, diantaranya dataset yang berjumlah 240 data. Dataset tersebut didapatkan dari penelitian yang telah dilakukan oleh Made Bela pada tahun 2017. Penelitian tersebut bertujuan untuk mendiagnosis penyakit kulit kucing dengan menggunakan Modified K-Nearest Neighbor. dengan jumlah kelas yang ada pada data ini sebanyak 5 kelas yaitu *pinjal (Cat Flea), abses, dermatitis, scabies, jamur*. Terdiri dari 14 parameter pada Tabel 3.1.

Tabel 3.1 Gejala Penyakit Kucing

No	Nama Gejala	Parameter
1	Krusta	L1
2	Kutuan	L2
3	Gatal pada Kulit	L3
4	Kulit Keropeng	L4
5	Bulu Rontok	L5
6	Kulit Kemerahan	L6

7	Pioderma	L7
8	Cairan Kuning (nanah)	L8
9	Luka	L9
10	Bulatan cincin pada kulit	L10
11	Ketombe	L11
12	Benjolan Lunak	L12
13	Kulit Bersisik	L13
14	Kotoran berwarna merah kecoklatan pada kulit	L14

3.4 Perancangan Sistem

Pada perancangan sistem, sistem yang dibuat bertujuan untuk mengklasifikasi penyakit kulit kucing menggunakan *Supoort Vevtor Machine (SVM).* Pada tahap perancangan sistem menjabarkan mengenai perancangan kinerja sistem secara keseluruhan, dimulai dengan *input* yang diproses dan menghasilkan *output*. Pada proses yang dilakukan antara lain untuk mengklasifikasi beberapa penyakit kulit kucing menggunakan *Supoort Vevtor Machine (SVM)*. Terdapat 3 alur perancangan sistem yaitu *input,* proses, dan *output.* Analisis kebutuhan dilakukan untuk mengetahui kebutuhan apa saja yang dibutuhkan dalam penelitian ini. Berikut merupakan daftar kebutuhan dalam penelitian ini:

1. Input
   Input atau masukan pada sistem untuk mengklasifikasi jenis penyakit kulit kucing dengan memasukan parameter-parameter yang sudah ditentukan pada algoritme *Supoort Vevtor Machine (SVM).*

2. Proses
   Proses perhitungan pada sistem ini menggunakan algoritme *Supoort Vevtor Machine (SVM).*

3. Output
   Output atau keluaran dari sistem ini adalah hasil dari pengklasifikasian mengklasifikasi jenis penyakit kulit kucing dengan memasukan parameter-parameter yang sudah ditentukan pada algoritme *Supoort Vevtor Machine (SVM).* Hasil ini berupa keluaran kelas dan akurasi yang didapatkan.

3.5 Pengujian dan Analisis Algoritme

Pada sub bab ini akan menjabarkan pengujian yang akan dilakukan untuk mengetahui apakah sistem telah sesuai dengan spesifikasi kebutuhan agar tidak terjadi *error*. Berikut pengujian yang dilakukan pada penelitian ini :

1. Pengujian terhadap fungsi *kernel.*
2. Pengujian fungsi parameter yang ada pada *Sequential Training* SVM :

- Pengujian terhadap rasio data pada kernel RBF .
- Pengujian terhadap parameter λ *(Lambda)* pada kernel RBF.
- Pengujian terhadap parameter C *(Complexity)* pada kernel RBF.
- Pengujian terhadap parameter *Y (Gamma)* pada kernel RBF.
- Pengujian terhadap nilai iterasi maximal *(Itermax)* pada kernel RBF.

3.6 Penarikan Kesimpulan dan Saran

Pada tahap akhir dalam metodologi ini akan menyimpulkan hasil dari penelitian yang sudah dilakukan. Kesimpulan dapat dilakukan setelah semua tahapan penelitian telah selesai. Pada kesimpulan yang ditulis diharapkan dapat menjawab dari rumusan masalah yang dibuat. Saran dibuat agar penelitian selanjutnya bisa memperbaiki kekurangan yang ada, dan dapat memberikan pertimbangan agar dijadikan pada pengembangan sistem selanjutnya.

BAB 4 PERANCANGAN

Bab 4 yakni membahas perancangan sistem yang berisikan tentang formulasi permasalahan, deskripsi data, dan diagram–diagram perancangan sistem yang akan dibuat. Selain itu juga berisi siklus tahap penyelesaian klasifikasi pada penyakit kulit kucing berdasarkan gejalanya dengan menggunakan algoritme *Support Vector Machine* dan teori *One-Againts-All.* Pada bab perancangan akan dijelaskan tentang algoritme serta proses untuk mendapatkan akurasi dari metode SVM.

4.1 Formulasi Permasalahan

Pada sub bab ini permasalahan yang akan dibahas ialah klasifikasi penyakit kulit kucing dengan menggunakan lima kelas, yaitu kelas jamur, abses, dermatitis, scabies, dan cat flea. Untuk menentukan pengklasifikasian digunakan metode *Support Vector Machine*, karena metode ini baik dalam mencari hyperlane. Sistem yang dirancang nantinya akan dimasukan data penyakit kulit kucing dalam format .xls. Pada proses perhitungan manual *Support Vector Machine* data tersebut akan digunakan. Terdadapat 14 parameter gejala penyakit kulit kucing, angka yang digunakan merupakan angka desimal selisih nilai yang tinggi sehingga angka perlu dinormalisasi dan dihitung jarak *euclidien* agar mendapatkan pembatas *hyperline* terbaik. Setelah dinormalisasi data masuk ke proses SVM menggunakan metode kernel RBF. Setelah itu data dihitung dengan metode *sequential training*. Proses perhitungan training SVM diperlukan nilai epsilon, alpha, dan delta alpha. Selanjutnya menghitung proses testing SVM, di dalam proses perhitungan testing mencari nilai fungsi x, wx^+ , wx^- , dan nilai b atau bias. Proses yang terakhir adalah menghitung *One-Againts-All,* karena data dimiliki lebih dari dua kelas.

4.2 Deskripsi Data

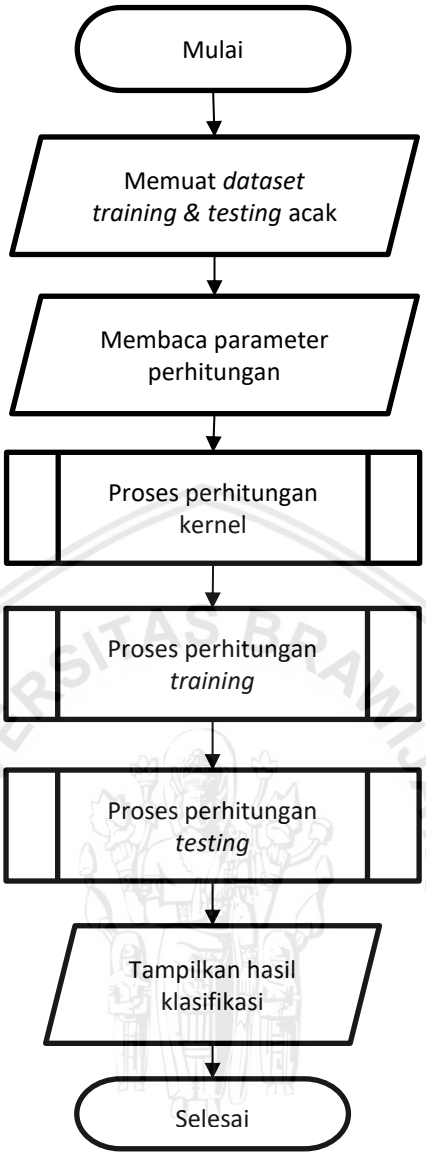
Pada sub bab ini menjelaskan deskripsi tentang data yang akan digunakan pada proses perhitungan manualisasi dalam penelitian ini. Data yang digunakan pada penelitian ini berasal dari pakar. Pada penelitian ini terdapat 14 parameter, 14 parameter tersebut merupakan gejala-gejala penyakit kulit kucing. Pada proses pehitungan manualisasi terdapat 5 data testing dan 15 data *training*. Data *testing* tersebut kemudian terdapat menjadi empat kelas, yaitu kelas *scabies*, *cat flea*, abses, dermatitis, dan jamur. Kelas K1 merupakan penyakit scabies yang diberi nilai 1, kelas K2 yang merupakan kelas Cat Flea, kelas K2 merupakan penyakit Cat Flea, kelas K3 merupakan penyakit dermatitis dan K4, dan kelas K5 merupakan adalah jamur, keempat kelas akan diberikan nilai -1. Tabel 4.1 menunjukkan sampel dataset penyakit kulit kucing yang sudah di normalisai sehingga, rentang nilai yang ditampilkan tidak tertalu tinggi selisihnya agar mendapatkan *hyperline* terbaik.

Tabel 4.1 Tabel sampel dataset penyakit kulit kucing

No	GEJALA														
	G1	G2	G3	G4	G5	G6	G7	G8	G9	G10	G11	G12	G13	G14	KELAS
1	1	0	1	1	1	0	0,6	0	1	0	0	0	0	0	K1
2	0,88	0	1	1	0,6	1	0,6	0	0,88	0	0	0	0	0	K1
3	0,88	0	1	1	0,6	1	0,6	0	1	0	0,6	0	0	0	K1
4	0	1	0,6	0	0,6	0	0	0	1	0	0	0	0	1	K2
5	0	1	0,6	0	0,6	0	0,6	0	1	0	0	0	0	1	K2
6	0	1	0,6	0	0,6	0	0,6	0	1	0	0	0	0	1	K2
7	0	0	0	0	0	0	0	0	0	0	0	1	0	0	K3
8	0	0	0	0	0	0	0	0	0	0	0	1	0	0	K3
9	0	0	0	0	0	1	0,6	1	0	0	0	1	0	0	K3
10	0	0	0,6	0	0,6	0	1	0	0,75	0	0	0	0	0	K4
11	0	0	0,6	0	0,6	0	1	0	1	0	0	0	0	0	K4
12	0	0	0,6	0	0,6	0	1	0	1	0	1	0	0	0	K4
13	0	0	0,6	0,6	0,6	0,6	1	0	0	1	0,6	0	1	0	K5
14	0	0	0,6	0,6	0,6	0,6	1	0	0	1	0,6	0	1	0	K5
15	0	0	0,6	0,6	0,6	0,6	1	0	0	1	1	0	1	0	K5

4.3 Alur Proses Algoritme *Support Vector Machine*

Pada sub bab ini untuk proses perhitungan algoritme SVM adalah dilakukannya pengambilan dataset secara acak dengan rasio tertentu. Dataset yang didapatkan dari pakar memiliki nilai desimal yang mana tiap kelas memiliki nilai selisih yang tinggi, sehingga data perlu dinormalisasi. Untuk proses perhitungan SVM dibutuhkan angka desimal yang tidak memiliki nilai selisih yang tinggi agar mendapatkan *hyperline* terbaik. Setelah data dimuat secara acak, kemudian dilakukan proses perhitungan *training* menggunakan metode SVM pada data latih yang dimuat secara acak dari *dataset*. Proses perhitungan *training* SVM terdiri dari beberapa proses lain yang akan dijelaskan pada sub bab selanjutnya. Kemudian hasil dari proses perhitungan *training* SVM digunakan untuk proses perhitungan *testing* yang juga terdiri dari beberapa sub proses lain. Proses pengujian dilakukan menggunakan data uji yang diambil secara acak dari dataset sesuai rasio yang dipilih. Setelah proses pengujian (*testing*) selesai, akan didapatkan hasil klasifikasi penyakit kucing. Hasil dari perhitungan akan dibandingkan dengan dataset sesungguhnya untuk mengetahui tingkat akurasi perhitungan. Setelah didapatkan nilai akurasi, semua data perhitungan akan ditampilkan pada tampilan aplikasi. Alur proses algoritma *support vector machine* ditunjukan pada Gambar 4.1.



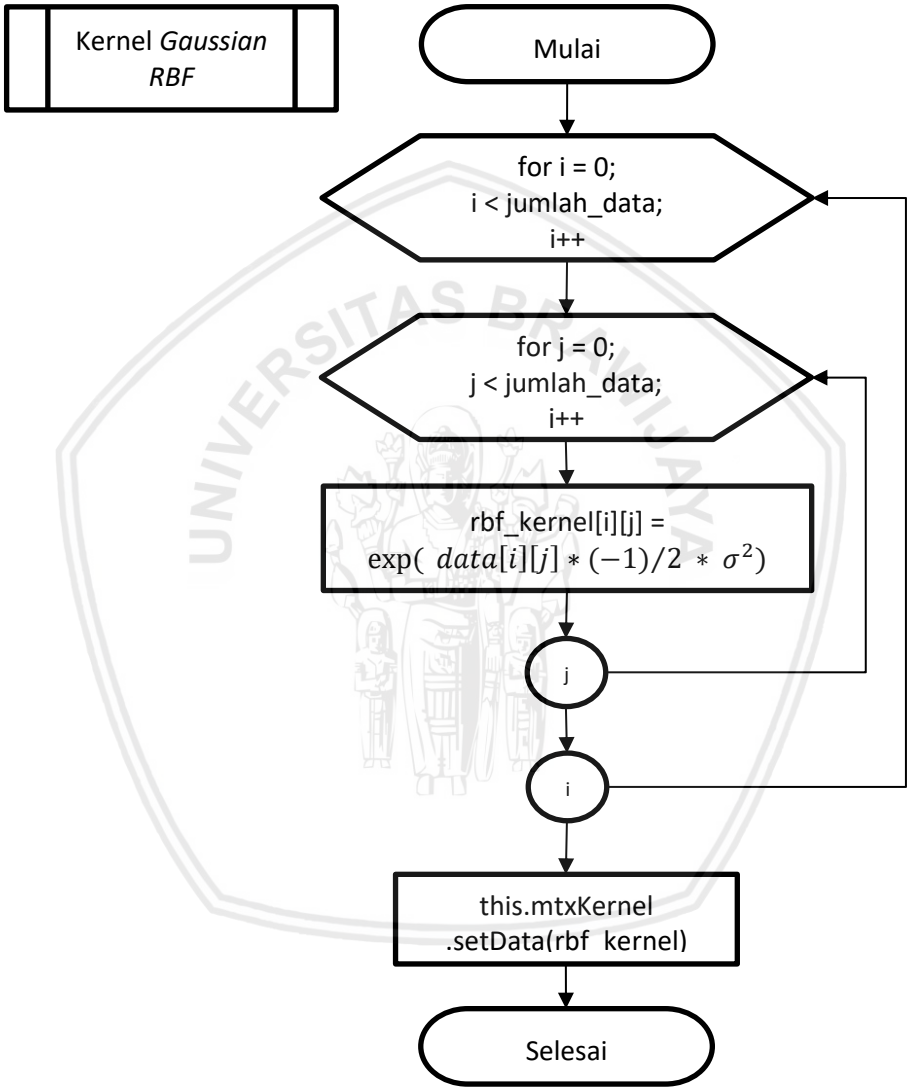
Gambar 4.1 Diagram alir algoritme *Support Vector Machine*

Dari Gambar 4.1 yang menunjukan diagram alir algoritme SVM akan dijelaskan sebagai berikut:

1. Memuat data *training* dan *testing* secara acak dari *dataset* yang telah dihimpun pada file *spreadsheet*.
2. Membaca *input* parameter α, λ, γ, ε, σ, C, rasio data, dan jenis kernel yang digunakan untuk perhitungan.
3. Melakukan proses perhitungan kernel sesuai dengan jenis kernel yang dipilih, ada 4 macam kernel yang digunakan pada penelitian ini.
4. Melakukan proses perhitungan *training*.
5. Melakukan proses perhitungan *testing*.
6. Menampilkan hasil klasifikasi berdasarkan perhitungan yang telah dilakukan sebelumnya.

4.3.1 Proses Perhitungan Kernel *Gaussian RBF*

Pada sub bab ini akan menjelaskan perhitungan yaitu proses perhitungan *kernel SVM* yaitu *kernel Gaussian RBF.* Proses pertama yang dilakukan adalah melakukan perulangan sebanyak jumlah baris data, kemudian data training yang sudah dimasukan akan dihitung menggunakan *kernel RBF.* Berikut merupakan diagram alir dari proses perhitungan kernel *Gaussian RBF* pada Gambar 4.2.



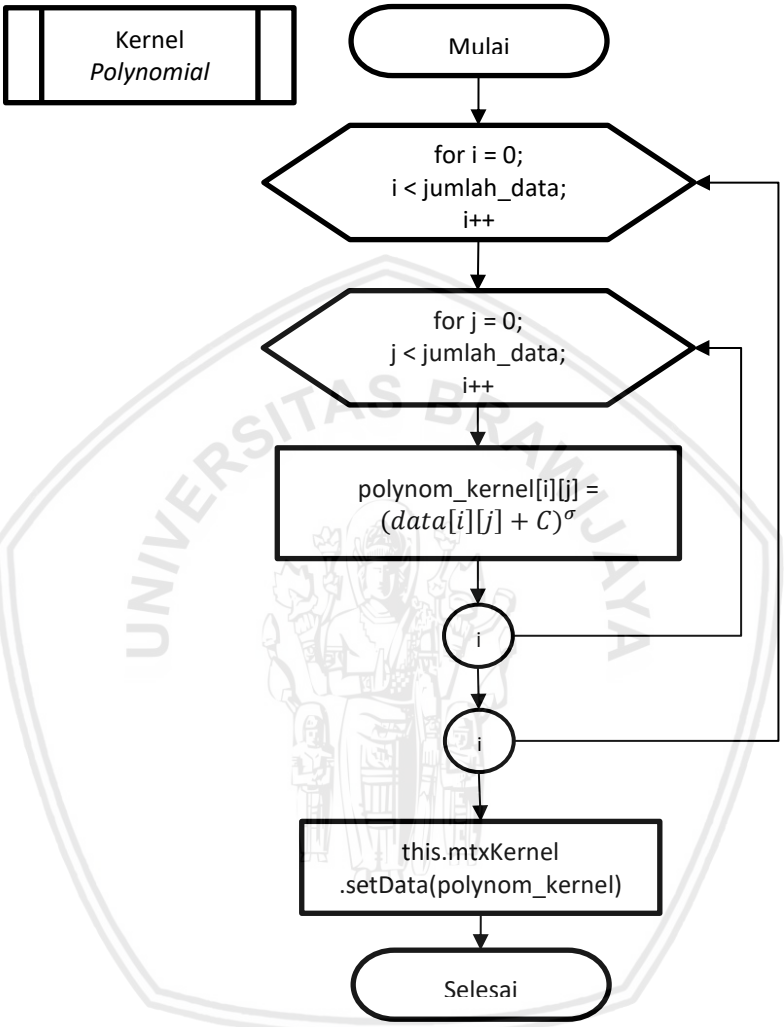
Gambar 4.2 Diagram alir perhitungan *kernel Gaussian RBF*

Berikut penjelasan mengenai Diagram Alir Perhitungan Kernel SVM berdasarkan Gambar 4.2:

1. Proses perulangan dengan indeks *i* dan *j* sebanyak jumlah baris data.
2. Perhitungan *kernel Gaussian RBF* pada data *training.*
3. Menyimpan nilai hasil perhitungan pada properti matriks kernel.

4.3.2 Proses Perhitungan *Kernel Polynomial*

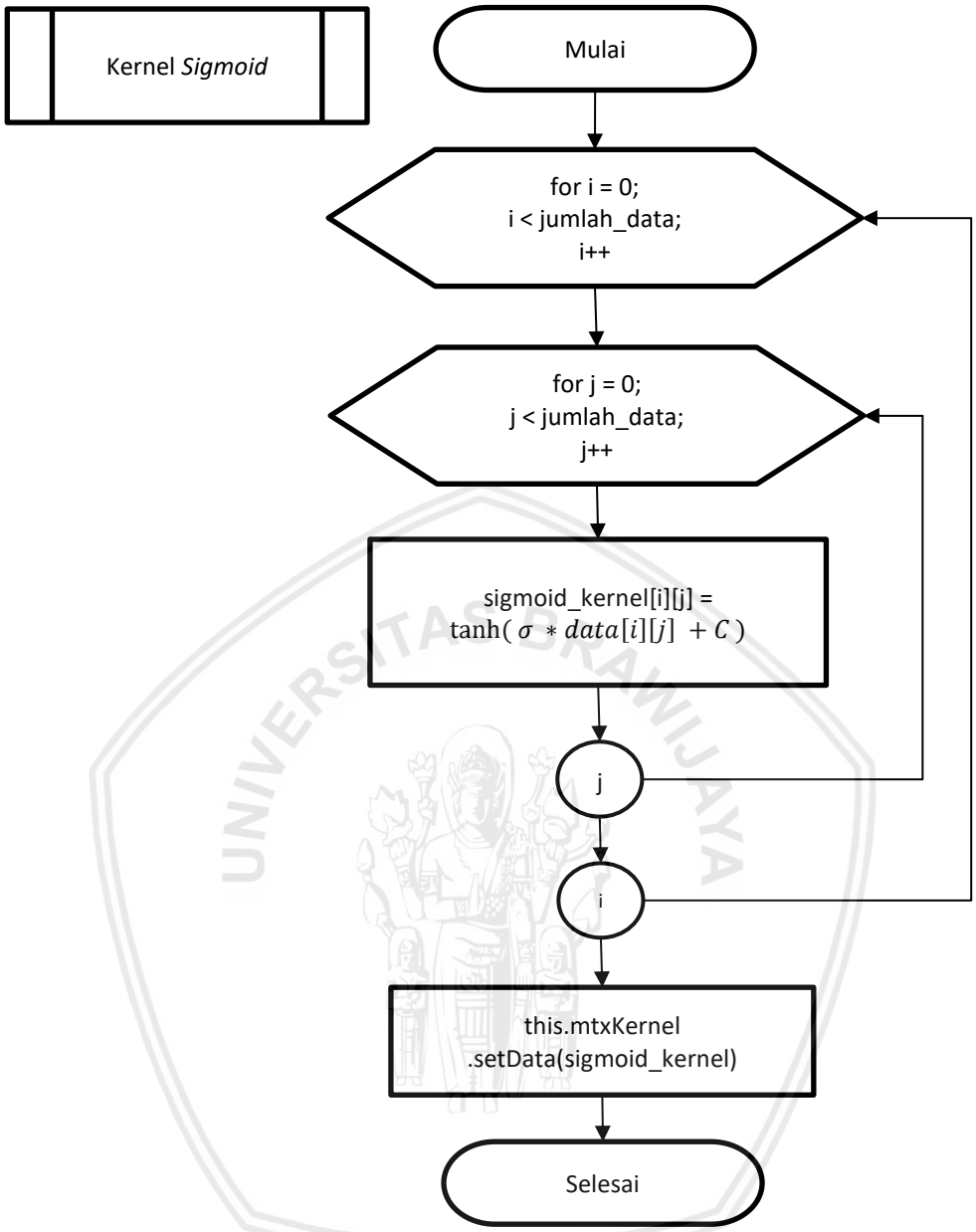
Untuk proses perhitungan *kernel polynomial* hampir sama dengan perhitungan *kernel Gaussian RBF*. Hanya saja pada proses ini, rumus perhitungan *kernel Gaussian RBF* diganti dengan rumus perhitungan *kernel polynomial*. Gambar 4.3 menjelaskan tentang diagram alir proses perhitungan *kernel polynomial*.



Gambar 4.3 Diagram alir perhitungan *kernel polynomial*

4.3.3 Proses Perhitungan *Kernel Sigmoid*

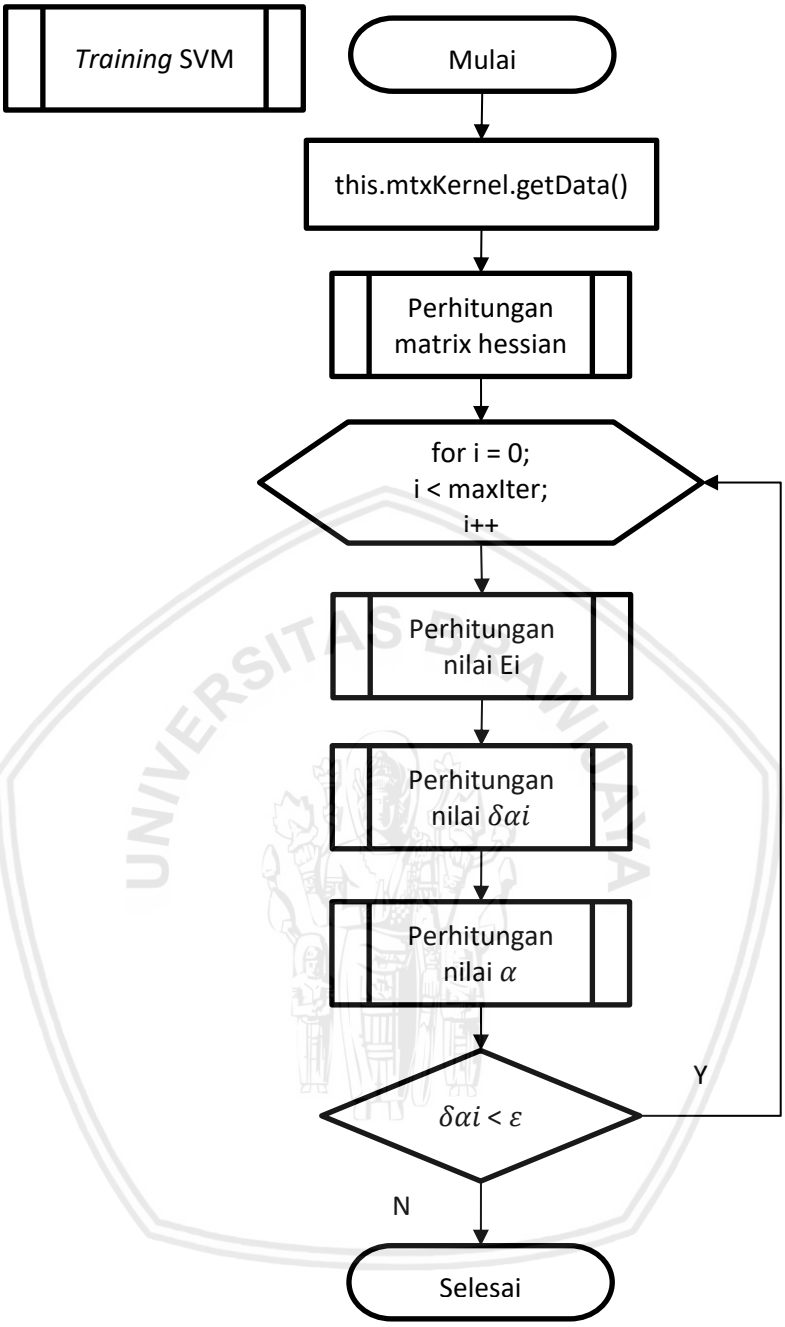
Selain *kernel Gaussian RBF* dan *kernel sigmoid*, dalam metode ini juga digunakan *kernel sigmoid*. Prosesnya algoritmenya juga hampir sama seperti proses perhitungan *kernel* sebelumnya, hanya diganti rumus proses perhitungannya dalam algoritme perulangan. Diagram alir perhitungan *kernel sigmoid* ditunjukkan pada Gambar 4.4.



Gambar 4.4 Diagram alir perhitungan *kernel sigmoid*

4.3.4 Proses Perhitungan *Training SVM*

Pada sub bab ini akan dijelaskan proses perhitungan *training*. Metode yang digunakan dalam perhitungan ini adalah *sequential training SVM,* dalam *Sequential Training SVM* terdapat proses perhitungan *matriks Hessian*. Diagram alir dari proses perhitungan *training* ditunjukkan pada Gambar 4.5.



Gambar 4.5 Diagram alir proses *sequential training SVM*

Berikut penjelasan mengenai Diagram Alir Proses Perhitungan Training SVM berdasarkan Gambar 4.5:

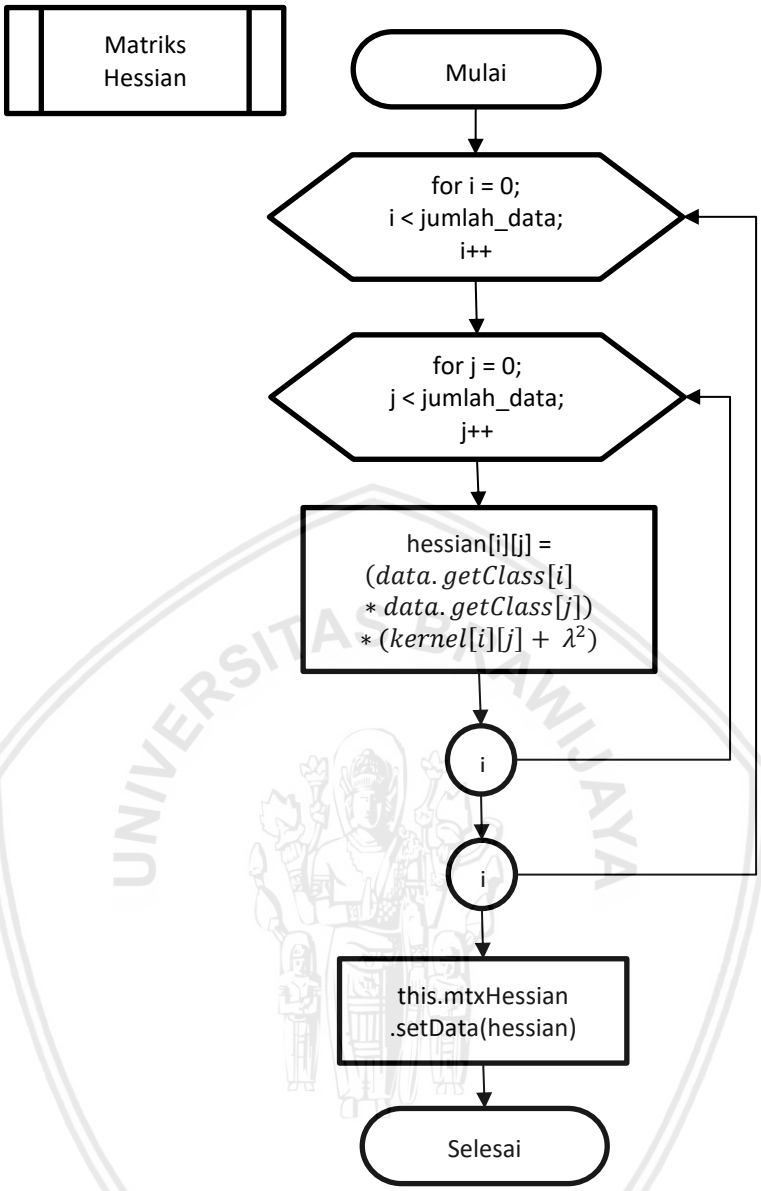
1. Memuat nilai matriks *kernel* sesuai dengan tipe kernel yang dipilih pada parameter.
2. Selanjutnya adalah menghitung matriks *Hessian* sebagai langkah pertama di dalam proses perhitungan *sequential training* SVM sesuai dengan persamaan 2.13
3. Kemudian proses selanjutnya adalah menghitung nilai *Ei* menggunakan persamaan 2.14

4. Setelah itu dilanjutkan dengan perhitungan untuk mendapatkan nilai $\delta\alpha i$ sesuai pada persamaan 2.15
5. Langkah selanjutnya yakni menghitung untuk mendapatkan nilai α sesuai dengan persamaan 2.16
6. Setelah mendapatkan nilai α dilanjutkan dengan melakukan iterasi pada data training. Iterasi pada data training akan berhenti apabila nilai sudah mencapai pada nilai konvergen ($|\delta\alpha i| < \varepsilon$) atau ketika nilai sudah mencapai nilai dari parameter iterasi (maxiter).
7. Terakhir pada proses perhitungan training SVM yaitu akan mendapatkan nilai αi.

4.3.5 Proses Perhitungan Matriks *Hessian*

Matriks *Hessian* diperlukan untuk mendapatkan nilai Ei, $\delta\alpha i$, dan α. Nilai – nilai tersebut diperlukan agar nilai *hyperplane* yang didapatkan menjadi optimal. Berikut adalah penjelasan mengenai diagram alir proses perhitungan matriks *Hessian* yang ditunjukkan pada Gambar 4.6 :

1. Dilakukan proses perulangan bertingkat 2 menggunakan indeks *i* dan *j* sebanyak jumlah data pada masing-masing perulangan.
2. Kemudian proses selanjutnya adalah proses menghitung matriks *Hessian* seperti pada persamaan 2.13. Hasil dari proses perhitungan pada variabel *Hessian* pada indeks baris *i* dan indeks kolom *j*.
3. Dari hasil perhitungan, variabel *Hessian* disimpan kedalam variabel mtx*Hessian* yang merupakan atribut objek proses *training*. Proses ini bertujuan agar nilai matriks *Hessian* dapat diakses pada semua *method* dalam objek tersebut tanpa memuat nilai variabel melalui argumen.

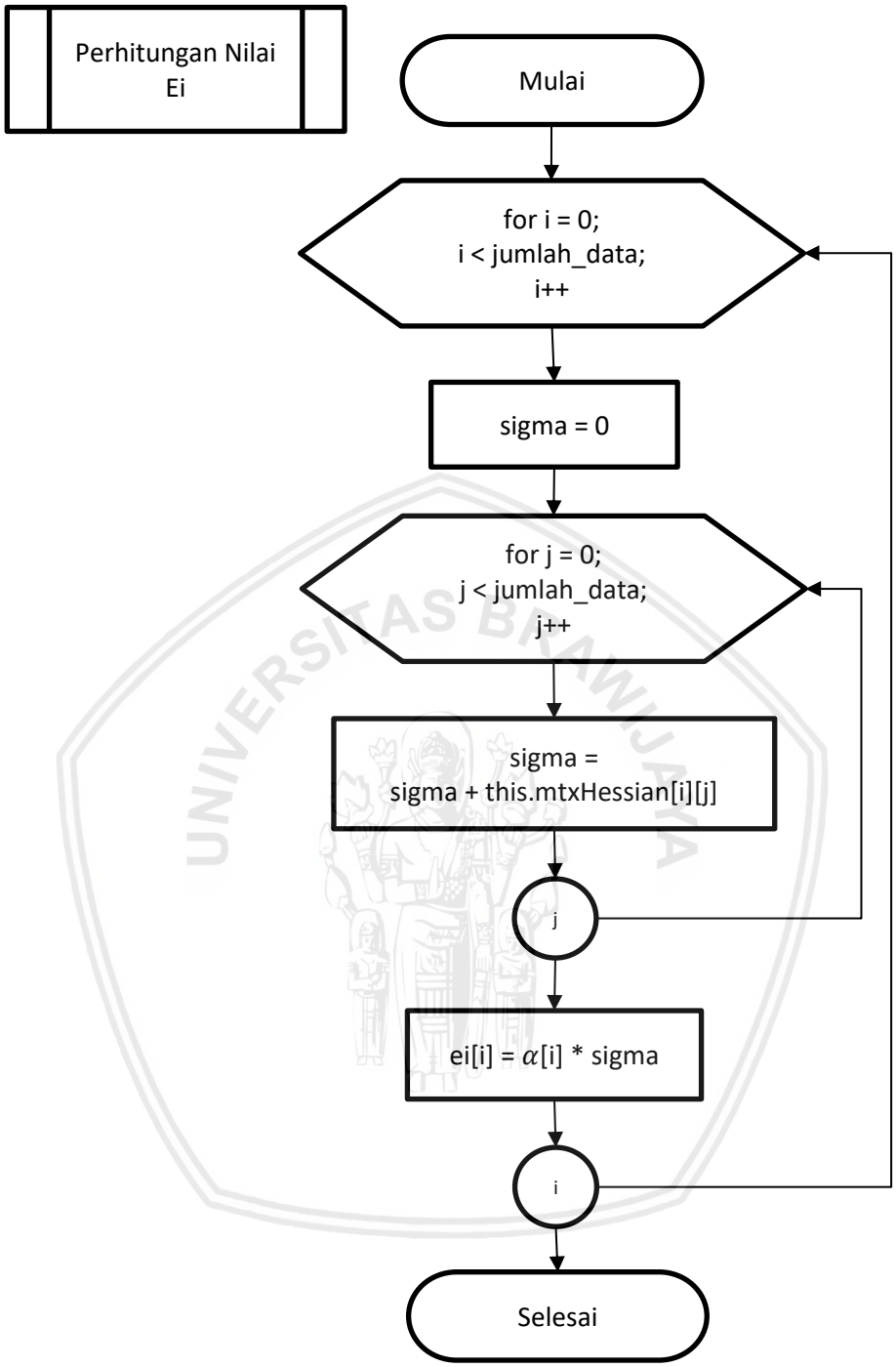


Gambar 4.6 Diagram Alir Proses Perhitungan Matriks *Hessian*

4.3.6 Proses Perhitungan Nilai *Ei*

Berikut penjelasan mengenai Diagram Alir Proses Perhitungan Nilai *Ei* yang ditunjukkan pada Gambar 4.7 :

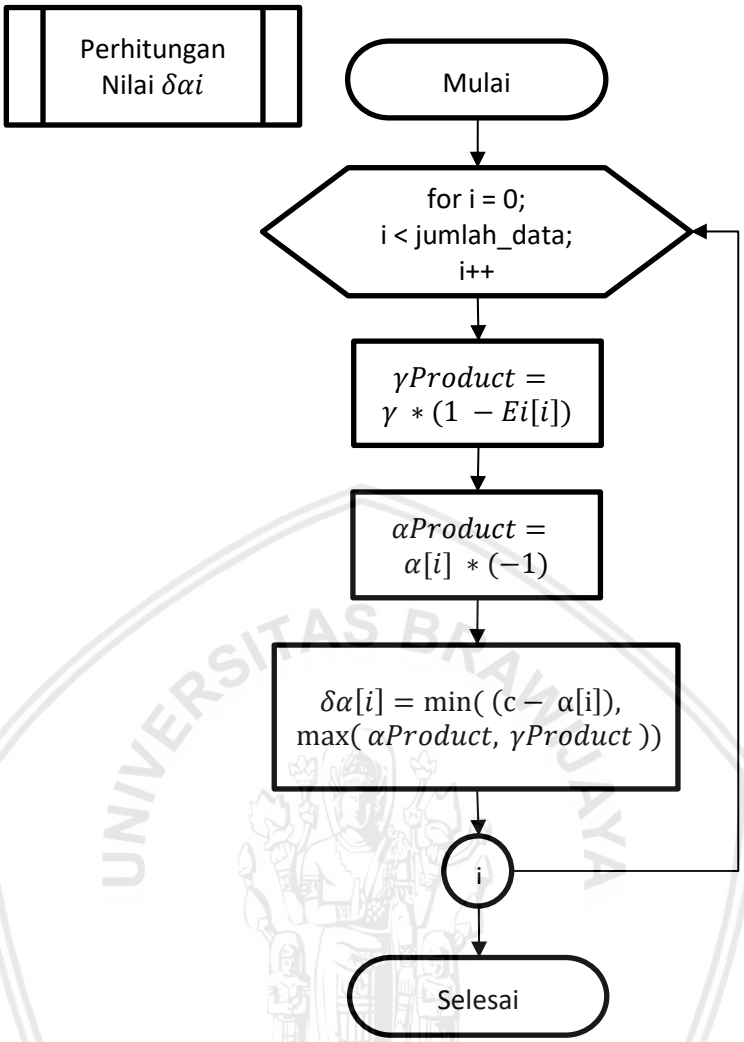
1. Pertama memasukan nilai matriks *Hessian* dan melakukan inisialisasi αi = 0
2. Selanjutnya menghitung nilai *Ei* dengan menggunakan persamaan 2.14
3. Kemudian perulangan pada sejumlah data training dan terjadi iterasi pada proses perhitungan nilai *Ei.*
4. Terakhir mendapatkan keluaran nilai *Ei.*



Gambar 4.7 Diagram alir proses perhitungan nilai *Ei*

4.3.7 Proses Perhitungan Nilai $\delta\alpha i$

Langkah selanjutnya adalah proses perhitungan nilai $\delta\alpha i$. Diagram alir untuk langkah perhitungan Nilai $\delta\alpha i$ akan ditunjukkan pada Gambar 4.8.



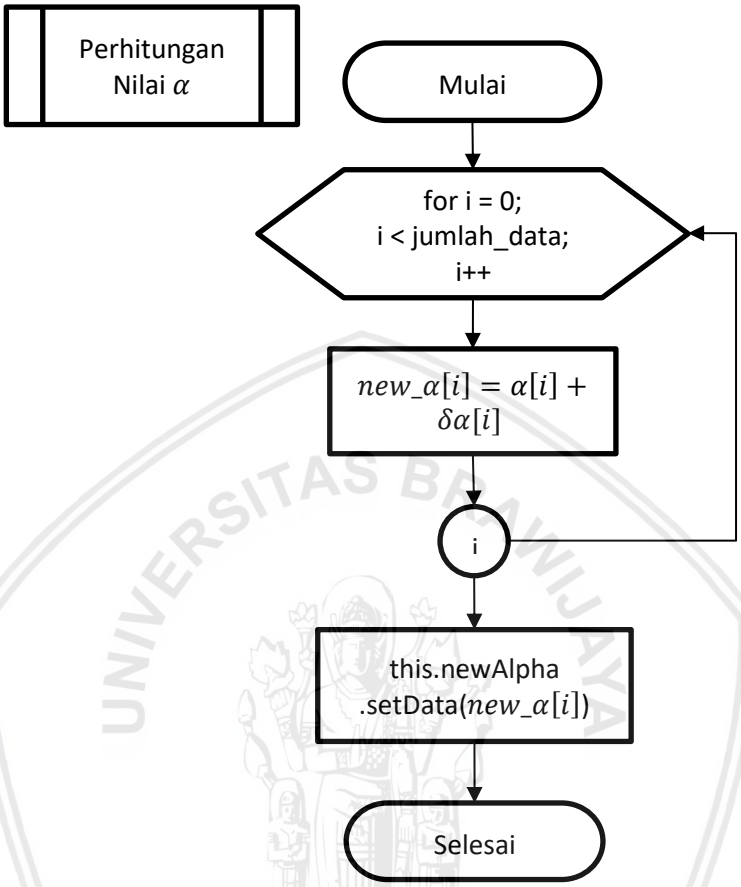
Gambar 4.8 Diagram alir perhitungan nilai $\delta\alpha i$

Berikut penjelasan mengenai Diagram Alir Perhitungan Nilai $\delta\alpha i$ berdasarkan Gambar 4.8:

1. Memasukan nilai *Ei* dan melakukan inisialisasi $C, \alpha i, y$
2. Kemudian menghitung nilai $\delta\alpha i$ dengan menggunakan persamaan 2.15
3. Setelah itu terjadi perulangan pada sejumlah data training didalam proses perhitungan untuk mendapatkan nilai $\delta\alpha i$.
4. Mendapatkan output yaitu nilai $\delta\alpha i$.

4.3.8 Proses Perhitungan Nilai α

Langkah pada perhitungan untuk mendapakan Nilai αi akan dijabarkan pada Gambar 4.9.



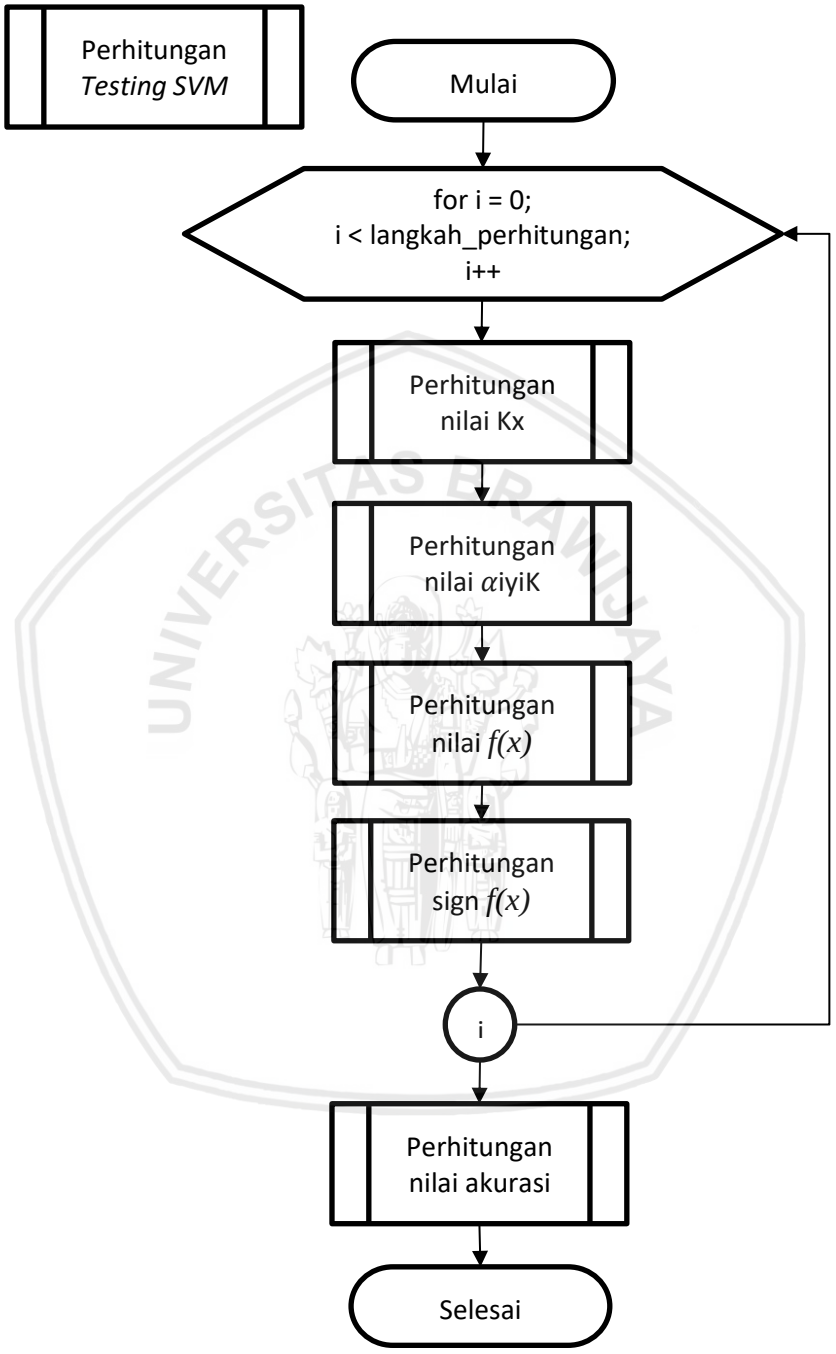
Gambar 4.9 Diagram Alir Proses Perhitungan Nilai αi

Berikut penjelasan mengenai Diagram Alir Proses Perhitungan Nilai αi berdasarkan Gambar 4.9:

1. Pertama adalah memasukan nilai $\delta\alpha i$
2. Selanjutnya menghitung nilai α seperti menggunakan persamaan 2.14
3. Kemudian terjadi perulangan pada sejumlah data training didalam proses perhitungan untuk mendapatkan nilai αi .
4. Keluaran yang didapatkan yaitu nilai α.

4.3.9 Proses Testing SVM

Proses perhitungan testing SVM akan dijelaskan dengan tahapan – tahapan pada Gambar 4.10.



Gambar 4.10 Diagram alir proses *testing SVM*

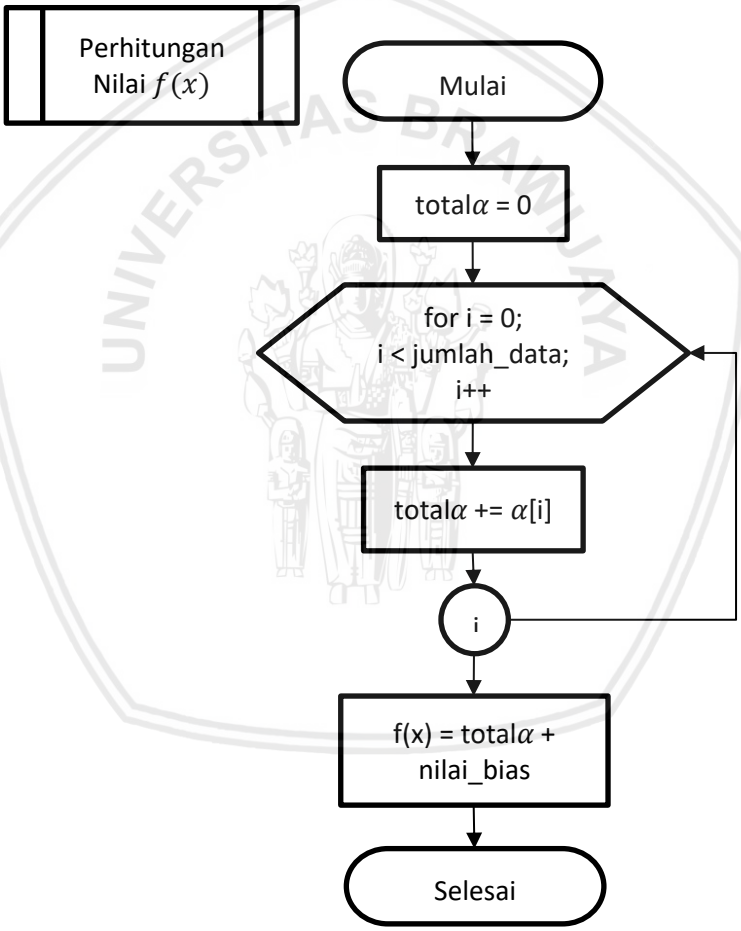
Berikut penjelasan mengenai diagram alir proses *testing SVM* berdasarkan Gambar 4.10 :

1. Melakukan proses perulangan dengan indeks *i* sebanyak langkah perhitungan. Langkah perhitungan didapatkan dari jumlah baris data testing per level perhitungan.

2. Melakukan fungsi perhitungan nilai Kx.
3. Melakukan fungsi perhitungan nilai αiyiK.
4. Melakukan fungsi perhitungan nilai *f(x)*
5. Memanggil fungsi *sign f(x)* untuk mendapatkan hasil klasifikasi pada level perhitungan saat ini.
6. Memanggil fungsi perhitungan akurasi yang didapatkan dari jumlah hasil klasifikasi yang sesuai dengan jumlah kelas penyakit sesungguhnya.

4.3.10 Proses Perhitungan Nilai *f(x)*

Proses perhitungan Nilai $f(x)$ akan dijelaskan pada diagram alir yang ditunjukkan pada Gambar 4.11.



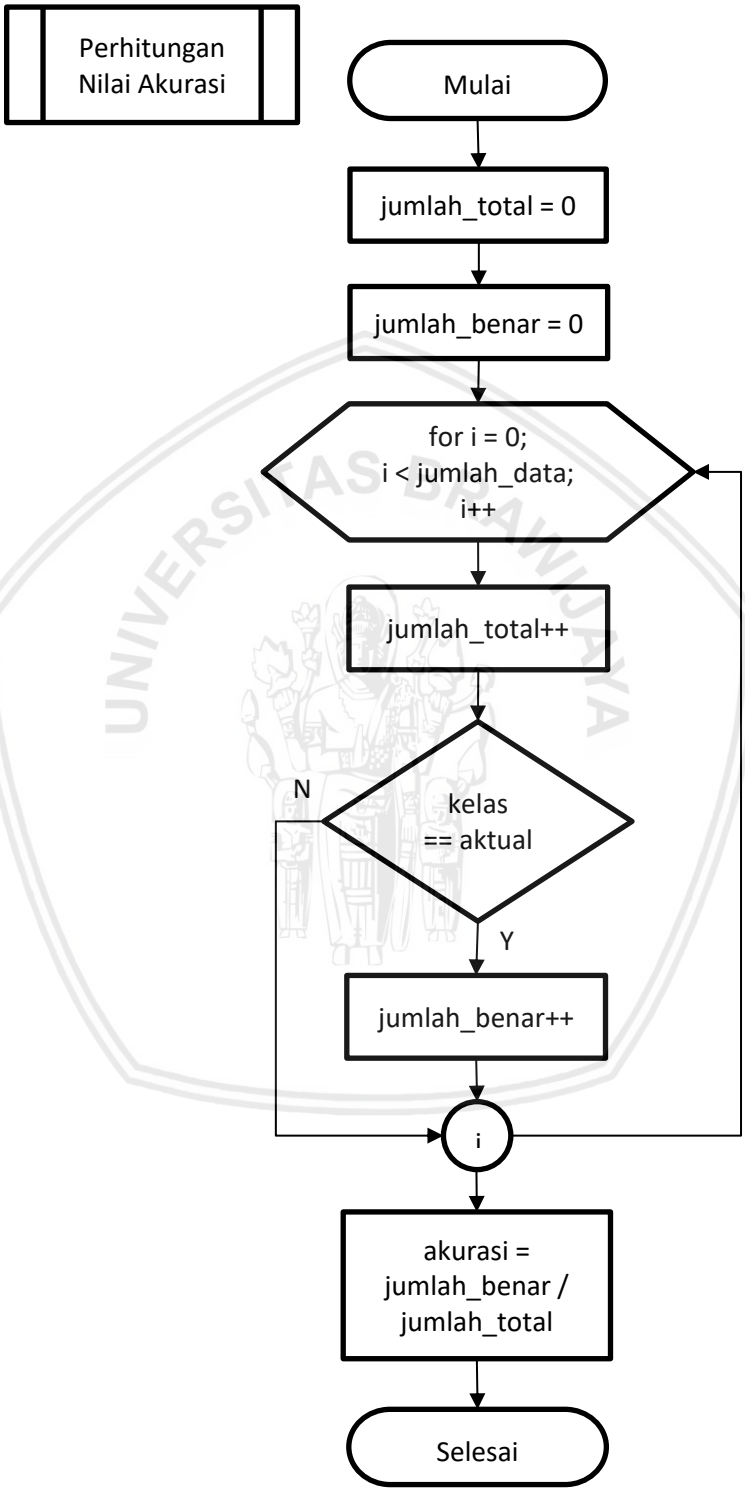
Gambar 4.11 Diagram alir proses perhitungan nilai *f(x)*

Berikut penjelasan mengenai diagram alir proses perhitungan nilai *f(x)* berdasarkan Gambar 4.11.

1. Inisiasi nilai totalα dengan nilai nol
2. Melakukan perulangan dengan indeks *i* sebanyak jumlah baris data
3. Menambahkan nilai α ke-i ke variabel totalα
4. Menghitung nilai *f(x)* dari hasil penjumlahan nilai totalα dengan nilai bias

4.3.11 Perhitungan Nilai Akurasi

Proses perhitungan nilai akurasi akan digambarkan dengan diagram alir yang ditunjukkan pada Gambar 4.12.



Gambar 4.12 Diagram alir perhitungan nilai akurasi

Berikut penjelasan mengenai Diagram Alir Perhitungan Nilai b berdasarkan Gambar 4.12:

1. Menginisiasi nilai jumlah total dan jumlah benar dengan nilai 0
2. Melakukan perulangan dengan indeks *i* sebanyak jumlah data
3. Menambahkan 1 pada nilai jumlah total pada setiap iterasi
4. Menambahkan 1 pada nilai jumlah benar jika kelas yang didapat dari hasil klasifikasi sama dengan kelas penyakit sesungguhnya pada setiap iterasi
5. Menghitung nilai akurasi dengan cara membagi nilai jumlah benar akhir dengan jumlah total.

4.4 Perhitungan Manualisasi SVM

Pada sub bab ini pada proses manualisasi SVM dengan menggunakan sequential training cara penerapannya dengan mengklasifikasikan suatu data menjadi beberapa kelas. Hasil klasifikasi digambarkan pada grafik yang dipisahkan oleh garis yang disebut dengan *hyperplane*. *Hyperlane* ini merupakan fungsi untuk mendapatkan nilai margin yang maksimal. Proses perhitungngan manualisasi *Support Vector Machine* terdapat beberapa tahap. Terdapat 4 proses tahap, yang pertama proses perhitungan adalah perhitungan normalisasi, kedua yakni proses perhitungan kernel *Support Vector Machine,* ketiga yakni proses perhitungan *sequential training,* dan keempat yakni proses perhitungan testing.

Dataset Support Vector Machine pada penelitian ini dipilih secara acak yang terdiri dari 15 dataset training dan 5 dataset testing. Masing – masing dataset training dan dataset testing memiliki 5 kelas, diantaranya kelas pertama yakni kelas scabies, kelas cat flea, kelas abses, kelas dermatitis, dan terakhir kelas jamur. Karena pada peneltian ini terdapat lima kelas oleh karna itu digunakan strategi one-against-all, dimana kelas pertama pada level 1 kelas scabies ditunjukan dengan nilai 1 dan kelas lainnya memilinilai -1. Tahap pertama pada proses manualisasi dataset yang dimasukan bernilai tinggi sehingga perlu dinormalisasi agar mendapatkan nilai *hyperlyne* terbaik. Berikut dataset sebelum dinormalisasi terdapat pada Tabel 4.2.

Tabel 4.2 Dataset *training* gejala penyakit kulit kucing sebelum dinormalisasi

No	GEJALA																KELAS
	G1	G2	G3	G4	G5	G6	G7	G8	G9	G10	G11	G12	G13	G14	G15	G14	
1	9	0	8	8	8	5	0	0	8	0	0	0	0	0	9	0	K1
2	8	0	8	8	5	5	0	0	7	0	0	0	0	0	8	0	K1
3	8	0	8	8	5	5	0	0	8	0	5	0	0	0	8	0	K1
4	0	8	5	0	5	0	0	0	8	0	0	0	0	8	0	8	K2
5	0	8	5	0	5	5	0	0	8	0	0	0	0	8	0	8	K2
6	0	8	5	0	5	5	0	0	8	0	0	0	0	8	0	8	K2
7	0	0	0	0	0	0	0	0	0	0	0	8	0	0	0	0	K3
8	0	0	0	0	0	0	0	0	0	0	0	8	0	0	0	0	K3

No	GEJALA																KELAS
	G1	G2	G3	G4	G5	G6	G7	G8	G9	G10	G11	G12	G13	G14	G15	G14	
9	0	0	0	0	0	5	0	5	0	0	0	8	0	0	0	0	K3
10	0	0	5	0	5	8	0	0	6	0	0	0	0	0	0	0	K4
11	0	0	5	0	5	8	5	0	8	0	0	0	0	0	0	0	K4
12	0	0	5	0	5	8	5	0	8	0	0	0	0	0	0	0	K4
13	0	0	5	5	5	8	0	0	0	5	5	0	5	0	0	0	K5
14	0	0	5	5	5	8	5	0	0	5	5	0	5	0	0	0	K5
15	0	0	5	5	5	8	5	0	0	5	8	0	5	0	0	0	K5

Tabel 4.3 Data Training sesudah dinormalisasi

DATA PENYAKIT KULIT KUCING																	
No	GEJALA																KELAS
	G1	G2	G3	G4	G5	G6	G7	G8	G9	G10	G11	G12	G13	G14	G15	G14	
1	0	0	1	1	1	0,625	0	0	1	0	0	0	0	0	0	0	K1
2	1	0	1	1	0,625	0,625	0	0	0,875	0	0	0	0	0	1	0	K1
3	1	0	1	1	0,625	0,625	0	0	1	0	0,625	0	0	0	1	0	K1
4	0	1	0,625	0	0,625	0	0	0	1	0	0	0	0	1	0	1	K2
5	0	1	0,625	0	0,625	0,625	0	0	1	0	0	0	0	1	0	1	K2
6	0	1	0,625	0	0,625	0,625	0	0	1	0	0	0	0	1	0	1	K2
7	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	K3
8	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	K3
9	0	0	0	0	0	0,625	0	1	0	0	0	1	0	0	0	0	K3
10	0	0	0,625	0	0,625	1	0	0	0,75	0	0	0	0	0	0	0	K4
11	0	0	0,625	0	0,625	1	1	0	1	0	0	0	0	0	0	0	K4
12	0	0	0,625	0	0,625	1	1	0	1	0	0	0	0	0	0	0	K4
13	0	0	0,625	0,625	0,625	1	0	0	0	1	0,625	0	1	0	0	0	K5
14	0	0	0,625	0,625	0,625	1	1	0	0	1	0,625	0	1	0	0	0	K5
15	0	0	0,625	0,625	0,625	1	1	0	0	1	1	0	1	0	0	0	K5

Tabel 4.4 Dataset *training* gejala penyakit kulit kucing sebelum dinormalisasi

DATA PENYAKIT KULIT KUCING																	
No	GEJALA																KELAS
	G1	G2	G3	G4	G5	G6	G7	G8	G9	G10	G11	G12	G13	G14	G15	G14	
1	0,88	0	1	1	0,62	0,62	0	0	1	0	0	0	0	0	0,88	0	K1
2	**0**	**1**	**0,75**	**0**	**0,62**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**0**	**1**	K2
3	**0**	**0**	**0**	**0**	**0**	**0,62**	**0**	**0**	**0**	**0**	**0**	**1**	**0**	**0**	**0**	**0**	K3
4	**0**	**0**	**0,62**	**0**	**0,62**	**1**	**1**	**0**	**1**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	K4
5	**0**	**0**	**1**	**1**	**0,62**	**0,62**	**0**	**0**	**0**	**1**	**1**	**0**	**1**	**0**	**0**	**0**	K5

4.4.1 Perhitungan Manual Kernel SVM

Perhitungan manual kernel SVM terbagi dua proses dimana langkah pertama melakukan perhitungan jarak euclididen kemudian melalukan perhitungakernel RBF.

1. Langkah pertama pada proses manualisasi SVM adalah data yang sudah dinormalisasi kemudian di hitung jarak *euclidien*. Fungsi jarak euclidien ada pada Tabel 4.5.

   Contoh perhitungan manualisasi jarak euclidien *:*

   jarak $euclidien = ||xi - xj||^2$

- Perhitungan manualisasi jarak *euclidien* baris pertama dan kolom pertama (1,1) :

  $=(1-1)^2 + (0-0)^2 + (1-1)^2 + (1-1)^2 + (1-1)^2 + (0{,}625-0{,}625)^2 + (0-0)^2 + (0-0)^2 + (1-1)^2 + (0-0)^2 + (0-0)^2 + (0-0)^2 + (0-0)^2 + (0-0)^2 = 0$

- Perhitungan manualisasi jarak *euclidien* baris kedua dan kolom pertama, (2,1) :

  $= (1- 0{,}88889)^2 + (0-0)^2 + (1-1)^2 + (1-1)^2 + (1-0{,}625)^2 + (0{,}625-0{,}625)^2 + (0-0)^2 + (0-0)^2 + (1-0{,}875)^2 + (0-0)^2 + (0-0)^2 + (0-0)^2 + (0-0)^2 + (0-0)^2 = 0{,}186$

Dibawah ini proses perhitungan manual jarak *euclidean* sehingga diperoleh hasil yang ditunjukkan pada Tabel 4.5.

Tabel 4.5 Hasil Perhitungan Manualisasi Jarak Euclidean

euclidien	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	0	0,168	0,543	4,67	4,281	4,281	6,390	6,390	7	2,48	3,421	3,421	4,953	5,953	6,562
2	0,168	0	0,406	4,336	3,946	3,946	5,336	5,336	5,946	2,08	3,086	3,086	4,368	5,368	5,977
3	0,543	0,406	0	4,711	4,321	4,321	5,961	5,961	6,571	2,524	3,461	3,461	4,211	5,211	5,352
4	4,671	4,336	4,711	0	0,390	0,390	4,781	4,781	6,171	3,062	4	4	6,781	7,781	8,390
5	4,281	3,946	4,321	0,390	0	0	5,171	5,171	5,78	2,203	3,140	3,140	5,921	6,921	7,531
6	4,281	3,94	4,321	0,390	0	0	5,171	5,171	5,781	2,203	3,140	3,140	5,921	6,921	7,531
7	6,390	5,336	5,961	4,781	5,171	5,171	0	0	1,390	3,343	4,781	4,781	5,562	6,56	7,171
8	6,390	5,336	5,961	4,781	5,171	5,171	0	0	1,390	3,343	4,781	4,781	5,562	6,56	7,171
9	7	5,946	6,571	6,171	5,781	5,781	1,390	1,390	0	3,484	4,921	4,921	5,703	6,703	7,312
10	2,484	2,086	2,52	3,062	2,203	2,203	3,34	3,343	3,484	0	1,062	1,06	3,343	4,343	4,953
11	3,421	3,086	3,461	4	3,140	3,140	4,781	4,781	4,921	1,062	0	0	4,781	3,781	4,390
12	3,421	3,086	3,461	4	3,140	3,140	4,781	4,781	4,921	1,062	0	0	4,781	3,781	4,390
13	4,953	4,368	4,211	6,781	5,921	5,921	5,562	5,56	5,703	3,343	4,781	4,781	0	1	1,140
14	5,953	5,368	5,211	7,781	6,921	6,921	6,562	6,562	6,703	4,343	3,781	3,781	1	0	0,140
15	6,562	5,977	5,352	8,390	7,531	7,531	7,171	7,171	7,312	4,953	4,39	4,390	1,140	0,140	0

2. Langkah kedua yakni melakukan proses perhitungan kernel SVM, proses perhitungan kernel RBF memasukan hasil perhitngan sebelumnya yakni hasil perhitungan jarak *euclidien.* Pada proses manualisasi kernel yang digunakan adalah kernel RBF. Fungsi kernel RBF ada pada Tabel 4.6.

   Contoh perhitungan manualisasi *Kernel RBF:*

$$\text{Kernel RBF } (xi, xj) = \exp\left(-\frac{jarak\ euclidien}{2\sigma^2}\right), \sigma = 0{,}7$$

$$\text{Kernel RBF } (xi, xj) = \exp\left(-\frac{||xi, xj||^2}{2\sigma^2}\right), \sigma = 0{,}7$$

- Perhitungan manualisasi kernel RBF baris pertama dan kolom pertama (1,1) :

  exp = (-1 x ((0)/ 2 x $0{,}7^2$)) = 1

- Perhitungan manualisasi kernel RBF baris kedua dan kolom pertama, (2,1) :

  exp = (-1 x ((0,1686)/ 2 x $0{,}7^2$)) = 0,841949

Di bawah ini proses perhitungan manual jarak *euclidien* sehingga diperoleh hasil seperti Tabel 4.6:

Tabel 4.6 Hasil Perhitungan Manualisasi Kernel RBF

K(x,y)	1	2	3	4	...	11	12	13	14	15
1	1	0,841949	0,574251	0,008504	....	0,030448785	0,030449	0,006382412	0,002301	0,001235316
2	0,841949	1	0,660644	0,011968	...	0,042852194	0,042852	0,011592506	0,004178	0,00224373
3	0,574251	0,660644	1	0,008163	...	0,029227317	0,029227	0,0135963	0,004901	0,004245649
4	0,008504	0,011968	0,008163	1	...	0,016879884	0,01688	0,000988183	0,000356	0,000191263
5	0,012669	0,017829	0,01216	0,671261	...	0,040570285	0,04057	0,002375067	0,000856	0,000459694
6	0,012669	0,017829	0,01216	0,671261	...	0,040570285	0,04057	0,002375067	0,000856	0,000459694
7	0,001472	0,004314	0,00228	0,007606	...	0,007605934	0,007606	0,00342717	0,001235	0,000663329
8	0,001472	0,004314	0,00228	0,007606	...	0,007605934	0,007606	0,00342717	0,001235	0,000663329
9	0,00079	0,002316	0,001224	0,00184	...	0,006589213	0,006589	0,002969044	0,00107	0,000574658
10	0,079256	0,118886	0,076076	0,043937	...	0,338177737	0,338178	0,03297552	0,011886	0,006382412
11	0,030449	0,042852	0,029227	0,01688	...	1	1	0,007605934	0,021101	0,011330811
12	0,030449	0,042852	0,029227	0,01688	...	1	1	0,007605934	0,021101	0,011330811
13	0,006382	0,011593	0,013596	0,000988	...	0,007605934	0,007606	1	0,360448	0,31226501
14	0,002301	0,004178	0,004901	0,000356	...	0,021101348	0,021101	0,360447789	1	0,86632522
15	0,001235	0,002244	0,004246	0,000191	...	0,011330811	0,011331	0,31226501	0,866325	1

4.4.2 Perhitungan Manualisasi Training SVM

Pada sub bab ini akan menghitung proses manualisasi training SVM, proses training SVM didalamnya terdapat proses perhitungan *sequential training SVM.* Proses *sequential training* dilakukan untuk mendapatkan nilai *matriks Hessian*, nilai αi, nilai *error,* dan nilai delta alfa. Berikut langkah-langkah dalam proses perhitungan manualisasi training SVM:

1. Menentukan nilai parameter – parameter SVM, berikut inisialisasi parameter *Support Vector Machine* inisialisasi parameter – parameter SVM , diantaranya Nilai *y (gamma)* = 0,00406, λ (lambda) $= 0{,}1$, $\varepsilon\,(epsilon) = 0{,}004$, C $(Complexity) = 100$, rumus alfa awal dan nilai error awal = 0.
2. Selanjutnya proses perhitungan manualisasi *matriks Hessian* seperti pada persamaan 2.6. Rumus perhitungan manualisasi matriks *Hessian*:

   $D = {}_{ij} (K (x_i , x_j) + \lambda^2)$

- Perhitungan manualisasi pada baris pertama, kolom pertama matriks *Hessian* (1,1):

  $D_{(1,1)} = 1.1\ (1 + (0,1)^2)$

  $= 1\ (1 + (0,1)) = 1,01$

- Perhitungan manualisasi baris kedua, kolom pertama matriks *Hessian* (2,1):

  $D_{(2,1)} = 1.1\ (0,84195 + (0,1)^2)$

  $= 1\ (0,84195 + (0,1)) = 0,85195$

Proses perhitungan manualisasi matriks *Hessian* dapat dilihat pada Tabel 4.7

Tabel 4.7 Proses Perhitungan Matriks *Hessian*

Dij	1	2	3	4	...	11	12	13	14	15
1	1,01	0,851949	0,584251	-0,0185	....	-0,040448785	-0,04045	-0,016382412	-0,0123	-0,011235316
2	0,851949	1,01	0,670644	-0,02197	...	-0,052852194	-0,05285	-0,021592506	-0,01418	-0,01224373
3	0,584251	0,670644	1,01	-0,01816	...	-0,039227317	-0,03923	-0,0235963	-0,0149	-0,014245649
4	-0,0185	-0,02197	-0,01816	1,01	...	0,026879884	0,02688	0,010988183	0,010356	0,010191263
5	-0,02267	-0,02783	-0,02216	0,681261	...	0,050570285	0,05057	0,012375067	0,010856	0,010459694
6	-0,02267	-0,02783	-0,02216	0,681261	...	0,050570285	0,05057	0,012375067	0,010856	0,010459694
7	-0,01147	-0,01431	-0,01228	0,017606	...	0,017605934	0,017606	0,01342717	0,011235	0,010663329
8	-0,01147	-0,01431	-0,01228	0,017606	...	0,017605934	0,017606	0,01342717	0,011235	0,010663329
9	-0,01079	-0,01232	-0,01122	0,01184	...	0,016589213	0,016589	0,012969044	0,01107	0,010574658
10	-0,08926	-0,12889	-0,08608	0,053937	...	0,348177737	0,348178	0,04297552	0,021886	0,016382412
11	-0,04045	-0,05285	-0,03923	0,02688	...	1,01	1,01	0,017605934	0,031101	0,021330811

Dij	1	2	3	4	...	11	12	13	14	15
12	-0,04045	-0,05285	-0,03923	0,02688	...	1,01	1,01	0,017605934	0,031101	0,021330811
13	-0,01638	-0,02159	-0,0236	0,010988	...	0,017605934	0,017606	1,01	0,370448	0,32226501
14	-0,0123	-0,01418	-0,0149	0,010356	...	0,031101348	0,031101	0,370447789	1,01	0,87632522
15	-0,01124	-0,01224	-0,01425	0,010191	...	0,021330811	0,021331	0,32226501	0,876325	1,01

3. Selanjutnya setelah melakukan proses manualisasi matriks *Hessian* dilanjutkan dengan menghitung *Ei*. Pada proses perhitungan untuk mendapatkan nilai *Ei* nilai *y (gamma)* ditentukan sebesar 0,00406 , proses iterasi ditentetukan sebesar 2, dan nilai C (Complexity) bernilai 100.

Iterasi 1:

- Menghitung nilai *Ei*. Perhitungan *Ei* dapat dilihat pada persamaan 2.7.

  Berikut contoh perhitungan *Ei:*

  $$Ei = \sum_{i=1}^{n} \alpha\, i\, Dij$$

  Nilai α awal selalu bernilai 0.

- Perhitungan manualisasi nilai *Ei* baris pertama:

  = (0 x 1,01) +(0 x 0,85195) + (0 x 0,58425) + (0 x -0,0185) + (0 x -0,0227) + (0 x-0,0227) + (0 x -0,0115) + (0 x -0,0115) + (0 x -0,0108) + (0 x -0,0893) + (0 x -0,0400448785) + (0 x -0,0404) + (0 x -0,016382412) + (0 x -0,0123) + (0 x -0,01123532) = 0

- Perhitungan manualisasi nilai *Ei* baris kedua:

  = (0 x 0,85195) +(0 x 1,01) + (0 x 0,67064) + (0 x -0,022) + (0 x -0,0278) + (0 x-0,0278) + (0 x -0,0143) + (0 x -0,0143) + (0 x -0,0123) + (0 x -0,01289) + (0 x -0,0529) + (0 x -0,0529) + (0 x -0,0216) + (0 x -0,0142) + (0 x -0,0112s) = 0

- Setelah menghitung nilai *Ei* dilanjutkan dengan perhitungan nilai $\delta\alpha_i$. Fungsi untuk mendapatkan nilai $\delta\alpha_i$ ada pada persamaan 2.8. Berikut proses perhitungan manualisasi $\delta\alpha_i$:

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

  $\delta\alpha_i$ = min {max[0,00406 (1-0), -0], 100 – 0}

  = min {max[0,00406 (1), -0], 1}

  = min {max[0,00406, 1), -0], 1}

= min (max[0,00406, 1}

= min {0,00406, 1} = 0,00406

Proses perhitungan manualisasi *sequential training* ditunjukan pada Tabel 4.8

Tabel 4.8 Perhitungan *Sequential Training*

	Iterasi 0		
no	Alpha	Error	δalpha
1	0	0	0,00406
2	0	0	0,00406
3	0	0	0,00406
4	0	0	0,00406
5	0	0	0,00406
6	0	0	0,00406
7	0	0	0,00406
8	0	0	0,00406
9	0	0	0,00406
10	0	0	0,00406
11	0	0	0,00406
12	0	0	0,00406
13	0	0	0,00406
14	0	0	0,00406
15	0	0	0,00406

4. Setelah menghitung *Ei* dan $\delta\alpha_i$, selanjutnya memperbarui nilai α_i. Fungsi memperbarui nilai α_i dapat dilihat pada fungsi 2.9. Berikut proses manualisasi memperbarui nilai α_i:

   $\alpha_i = \alpha_i + \delta\alpha_i$

   Memperbarui nilai α_i :

   $\alpha_i = 0 + 0{,}00406 = 0{,}00406$

   Berikut hasil pembaharuan nilai α_i dapat dilihat pada Tabel 4.9.

Tabel 4.9 Hasil Pembaharuan Nilai αi

1	2	3	4	....	12	13	14	15
0,00406	0,00406	0,00406	0,00406	....	0,00406	0,00406	0,00406	0,00406

5. Selanjutnya melakukan perhitungan iterasi ke 2. Pada proses iterasi langkah-langkah untuk mengihitung iterasi ke 2 sama seperti pada perhitungan iterasi pertama. Berikut hasil perhitungan *Ei* dan $\delta\alpha_i$. Berikut hasil perhitungan iterasi ke 2 pada Tabel 4.10.

Tabel 4.10 Perhitungan Iterasi ke-1

No	Iterasi 1		
	Alpha	Error	δalpha
1	0,00406	0,008683	0,004025
2	0,00406	0,008654	0,004025
3	0,00406	0,007914	0,004028
4	0,00406	0,010151	0,004019
5	0,00406	0,011863	0,004012
6	0,00406	0,011863	0,004012
7	0,00406	0,009725	0,004021
8	0,00406	0,009725	0,004021
9	0,00406	0,00659	0,004033
10	0,00406	0,007686	0,004029
11	0,00406	0,010091	0,004019
12	0,00406	0,010091	0,004019
13	0,00406	0,007287	0,00403
14	0,00406	0,009602	0,004021
15	0,00406	0,009309	0,004022

6. Langkah selanjunya adalah memperbarui nilai α_i akhir sama seperti fungsi persamaan 2.9. Berikut proses perhitungan dapat dilihat pada Tabel 4.11.

Tabel 4.11 Perhitungan Nilai αi Akhir

1	2	3	....	12	13	14	15
0,012074553	0,012074904	0,012083857	....	0,012057503	0,012091457	0,012063419	0,012066965

4.4.3 Proses Perhitungan Manualisasi Nilai *w* dan *b*

Pada sub bab ini dilakukan proses perhitungan manualisasi untuk mendapatkan nilai *w* dan *b* pada level 1. Sebelum menghitung bobot, dibutuhkan nilai alfa tertinggi pada kelas yang bernilai positif yang diambil dari perhitungan kernel K (x_i x^+) dan dibutuhkan nilai alfa tertinggi dari kelas negatif, nilai alfa tertinggi pada kelas negatif yang diambil dari perhitungan kernel K (x_i x^-). Diperoleh nilai alfa tertinggi pada kelas positif yaitu 0,012083857, yaitu data ke 3 maka yang dimasukan kedalan tabel K (x_i x^+) adalah data kernel ke 3. Selanjutnya pada nilai alfa tertinggi kelas negatif yaitu 0,012099905, yaitu data ke 9 maka yang dimasukan maka yang dimasukan kedalan tabel K (x_i x^-) adalah data kernel ke 9. Hasil perhitungan dapat dilihat pada Tabel 4.12

Tabel 4.12 Hasil Perhitungan Nilai K (xi x+) dan K (xi x-)

ID	$K(x_i, x^+)$	$K(x_i, x^-)$
1	0,574251	0,00079

ID	$K(x_i, x^+)$	K(x_i, x^-)
2	0,660644	0,002316
3	1	0,001224
4	0,008163	0,00184
5	0,01216	0,002742
6	0,01216	0,002742
7	0,00228	0,241955
8	0,00228	0,241955
9	0,001224	1
10	0,076076	0,028568
11	0,029227	0,006589
12	0,029227	0,006589
13	0,013596	0,002969
14	0,004901	0,00107
15	0,004246	0,000575

Setelah mendapatkan nilai masing – masing kelas, selanjutkan menghitung menghitung nilai w. Fungsi persamaan nilai w dapat dilihat pada persamaan 2.10 dan berikut contoh perhitungannya ditunjukkan pada Tabel 4.13.

w. $x^+ = (\sum_{i=1}^{i} a_i y_i \; K(x_i x_j))$. $x^+ = \sum_{i=1}^{i} a_i y_i \; K(x_i x^+)$

w. $x^- = (\sum_{i=1}^{i} a_i y_i \; K(x_i x_j))$. $x^- = \sum_{i=1}^{i} a_i y_i \; K(x_i x^-)$

Tabel 4.13 Hasil Perhitungan nilai w. x+ dan w. x-

ID	w. x^+	w. x^-
1	0,006934	9,54E-06
2	0,007977	2,8E-05
3	0,012084	1,48E-05
4	-9,8E-05	-2,2E-05
5	-0,00015	-3,3E-05
6	-0,00015	-3,3E-05
7	-2,7E-05	-0,00292
8	-2,7E-05	-0,00292
9	-1,5E-05	-0,0121
10	-0,00092	-0,00035
11	-0,00035	-7,9E-05
12	-0,00035	-7,9E-05
∑	-0,00016	-3,6E-05

Terakhir ialah proses perhitungan manualisasi b sama dengan pada persamaan 2.11. Berikut proses perhitungan nilai b:

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

$$= -\frac{1}{2}(0{,}02463 + -0{,}0185) = -0{,}003051131$$

4.4.4 Proses Perhitungan Manualisasi Evaluasi SVM

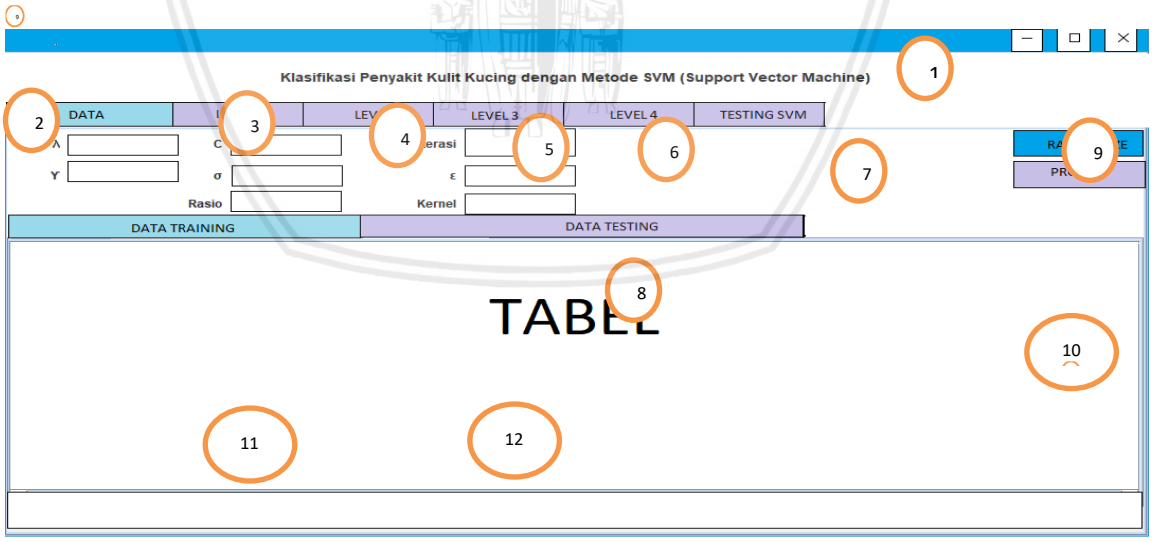
Proses perhitungan manualisasi memiliki beberapa tahap, salah satunya tahap evaluasi. Tahap evaluasi ini berfungsi untuk mengukur keberhasilan dengan membandingkan proses perhitungan manual dan dari pakar. Tahap evaluasi dimodelkan dengan membandingkan hasl penentuan dari pakar dengan hasil perhitungan manual. Evaluasi tersebut dapat dihitung dengan menggunakan perhitungan seperti berikut:

$$Akurasi = \frac{jumlah\ data\ testing\ manual}{jumlah\ seluruh\ data\ testing} \times 100\ \% = \frac{5}{5} \times 100\% = 100\ \%$$

4.5 Perancangan Antarmuka

4.5.1 Antarmuka Halaman Load Data

Antarmuka halaman utama dari sistem klasifikasi penyakit kulit kucing menggunakan metode SVM dirancang memiliki 2 tab yang digunakan untuk menampilkan data latih dan data training. Selain itu terdapat *checkbox* yang digunakan untuk menampilkan data yang baru dimuat dan data yang telah dinormalisasi. Gambar 4.13 menunjukkan rancangan antarmuka sistem.



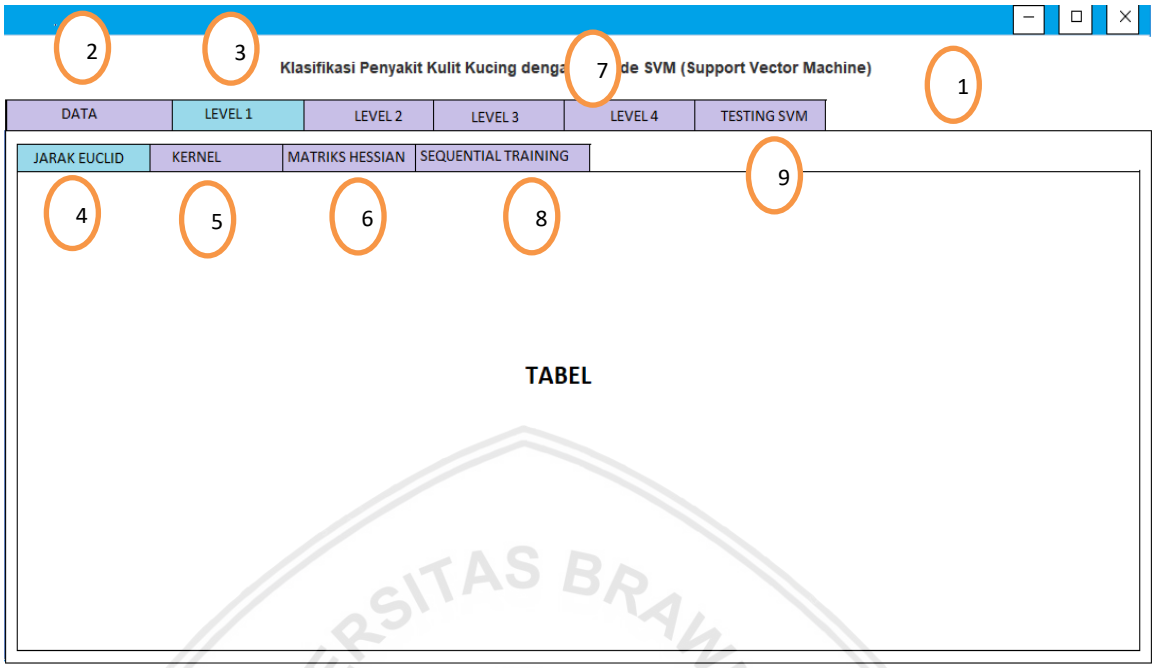
Gambar 4.13 Perancangan Antarmuka Halaman Load Data Klasifikasi Penyakit Kulit Kucing Menggunakan Metode *Support Vector Machine*

Penjelasan tiap panel pada halaman *load* data:

1. Menampilkan judul sistem yaitu Klasifikasi Kulit Kucing menggunakan metode *Support Vector Machine*

2. *Tab Data* berguna menampilkan dataset training dan dataset testing yang digunakan pada halaman data.
3. *Tab Level 1* berguna menampilkan hasil perhitungan jarak *euclidien,* kernel RBF, matriks *Hessian*, dan *sequential training Support Vector Machine* pada level 1.
4. *Tab Level* 2 berguna menampilkan hasil perhitungan jarak *euclidien,* kernel RBF, matriks *Hessian*, dan *sequential training Support Vector Machine* pada level 2.
5. *Tab Level 3* berguna menampilkan hasil perhitungan jarak *euclidien,* kernel RBF, matriks *Hessian*, dan *sequential training Support Vector Machine* pada level 3.
6. *Tab Level 4* berguna menampilkan hasil perhitungan jarak *euclidien,* kernel RBF, matriks *Hessian*, dan *sequential training Support Vector Machine* pada level 4.
7. *Tab Testing* berguna menampilkan hasil klasifikasi pada tiap level, tiap level menampilkan hasil perhitungan *f(x)*, kelas aktual dan hasil akurasi yang didapatkan.
8. *TextField Parameter* pada halaman data dimana tiap parameter dapat diinputkan sesuai keinginan user. Textfield yang terdapat pada halaman data diantaranya adalah parameter *lamda*(λ)*, epsilon, gamma (y),* iterasi*, complecxity (C),* dan *sigmoid* (*σ)*. *Combobox* rasio menampilkan pilihan perbandingan untuk perbandingan jumlah dataset pada dataset training dan dataset testing. Combobox kernel menampilkan pilihan beberapa macam kernel yang akan digunakan oleh user.
9. *Button Randomize* berfungsi untuk menampilkan data yang telah diambil secara acak oleh sistem, kemudian data yang diambil dari *file dataset* .xls ditampilkan pada tabel aplikasi.
10. *Button Proses* berfungsi untuk menampilkan hasil perhitungan ke halaman selanjutnya yaitu pada halaman level 1.
11. *Tab Data Training* berfungsi untuk menampilkan data training yang sudah diacak oleh sistem dengan ketentuan rasio perbandingan dataset training dan dan dataset testing yang ditentukan oleh user.
12. *Tab Data Testing* berfungsi untuk menampilkan data testing yang sudah diacak oleh sistem dengan ketentuan rasio perbandingan dataset training dan dan dataset testing yang ditentukan oleh user

4.5.2 Perancangan Halaman Level 1 Jarak Euclid



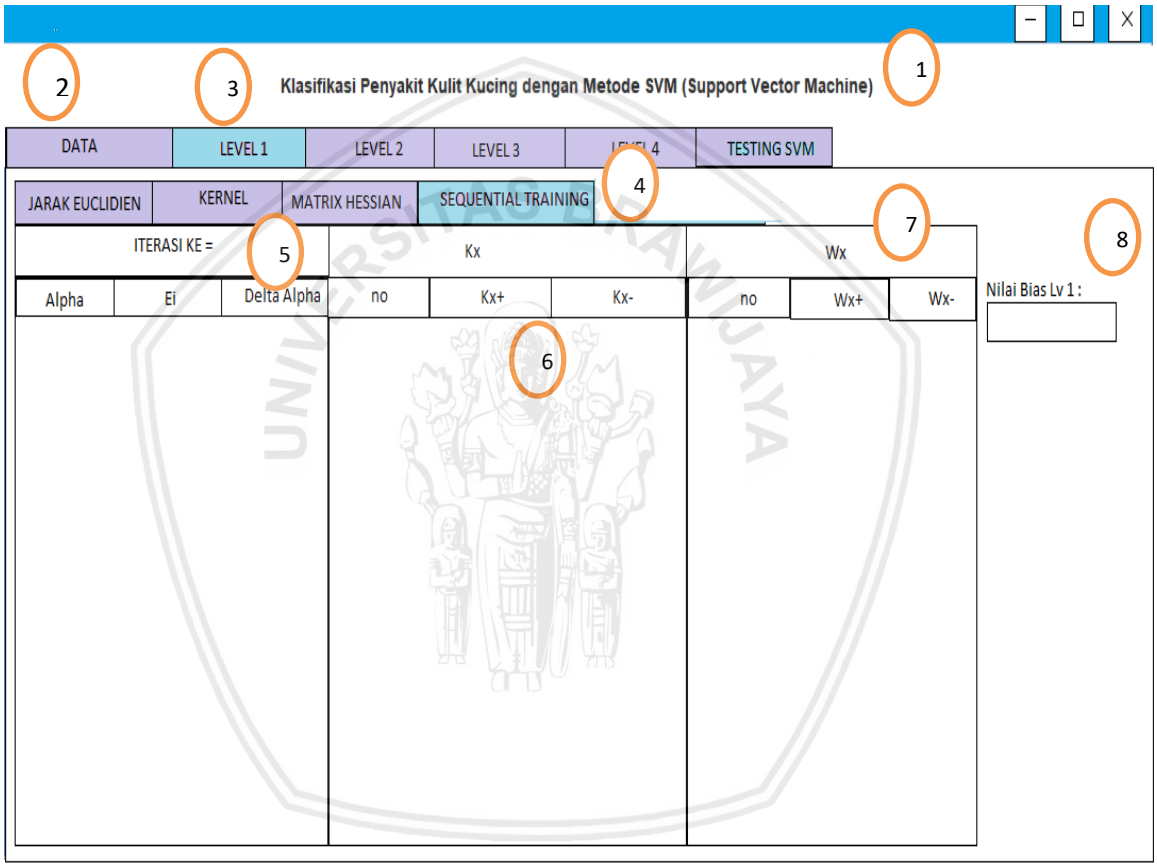
Gambar 4.14 Perancangan Halaman Jarak *Euclid*

Gambar 4.14 menunjukkan rancangan desain antarmuka halaman level 1, berikut merupakan penjelasan tiap panel pada halaman level 1 jarak *euclid*:

1. Menampilkan judul sistem yaitu Klasifikasi Kulit Kucing menggunakan metode *Support Vector Machine.*
2. Tab *Data* berguna menampilkan dataset training dan dataset testing yang digunakan pada halaman data.
3. Tab *Level 1* berguna menampilkan hasil perhitungan jarak *euclidien,* kernel RBF, matriks *Hessian*, dan *sequential training Support Vector Machine* pada level 1.
4. Tab *Jarak Euclidien* pada level 1 berguna untuk menampilkan hasil perhitungan jarak euclidien pada level 1 dimana dataset yang ditampilkan adalah dataset training yang sudah ditentukan rasio perbandingan untuk dataset training dan dataset testing.
5. Tab *Kernel* berguna pada level 1 untuk menampilkan hasil perhitungan kernel pada level 1 dimana dataset yang ditampilkan adalah dataset training yang sudah ditentukan rasio perbandingan untuk dataset training dan dataset testing.
6. Tab *Matrix Hessian* pada level 1 berguna menampilkan hasil perhitungan *matrix Hessian* pada level 1 dimana dataset yang ditampilkan adalah dataset training yang sudah ditentukan rasio perbandingan untuk dataset training dan dataset testing.

7. *Tab Sequential Training* pada level 1 menampilkan hasil perhitungan *Sequential Training.*
8. Tab *Level 2,* Tab *Level 3,* dan Tab *Level 4* berguna menampilkan berguna menampilkan hasil perhitungan jarak *euclidien,* kernel RBF, matriks *Hessian*, dan *sequential training Support Vector Machine* pada tiap level masing – masing.
9. Tab *Testing SVM* berguna menampilkan dataset hasil perhitungan testing SVM.

4.5.3 Perancangan Halaman Level 1 Sequential Training



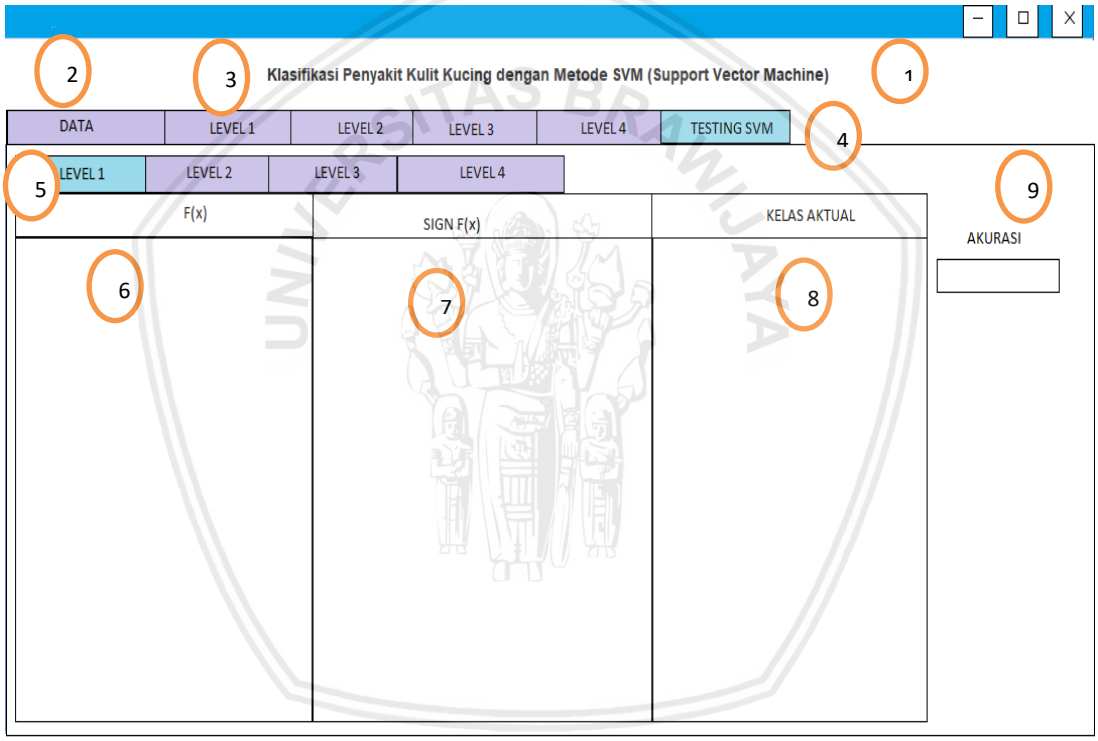
Gambar 4.15 Perancangan Halaman *Sequential Training*

Penjelasan tiap panel pada Gambar 4.15 halaman *sequential training*:

1. Menampilkan judul sistem yaitu Klasifikasi Kulit Kucing menggunakan metode *Support Vector Machine*.
2. Tab *Data* berguna menampilkan dataset training dan dataset testing yang digunakan pada halaman data.
3. Tab *Level 1* berguna menampilkan hasil perhitungan jarak *euclidien,* kernel RBF, matriks *Hessian*, dan *sequential training Support Vector Machine* pada level 1. Begitupun untuk Tab *Level 2,* Tab *Level 3* dan Tab *Level 4.*

4. *Tab Sequential Training* berguna menampilkan hasil perhitungan *sequential training,* dimana pada tab tersebut menampilkan hasil perhitungan nilai alfa, nilai eror, nilai delta alfa, nilai Kx, nilai Wx, dan nilai bias pada level 1 dengan ketentuan parameter sebelumnya yang diinputkan oleh user. Tiap level dapat menampilkan hasil perhitungan *sequential training.*
5. Combo box iterasi ke menampilkan hasil perhitungan nilai alfa, nilai eror dan nilai delta alfa sesuai dengan parameter iterasi yang diisi oleh user.
6. Menampilkan nilai Kx yang terdiri dari Kx+ dan Kx-
7. Menampilkan nilai Wx yang terdiri dari Wx+ dan Wx-
8. Menampilkan nilai bias pada level 1.

4.5.4 Perancangan Halaman Testing SVM level 1



Gambar 4.16 Perancangan Halaman Testing SVM

Berikut penjelasan dari desain antarmuka pada Gambar 4.16:

1. Menampilkan judul sistem yaitu Klasifikasi Kulit Kucing menggunakan metode *Support Vector Machine*.
2. Tab *Data* berguna menampilkan dataset training dan dataset testing yang digunakan pada halaman data.
3. Tab *Level 1* berguna menampilkan hasil perhitungan jarak *euclidien,* kernel RBF, matriks *Hessian*, dan *sequential training Support Vector Machine* pada level 1. Begitupun untuk Tab *Level 2,* Tab *Level 3* dan Tab *Level 4*.

4. Tab *Testing SVM* berfungsi menampilkan hasil perhitungan SVM yaitu hasil klasifikasi dimana data ditampilkan dengan nilai f(x), nilai sign f(x), kelas aktual, dan akurasi yang didapatkan dari parameter, rasio, dan jenis kernel yang diinputkan sebelumnya oleh user.
5. Tab *Level 1* pada testing svm menampilkan hasil perhitungan testing SVM pada level 1 yang tak lain level 1.
6. Menampilkan hasil perhitungan hasil klasifikasi yakni nilai f(x) pada tiap level.
7. Menampilkan hasil perhitungan hasil klasifikasi yakni nilai sign f(x) pada tiap level.
8. Menampilkan hasil perhitungan hasil klasifikasi yakni nilai kelas aktual pada tiap level.
9. Menampilkan nilai akurasi yang didapatkan.

4.6 Perancangan Pengujian

Pada sub bab ini, pengujian sistem pada akan dilakukan untuk mengetahui tingkat akurasi sistem dengan sebuah table data. Pada sistem klasifikasi penyakit kulit kucing menggunakan beberapa skenario pengujian diantaranya pengujian untuk mendapatkan nilai rasio data, pengujian untuk mendapatkan nilai kernel, pengujian untuk mendapat nilai parameter λ *(Lamda),* pengujian untuk mendapatkan nilai parameter C $(Complexity)$, pengujian untuk mendapatkan nilai parameter y $(Gamma)$, dan pengujian untuk mendapatkan nilai jumlah iterasi SVM. Pengujian dilakukan agar mendapatkan nilai parameter terbaik sehingga sistem dapat mengetahui hasil akurasi yang terbaik.

4.6.1 Pengujian Terhadap Nilai Akurasi Perbandingan Rasio Data Training dengan Data Testing

Pengujian untuk mendapatkan nilai akurasi rasio data dengan menggunakan data training dan data testing agar sistem dapat menghasilkan nilai akurasi yang terbaik terbaik. Pada penelitian ini sistem menggunakan 240 data, pada data training dan data testing sistem secara random memaksukan data kedalam proses perhitungan. Untuk mendapatkan perbandingan rasio data, sistem akan memasukan perbandingan rasio data training dan data testing sebesar 90%:10%, 80%:20%, 70%:30%, 60%:40%, 50%:50%, 40%:60%, 30%:70%, 20%:80%, 10%:90%. Berikut merupakan perancangannya dapat dilihat pada Tabel 4.14.

Tabel 4.14 Perancangan Hasil Pengujian

Nilai *Rasio Data*	Dataset ke-										Rata-Rata Akurasi
	1	2	3	4	5	6	7	8	9	10	
90:10											
80:20											
70:30											
60:40											
50:50											
40:60											
30:70											
20:80											
10:90											

4.6.2 Pengujian Terhadap Nilai Akurasi pada Jenis Kernel

Pengujian terhadap jenis kernel untuk mendapatkan nilai akurasi yang terbaik. Pengujian menggunakan 4 jenis kernel diantaranya kernel RBF, kernel sigmoid, kernel polynomial, dan kernel linear. Masing-masing kernel diuji dengan mendapatkan nilai akurasi yang tinggi sehingga dapat mengetahui kernel yang terbaik. Berikut perancangan pengujian terhadap nilai akuraso pada henis kernel pada

Jenis	Dataset ke-										Rata-Rata Akurasi
	1	2	3	4	5	6	7	8	9	10	
RBF											
Sigmoid											
Linear											
Polynomial											

.

Tabel 4.15 Pengujian Jenis Kernel

Jenis	Dataset ke-										Rata-Rata Akurasi
	1	2	3	4	5	6	7	8	9	10	
RBF											
Sigmoid											
Linear											
Polynomial											

4.6.3 Pengujian Terhadap Nilai Akurasi pada Parameter λ (*Lamda)*

Pengujian pada parameter λ (*Lamda*) untuk mendapatkan tingkat akurasi yang baik. Skenario penggujian dilakukan dengan memasukan nilai lamda diantaranya [0.1], [0.5], [1], [10], [50], [100], [500], [1000]. Dengan begitu diharapkan pengujian dengan beberapa nilai parameter λ (*Lamda*) dapat diketahui akurasi terbaik. Berikut perncangan pengujian terhadap akurasi pada parameter λ (*Lamda)* ditunjukan pada Tabel 4.16.

Tabel 4.16 Pengujian Parameter λ *(Lamda)*

Nilai λ (*Lamda)*	Percobaan ke-*i* (%)					Nilai Rata – Rata
	1	2	…	9	10	
0,00001						
0,0001						
0,001						
0,01						
0,1						
1						
50						
100						

4.6.4 Pengujian Terhadap Akurasi pada Parameter *C (Complexity)*

Selanjutnya pengujian yang di uji untuk melihat pengaruh akurasi dengan menggunakan parameter *C* dengan tujuan mendapatkan nilai akurasi terbaik. Skenario penggujian dilakukan dengan memasukan nilai *C (complexity)* diantaranya [0,01], [0,1], [1], [10], [30], [50], [70], [100]. Dengan begitu diharapkan pengujian dengan beberapa nilai parameter *C (complexity)* dapat diketahui akurasi

terbaik. Berikut perancangan pengujian terhadap akurasi pada parameter *C (complexity)* ditunjukan pada **Error! Reference source not found.**

Tabel 4.17 Pengujian Parameter C *(Complexity)*

Nilai *complexity*	Dataset ke-										Rata-Rata
	1	2	3	4	5	6	7	8	9	10	
0,000001											
0,00001											
0,0001											
0,001											

Nilai *complexity*	Dataset ke-										Rata-Rata
	1	2	3	4	5	6	7	8	9	10	
0,01											
0,1											
1											
10											

4.6.5 Pengujian Terhadap Akurasi pada Parameter *y (Gamma)*

Pengujian berikutnya ialah pengujian terhadap akurasi pada parameter *y (gamma)* untuk mendapatkan nilai akurasi yang tinggi. Pengujian terhadap parameter *y (gamma)* dengan memasukan nilai parameter diantaranya 0.001, 0.01, 0.1, 0.5, 1, 5, 10 dan 50. Berikut tabel perancangannya parameter y (*gamma*) pada sistem klasifikasi penyakit kulit kucing dengan menggunakan SVM ditunjukkan pada Tabel 4.18.

Tabel 4.18 Pengujian Akurasi pada Parameter y *(Gamma)*

Nilai *gamma*	Dataset ke-										Rata-Rata Akurasi
	1	2	3	4	5	6	7	8	9	10	
0,000001											
0,00001											
0,0001											
0,001											

0,01											
0,1											
1											
10											

4.6.6 Pengujian Terhadap Akurasi pada Jumlah Iterasi SVM

Pengujian yang di uji selanjutnya ialah jumalah iterasi pada SVM, pengujian jumlah iterasi dilakukan agar mengetahui pengaruh akurasi terhadap jumlah iterasi SVM. Pengujian dilakukan dengan memasukan nilai parameter jumlah iterasi diantaranya adalah 2, 5, 7, 10, 50, 100, 150, 200, 500. Berikut tabel perancangan pengujian terhadap akurasi pada jumlah iterasi SVM dapat dilihat pada Tabel 4.19.

Tabel 4.19 Pengaruh Jumlah Iterasi Terhadap Akurasi

Nilai Iterasi	Dataset ke-										Rata-rata Akurasi
	1	2	3	4	5	6	7	8	9	10	
1											
5											
10											
30											
50											
100											
500											
1000											

BAB 5 PEMBAHASAN

Pada bab pembahasan ini didalamnya menjelaskan penjabaran bagaimana mengiplementasikan sebuah sistem yang dibutuhkan dan proses bagaimana perancangan dibuat. Pada bab pembahasan ini akan dijelaskan mengenai implementasi algoritme *Support Vector Machine* dalam pengklasifikasi penyakit kulit kucing.

5.1 Spesifikasi Sistem

Pada spesifikasi sitem akan dijelaskan tentang spesifikasi sitem yang terbagi menjadi dua sub bab, diantaranya spesifikasi perangkat keras dan perangkat lunak yang bertujuan untuk membangun sistem. Sistem ini dibuat untuk mengklasifikasikan penyakit kulit kucing dengan 14 gejala dan lima kelas menggunakan algoritme *Support Vector Machine* yang telah disesuaikan dengan perancangan dan metodologi yang telah dibuat pada bab sebelumnya. Sistem dengan menggunakan algoritme *Support Vector Machine* ini dibangun dengan bahasa pemrograman *Java*.

5.1.1 Spesifikasi perangkat keras

Berikut adalah spesifikasi perangkat keras yang digunakan dalam proses bagaimana meimplementasikan sistem untuk klasifikasi penyakit kulit kucing. Spesifikasi tersebut dapat dilihat pada Tabel 5.1.

Tabel 5.1 Spesifikasi Perangkat Keras

Komponen	Spesifikasi
Prosesor	Intel(R) Core(TM) i5 HP CPU @ 1.90GHz
Memori	8 GB
Hardisk	700 GB

5.1.2 Spefikasi perangkat lunak

Spesifikasi pada lingkup perangkat lunak yang digunakan dalam penelitian ini sebagai yang ditunjukkan pada Tabel 5.2 sebagai berikut:

Tabel 5.2 Spesifikasi Perangkat Lunak

Komponen	Spesifikasi
Sistem Operasi	Windows 10, 64 bit
Bahasa Pemrograman	*Java*
Tools Pemrograman	IntelliJ IDEA

Beberapa batasan yang membatasi pengembangan dari sistem ini adalah :

1. Data masukan yang digunakan berupa data gejala penyakit kucing yang telah dihimpun sebelumnya.
2. Data keluaran sistem berupa hasil perhitungan klasifikasi penyakit kucing berdasarkan data masukan.
3. Sistem dibangun menggunakan bahasa Pemrograman Java.
4. Metode perhitungan klasifikasi yang digunakan dalam sistem ini adalah metode *Support Vector Machine.*

5.2 Implementaasi Program

Sistem diimplementasikan menggunakan perangkat keras dan perangkat lunak yang bertujuan dengan metode SVM pada klasifikasi penyakit kulit kucing yang memiliki 8 algoritme. Berikut akan dijelaskan tiap – tiap algoritmenya.

5.2.1 Implementasi Algoritme Perhitungan *Kernel Gaussian RBF*

Pada sub bab ini menjelaskan algoritme yang digunakan pada sistem yaitu 4 macam kernel pada SVM diantaranya adalah kernel RBF. Implementasi program kernel SVM dilakukan setelah implementasi perhitungan jarak data. Algoritme perhitungan kernel melibatkan indeks n x n, n sebagai jumlah data training pada kernel RBF yang berupa matriks kernel. pada *source code* Gambar 5.1.

No	Kode Program
1	`private void kernel_gaussianRBF() {`
2	`  double[][] rbfKernel = new double[rows][rows];`
3	`  for (int i = 0; i < rows; i++) {`
4	`    for (int j = 0; j < rows; j++) {`
5	`      double euclidCell = mtxEuclid[levelIdx].getData()[i][j]`
6	`        * -1;`
7	`      double divisor = 2 * Math.pow(params.getD(),2);`
8	`      rbfKernel[i][j] = Math.exp((euclidCell / divisor));`
9	`    }`
10	`  }`
11	`  this.mtxKernel[levelIdx].setData(rbfKernel);`
12	`}`

Gambar 5.1 Implementasi algoritme perhitungan *kernel gaussian RBF*

Berikut penjelasan mengenai *source code* Gambar 5.1 :

1. Baris 2 merupakan baris kode untuk inisialisasi variabel rbfKernel
2. Baris 3 – 4 menjelaskan tentang perulangan untuk mengambil nilai data yang akan dihitung
3. Baris 5 – 8 merupakan baris kode rumus perhitungan *kernel RBF*.

4. Baris 12 adalah kode untuk menyimpan hasil perhitungan kernel RBF ke variabel atribut objek bernama mtxKernel.

5.2.2 Implementasi Algoritme Perhitungan Matriks *Hessian*

Pada sub bab implementasi algoritme perhitungan matriks *Hessian* pada Gambar 5.2 didapatkan dari variabel kernel dan nilai konstanta lambda. Source code matriks *Hessian* metode SVM dapat dilihat pada Gambar 5.2.

No	Kode Program
1	`private void computeHessian() {`
2	`    double[][] Hessian = new double[rows][rows];`
3	
4	`    for (int i = 0; i < rows; i++) { // baris matriks`
5	`      for (int j = 0; j < rows; j++) { // kolom matriks`
6	`        double classSign =`
7	`          dataTraining.getClasses(levelProgress)[i] *`
8	`        dataTraining.getClasses(levelProgress)[j];`
9	`        double kernelPlusLambda =`
10	`          mtxKernel[levelIdx].getData()[i][j] +`
11	`        Math.pow(params.getLambda(), 2);`
12	`        Hessian[i][j] = classSign * kernelPlusLambda;`
13	`      }`
14	`    }`
15	
16	`    this.mtxHessian[levelIdx].setData(Hessian);`
17	`  }`

Gambar 5.2 Implementasi algoritme perhitungan matriks *Hessian*

Berikut penjelasan mengenai *source code* Gambar 5.2:

1. Baris 2 merupakan inisialisasi awal variabel *Hessian*
2. Baris 4-5 menjelaskan tentang perulangan untuk data ke – i dan ke – j sebanyak jumlah baris data.
3. Baris 6-12 merupakan rumus perhitungan matriks *Hessian.*

5.2.3 Implementasi Algoritme Perhitungan *Sequential Training SVM*

a. Implementasi Algoritme Perhitungan E_i

Gambar 5.3 merupakan source code algoritme perhitungan E_i dimana perkalian alpha ke-i dengan matriks *Hessian* lalu dijumlahkan. Proses ini adalah proses awal pada *sequential training* SVM. Kode program iterasi awal pada perhitungan E_i pada SVM dapat dilihat pada Gambar 5.3

No	Kode Program
1	`private void computeEi(int iter) {`
2	`  double[] eiValues = new double[rows];`
3	`  for (int i = 0; i < rows; i++) { // iterasi`
4	`    double sigma = 0;`
5	`    for (int j = 0; j < rows; j++) { // baris data`
6	`      sigma += mtxHessian[levelIdx].getData()[i][j];`
7	`    }`
8	`    eiValues[i]= this.iterationData[levelIdx].getPreAlpha(`
9	`    iter)[i]*sigma;`

```
10    }
11    this.iterationData[levelIdx].setEi(iter, eiValues);
12  }
```

Gambar 5.3 Implementasi algoritme perhitungan Ei

Berikut penjelasan mengenai *source code* Gambar 5.3 :

1. Baris kode 2 adalah inisialisasi variabel untuk menyimpan nilai perhitungan Ei
2. Baris 3 & 5 merupakan baris kode perulangan untuk data ke – i dan ke – j sampai panjang datanya.
3. Baris 4 untuk menginisiasi variabel sigma dengan nilai 0.
4. Baris 6 untuk menambahkan nilai matrix *Hessian* pada baris ke – i dan dan kolom ke – j kepada variabel sigma.
5. Baris 8-9 menghitung nilai Ei dengan mengalikan α lama dengan hasil akhir dari sigma.
6. Baris 11 untuk menyimpan nilai perhitungan ke atribut objek.

b. Implementasi Algoritme Perhitungan $\delta\alpha_i$

Algoritme pada *source code* Gambar 5.4 menunjukam perhitungan $\delta\alpha_i$, pada proses perhitungan ini sistem menghitung nilai gamma minimal dan gamma maksimal, menghitung jumlah *Ei,* menghitung nilai *complexity*, dan menghitung nilai alpha (ke-i*)*. Untuk proses perhitungan algoritme perhitungan $\delta\alpha$ dapat dilihat pada kode program dibawah ini.

```
No  Kode Program
1   private boolean computeDeltaAlpha(int iter) {
2     double[] deltaAlphas = new double[rows];
3     boolean epsilonMet = true;
4
5     for (int i = 0; i < rows; i++) {
6       double gammaProduct = params.getGamma() * (1d -
7         this.iterationData[levelIdx].getEi(iter)[i]);
8       double alphaProduct =
9         this.iterationData[levelIdx].getPreAlpha(iter)[i]*-1;
10      double cMinusAlpha = params.getC() -
11        this.iterationData[levelIdx].getPreAlpha(iter)[i];
12      double maxDeltaAlpha = (gammaProduct > alphaProduct) ?
13        gammaProduct : alphaProduct;
14      deltaAlphas[i] = (maxDeltaAlpha < cMinusAlpha) ?
15        maxDeltaAlpha : cMinusAlpha;
16      epsilonMet = deltaAlphas[i] < params.getEpsilon();
17    }
18    this.iterationData[levelIdx]
19      .setDeltaAlpha(iter,deltaAlphas);
20
21    return epsilonMet;
22  }
```

Gambar 5.4 Implementasi algoritme perhitungan nilai δαi

Berikut penjelasan mengenai *source code* Gambar 5.4:

1. Baris 2-3 untuk inisiasi variabel penyimpan nilai $\delta\alpha$.

2. Baris 5 merupakan perulangan dengan indeks *i* sebanyak jumlah baris data.
3. Baris 6-15 untuk menghitung nilai $\delta\alpha_i$ sesuai dengan rumus perhitungan.
4. Baris 16 untuk menentukan apakah nilai epsilon memenuhi jika nilai $\delta\alpha_i$ lebih kecil dari parameter epsilon yang dimasukkan
5. Baris 19-20 untuk menyimpan nilai $\delta\alpha_i$ ke dalam atribut objek.
6. Baris 22 untuk mengembalikan terpenuhi atau tidaknya nilai epsilon.

c. Implementasi Algoritme Perhitungan α_i.

Gambar 5.5 *source code* algoritme perhitungan α_i yaitu perhitungan yang didapatkan dari penjumlahkan nilai baru $\delta\alpha_i$ dan nilai α_i sebelumnya. Dibawah ini merupakan *source code* algoritme perhitungan nilai α_i pada metode SVM.

No	Kode Program
1	`private void computeAlpha(int iter) {`
2	`  double[] postAlphas = new double[rows];`
3	`  for (int i = 0; i < rows; i++) {`
4	`  postAlphas[i]=this.iteration`
5	`  Data[levelIdx].getPreAlpha(iter)[i]+`
6	`  this.iterationData[levelIdx].getDeltaAlpha(iter)[i];`
7	`  }`
8	`  this.iterationData[levelIdx]`
9	`  .setPostAlpha(iter,postAlphas);`
10	`  if(iter<params.getIteration()-1) {`
11	`  this.iterationData[levelIdx]`
12	`  .setPreAlpha(iter+1, postAlphas);`
13	`  }`
14	`}`

Gambar 5.5 Implementasi algoritme perhitungan αi

Berikut penjelasan mengenai *source code* Gambar 5.5 :

1. Baris 2 menjelaskan deklarasai variabel alfa baru
2. Baris 3 menjelaskan tentang perulangan untuk menghitung alfa baru sebanyak jumlah baris data perhitungan
3. Baris 4 - 7 menjelaskan tentang rumus perhitungan nilai α_i
4. Baris 8-9 untuk menyimpan nilai α_i ke dalam atribut objek.
5. Baris 10 - 12 untuk mengubah nilai alfa baru jika berada pada iterasi terakhir

5.2.4 Implementasi Algoritme Perhitungan Nilai *kernel x+, kernel x-, bias*

Untuk implementasi algoritme perhitungan nilai kernel kernel x+ , kernel x- dan bias dapat dilihat pada *source code* Gambar 5.6. Pada *source code* dapat dilihat fungsi untuk menghitung kernel x+ , kernel x- dan bias. Untuk mendapatkan nilai kernel x+ yaitu dengan mengambil nilai alpha yang maksimal pada kelas (kernel x+) dan nilai untuk mendapatkan nilai kernel x- yaitu dengan mengambil nilai alpha yang maksimal pada kelas negatif wx^- (kernel x-). Lalu nilai wx^+ dan wx^- didapatkan dari nilai alpha tertinggi yang dikalikan dengan kernel dari kelas positif

dan kelas negatif. Berikut kode program untuk proses menghitung nilai kernel x- , kernel x+ dan bias.

```
No | Kode Program
1  | private void computeKx() {
2  |   int iter = iterationCount[levelIdx]-1;
3  |   int plusIdx = 0;
4  |   int minsIdx = 0;
5  |   int delimiter = dataTraining.getRowsPerClass(levelIdx);
6  |   double tmpMinMax = 0;
7  |   for (int i = 0; i < delimiter; i++) {
8  |     double newAlpha =
9  |       iterationData[levelIdx].getPostAlpha(iter)[i];
10 |     if (i == 0) {
11 |       tmpMinMax = newAlpha;
12 |       plusIdx = i;
13 |     }
14 |     if (newAlpha > tmpMinMax) {
15 |       tmpMinMax = newAlpha;
16 |       plusIdx = i;
17 |     }
18 |   }
19 | }
```

Gambar 5.6 Implementasi algoritme perhitungan nilai Kx

Berikut penjelasan mengenai *source code* Gambar 5.6 :

1. Baris 2-6 merupakan inisiasi nilai variabel yang dibutuhkan untuk perhitungan.
2. Baris 7 merupakan perulangan dengan indeks *i* sebanyak jumlah baris per kelas aktual.
3. Baris 8-9 untuk mengganti nilai variabel lokal newAlpha menjadi alfa terbaru sesuai dengan indeks iterasi
4. Baris 10-13 untuk menentukan nilai Kx positif.
5. Baris 14-17 untuk menentukan nilai Kx negatif.

5.2.5 Implementasi Algoritme Perhitungan Nilai *f(x).*

Gambar 5.7 adalah algoritme nilai f(x) dimana program akan menampilkan hasil perhitungan nilai level 1 dan nilai f(x), perhitungan ini nilai bias akan dijumlahkan dengan hasil penjumlahan data uji. Nilai f(x) akan bernilai positif dan negatif dimana ketika bernilai positif maka termasuk kelas level 1 dan apabila bernilai negatif maka akan masuk kelas -1. namun apabila bernilai negatif akan masuk ke kelas -1 dan masuk kedalam level 2 atau level 3. Perhitungan nilai *f(x)* dilakukan pada tiap level kelas. Berikut kode program algoritme perhitungan nilai *f(x)* pada metode SVM.

```
No | Kode Program
1  | private void computeFx(int stepIdx) {
2  |     double sigmaAlphaTesting = 0;
3  |     int rows = dataTraining.getRowsCount(levelProgress);
4  |     for (int i = 0; i < rows; i++) {
```

```
5           sigmaAlphaTesting += alpha[levelIdx][stepIdx][i];
6       }
7       this.fx[levelIdx][stepIdx] = sigmaAlphaTesting +
8           training.getBias(levelProgress);
9   }
10  private void computeSign(int stepIdx) {
11      this.classification[levelIdx][stepIdx] =
12          (this.fx[levelIdx][stepIdx] < 0) ? -1 : 1;
13  }
```

Gambar 5.7 Implementasi algoritme perhitungan nilai f(x)

Berikut penjelasan mengenai *source code* Gambar 5.7 :

1. Baris 2-3 menjelaskan tentang inisialisasi variabel pada perhitungan nilai f(x).
2. Baris 4-6 menjelaskan tentang perulangan untuk menghitung nilai sigma alfa.
3. Baris 7-8 untuk menyimpan nilai f(x) kedalam atribut objek.
4. Baris 10-13 menjelaskan tentang rumus perhitungan untuk mencari nilai absolut f(x).

5.2.6 Implementasi Algoritme Nilai Akurasi.

Algoritme pada *source code* Gambar 5.8 merupakan algoritme untuk mendapatkan nilai akurasi. Nilai akurasi merepresentasikan tingkat keakuratan metode klasifikasi *SVM* dengan berbagai macam parameter yang digunakan berdasarkan nilai kelas penyakit sesungguhnya yang didapat dari data.

No	Kode Program
1	`private void computeAccuracy(){`
2	`    double totalCount = 0;`
3	`    double correctCount = 0;`
4	`    for (int i = 0; i < classification.length; i++) {`
5	`       for (int j = 0; j < classification[i].length; j++) {`
6	`            totalCount++;`
7	`            if (dataTesting.getClasses(i+1)[j] ==`
8	`              classification[i][j]) correctCount++;`
9	`       }`
10	`    }`
11	
12	`    this.accuration = correctCount/totalCount * 100;`
13	`}`

Gambar 5.8 Implementasi algoritme perhitungan nilai akurasi

Berikut penjelasan mengenai *source code* Gambar 5.8 :

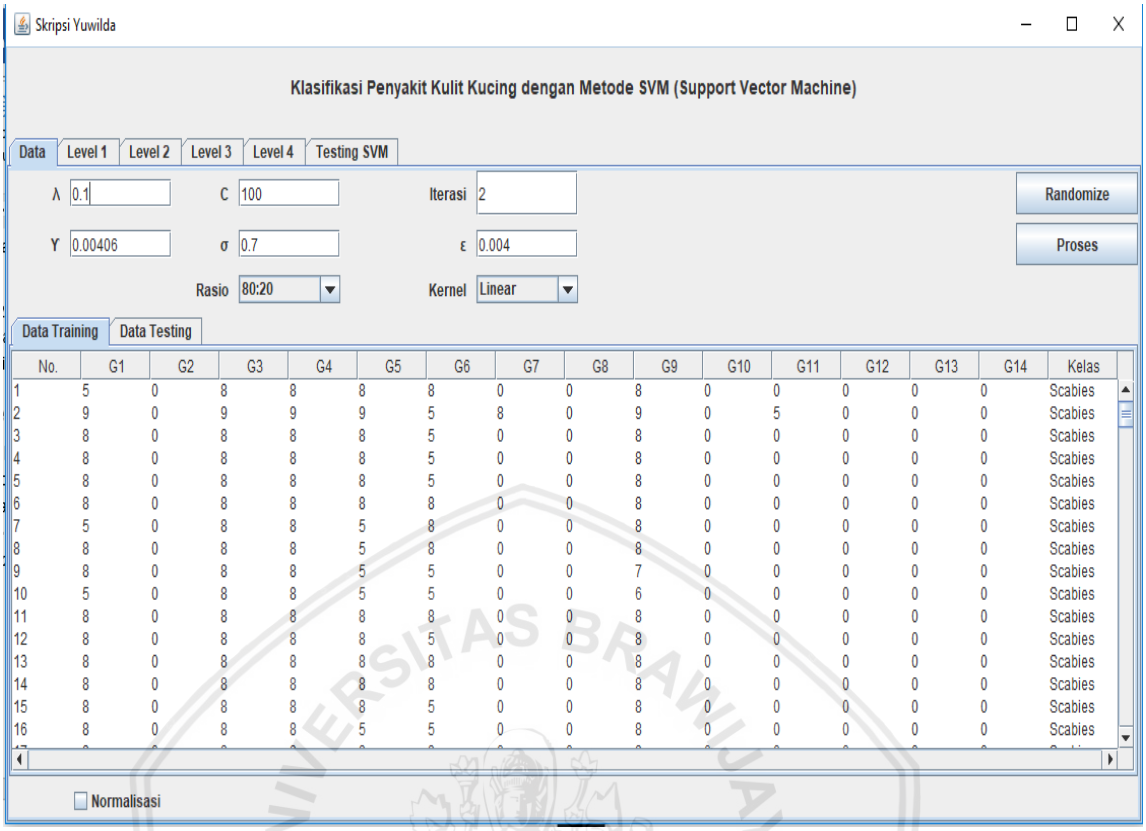
1. Baris 2-3 adalah baris kode untuk inisiasi variabel perhitungan.
2. Baris 4-10 adalah baris kode untuk perulangan penjumlahan jika nilai sign f(x) sesuai dengan nilai aktual data.
3. Baris 12 adalah baris kode untuk perhitungan presentase nilai akurasi dengan pembagian nilai jumlah akurat dengan jumlah total data.

5.3 Implementasi Antarmuka

Pengertian dari implementasi antarmuka sendiri adalah menemukan cara untuk menghubungkan *user* atau pengguna dengan sistem dengan mudah dan secara langsung. Implementasi antarmuka suatu cara untuk menghubungkan pengguna dengan sistem secara langsung. Pada bab sebelumnya perancangan telah dibuat sesuai skenari yang dirancangkan, oleh sebab itu dibuatlah sebuah sistem dengan tujuan untuk mengklasifikasikan penyakit kulit kucing dengan metode *Support Vector Machine.* Sistem yang dibuat ialah dengan menggunakan pemrograman java. Sistem ini dibuat dengan 5 halaman diantaranya halaman perhitungan level 1, halaman kedua ialah halaman perhitungan level 2, selanjutnta halaman perhitungan level 3, kemudian halaman perhitungan level 4, dan terakhir ialag halaman testing SVM. Pada halaman testing sistem akan menampilkan akurasi yang didapatkan.

5.3.1 Implementasi antarmuka Data

Dapat dilihat pada Gambar 5.9 adalah implementasi antarmuka dimana tampilan pertama yang akan dilihat user pada sistem. Pada sistem akan memperlihatkan tampilan data dengan menggunakan metode SVM. Pada halaman tesebut menampilkan data training dan data testing. Data training dan data testing ditampilkan dalam bentuk format .xls.

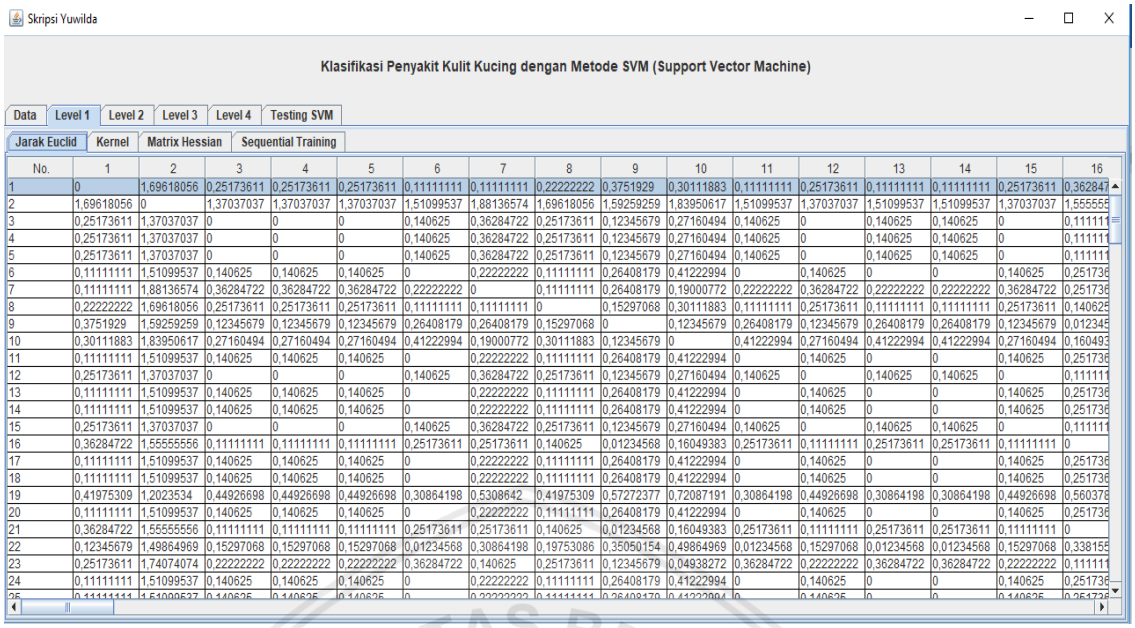


Gambar 5.9 Tampilan halaman antarmuka data

Pada Gambar 5.9 menampilan antarmuka data, terdapat tombol data training yang berfungsi menampilkan data training dan data testing untuk menampilkan data testing. Diatasnya terdapat text field parameter. Pada halaman data terdapat juga pilihan untuk memilih rasio dan kernel yang kita gunakan. Tombol randomize digunakan saat sistem menghitung data secara random dan tombol proses berfungsi untuk memproses perhitungan menggunakan metode SVM.

5.3.2 Implementasi antarmuka halaman Perhitungan Level 1

Pada implementasi antarmuka halaman perhitungan level 1 menampilkan hasil perhitungan yang dimana perhitungan menampilkan nilai dari perhitungan kernel, perhitungan matriks *Hessian*, perhitungan *sequential training* SVM, Halaman perhitungan level 1 berfungsi menampilkan semua perhitungan yang ada pada metode SVM, yakni perhitungan kernel, matriks *Hessian*, sequential training SVM. Pada halaman ini nilai parameter seperti parameter lambda, parameter gamma, parameter alfa, *complexity,* iterasi, kernel, epsilon, dan rasio data telah ditentukan. Berikut halaman perhitungan level 1 ditunjukan pada Gambar 5.10.



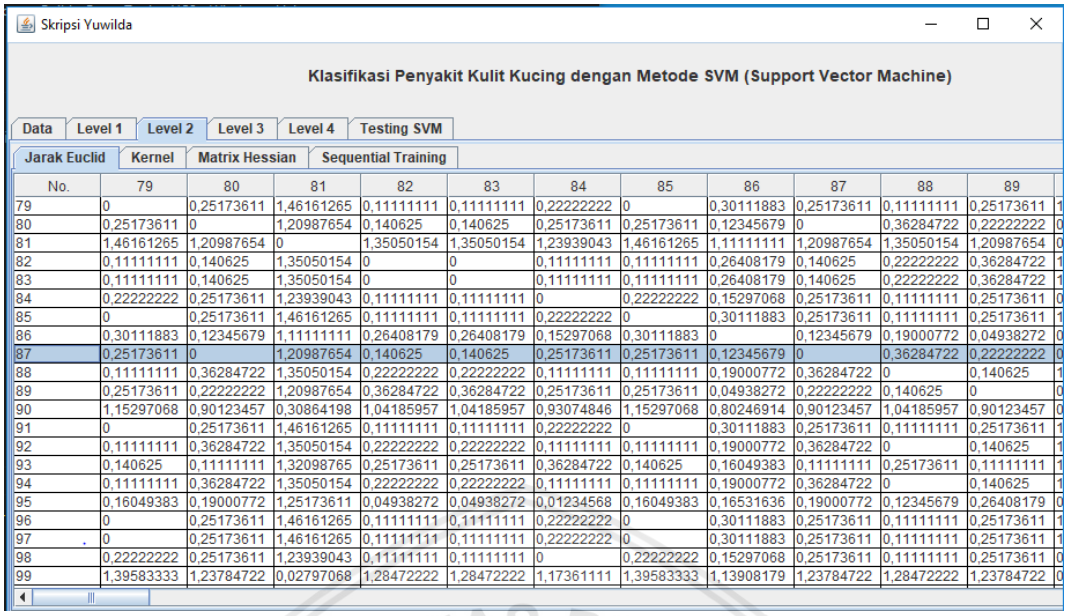
No.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	0	1,69618056	0,25173611	0,25173611	0,25173611	0,11111111	0,11111111	0,22222222	0,3751929	0,30111883	0,11111111	0,25173611	0,11111111	0,11111111	0,25173611	0,362847
2	1,69618056	0	1,37037037	1,37037037	1,37037037	1,51099537	1,88136574	1,69618056	1,59259259	1,83950617	1,51099537	1,37037037	1,51099537	1,51099537	1,37037037	1,555555
3	0,25173611	1,37037037	0	0	0	0,140625	0,36284722	0,25173611	0,12345679	0,27160494	0,140625	0	0,140625	0,140625	0	0,111111
4	0,25173611	1,37037037	0	0	0	0,140625	0,36284722	0,25173611	0,12345679	0,27160494	0,140625	0	0,140625	0,140625	0	0,111111
5	0,25173611	1,37037037	0	0	0	0,140625	0,36284722	0,25173611	0,12345679	0,27160494	0,140625	0	0,140625	0,140625	0	0,111111
6	0,11111111	1,51099537	0,140625	0,140625	0,140625	0	0,22222222	0,11111111	0,26408179	0,41222994	0	0,140625	0	0	0,140625	0,251736
7	0,11111111	1,88136574	0,36284722	0,36284722	0,36284722	0,22222222	0	0,11111111	0,26408179	0,19000772	0,22222222	0,36284722	0,22222222	0,22222222	0,36284722	0,251736
8	0,22222222	1,69618056	0,25173611	0,25173611	0,25173611	0,11111111	0,11111111	0	0,15297068	0,30111883	0,11111111	0,25173611	0,11111111	0,11111111	0,25173611	0,140625
9	0,3751929	1,59259259	0,12345679	0,12345679	0,12345679	0,26408179	0,26408179	0,15297068	0	0,12345679	0,26408179	0,12345679	0,26408179	0,26408179	0,12345679	0,012345
10	0,30111883	1,83950617	0,27160494	0,27160494	0,27160494	0,41222994	0,19000772	0,30111883	0,12345679	0	0,41222994	0,27160494	0,41222994	0,41222994	0,27160494	0,160493
11	0,11111111	1,51099537	0,140625	0,140625	0,140625	0	0,22222222	0,11111111	0,26408179	0,41222994	0	0,140625	0	0	0,140625	0,251736
12	0,25173611	1,37037037	0	0	0	0,140625	0,36284722	0,25173611	0,12345679	0,27160494	0,140625	0	0,140625	0,140625	0	0,111111
13	0,11111111	1,51099537	0,140625	0,140625	0,140625	0	0,22222222	0,11111111	0,26408179	0,41222994	0	0,140625	0	0	0,140625	0,251736
14	0,11111111	1,51099537	0,140625	0,140625	0,140625	0	0,22222222	0,11111111	0,26408179	0,41222994	0	0,140625	0	0	0,140625	0,251736
15	0,25173611	1,37037037	0	0	0	0,140625	0,36284722	0,25173611	0,12345679	0,27160494	0,140625	0	0,140625	0,140625	0	0,111111
16	0,36284722	1,55555556	0,11111111	0,11111111	0,11111111	0,25173611	0,25173611	0,140625	0,01234568	0,16049383	0,25173611	0,11111111	0,25173611	0,25173611	0,11111111	0
17	0,11111111	1,51099537	0,140625	0,140625	0,140625	0	0,22222222	0,11111111	0,26408179	0,41222994	0	0,140625	0	0	0,140625	0,251736
18	0,11111111	1,51099537	0,140625	0,140625	0,140625	0	0,22222222	0,11111111	0,26408179	0,41222994	0	0,140625	0	0	0,140625	0,251736
19	0,41975309	1,2023534	0,44926698	0,44926698	0,44926698	0,30864198	0,5308642	0,41975309	0,57272377	0,72087191	0,30864198	0,44926698	0,30864198	0,30864198	0,44926698	0,560378
20	0,11111111	1,51099537	0,140625	0,140625	0,140625	0	0,22222222	0,11111111	0,26408179	0,41222994	0	0,140625	0	0	0,140625	0,251736
21	0,36284722	1,55555556	0,11111111	0,11111111	0,11111111	0,25173611	0,25173611	0,140625	0,01234568	0,16049383	0,25173611	0,11111111	0,25173611	0,25173611	0,11111111	0
22	0,12345679	1,49864969	0,15297068	0,15297068	0,15297068	0,01234568	0,30864198	0,19753086	0,35050154	0,49864969	0,01234568	0,15297068	0,01234568	0,01234568	0,15297068	0,338155
23	0,25173611	1,74074074	0,22222222	0,22222222	0,22222222	0,36284722	0,140625	0,25173611	0,12345679	0,04938272	0,36284722	0,22222222	0,36284722	0,36284722	0,22222222	0,111111
24	0,11111111	1,51099537	0,140625	0,140625	0,140625	0	0,22222222	0,11111111	0,26408179	0,41222994	0	0,140625	0	0	0,140625	0,251736

Gambar 5.10 Tampilan halaman perhitungan level 1

Gambar 5.10 menampilkan halaman perhitungan diantaranya jarak euclidien, kernel, matrix *Hessian*, dan *sequential training*. Halaman perhitungan level 1 yang berisi sub menu seperti tombol jarak euclidien yang berfungsi menampilkan hasil perhitungan jarak. Disebelahnya terdapat tombol kernel untuk menampilkan perhitungan kernel. Tombol Matrix Hesiian berfungsi menampilkan hasil perhitungan matrix *Hessian* tombol sequential training berfungsi untuk menampilkan hasil perhitungan sequential training yang berisi nilai *Ei*, nilai *delta alfa,* nilai *alfa,* nilai kx_i, nilai wx_i, dan nilai *bias.* Proses perhitungan *sequential training* dilakukan tiap level.

5.3.3 Implementasi antarmuka halaman perhitungan Level 2

Implementasi antarmuka halaman level 2 menampilkan proses perhitungannya yang ada pada halaman sebelumnya dimana semua perhitungan yang ada pada halaman perhitungan level 1. Dimana menampilkan perhitungan kernel, matriks *Hessian*, sequential training SVM, dan perhitungan nilai f(x). Implementasi antarmuka ditunjukan dengan Gambar 5.11.

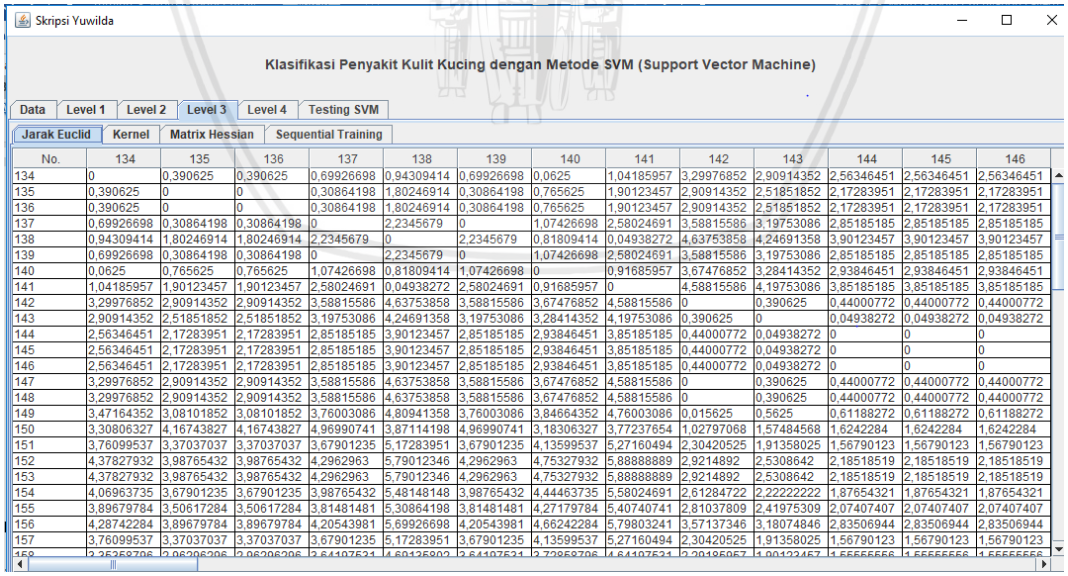


No.	79	80	81	82	83	84	85	86	87	88	89
79	0	0,25173611	1,46161265	0,11111111	0,11111111	0,22222222	0	0,30111883	0,25173611	0,11111111	0,25173611
80	0,25173611	0	1,20987654	0,140625	0,140625	0,25173611	0,25173611	0,12345679	0	0,36284722	0,22222222
81	1,46161265	1,20987654	0	1,35050154	1,35050154	1,23939043	1,46161265	1,11111111	1,20987654	1,35050154	1,20987654
82	0,11111111	0,140625	1,35050154	0	0	0,11111111	0,11111111	0,26408179	0,140625	0,22222222	0,36284722
83	0,11111111	0,140625	1,35050154	0	0	0,11111111	0,11111111	0,26408179	0,140625	0,22222222	0,36284722
84	0,22222222	0,25173611	1,23939043	0,11111111	0,11111111	0	0,22222222	0,15297068	0,25173611	0,11111111	0,25173611
85	0	0,25173611	1,46161265	0,11111111	0,11111111	0,22222222	0	0,30111883	0,25173611	0,11111111	0,25173611
86	0,30111883	0,12345679	1,11111111	0,26408179	0,26408179	0,15297068	0,30111883	0	0,12345679	0,19000772	0,04938272
87	0,25173611	0	1,20987654	0,140625	0,140625	0,25173611	0,25173611	0,12345679	0	0,36284722	0,22222222
88	0,11111111	0,36284722	1,35050154	0,22222222	0,22222222	0,11111111	0,11111111	0,19000772	0,36284722	0	0,140625
89	0,25173611	0,22222222	1,20987654	0,36284722	0,36284722	0,25173611	0,25173611	0,04938272	0,22222222	0,140625	0
90	1,15297068	0,90123457	0,30864198	1,04185957	1,04185957	0,93074846	1,15297068	0,80246914	0,90123457	1,04185957	0,90123457
91	0	0,25173611	1,46161265	0,11111111	0,11111111	0,22222222	0	0,30111883	0,25173611	0,11111111	0,25173611
92	0,11111111	0,36284722	1,35050154	0,22222222	0,22222222	0,11111111	0,11111111	0,19000772	0,36284722	0	0,140625
93	0,140625	0,11111111	1,32098765	0,25173611	0,25173611	0,36284722	0,140625	0,16049383	0,11111111	0,25173611	0,11111111
94	0,11111111	0,36284722	1,35050154	0,22222222	0,22222222	0,11111111	0,11111111	0,19000772	0,36284722	0	0,140625
95	0,16049383	0,19000772	1,25173611	0,04938272	0,04938272	0,01234568	0,16049383	0,16531636	0,19000772	0,12345679	0,26408179
96	0	0,25173611	1,46161265	0,11111111	0,11111111	0,22222222	0	0,30111883	0,25173611	0,11111111	0,25173611
97	0	0,25173611	1,46161265	0,11111111	0,11111111	0,22222222	0	0,30111883	0,25173611	0,11111111	0,25173611
98	0,22222222	0,25173611	1,23939043	0,11111111	0,11111111	0	0,22222222	0,15297068	0,25173611	0,11111111	0,25173611
99	1,39583333	1,23784722	0,02797068	1,28472222	1,28472222	1,17361111	1,39583333	1,13908179	1,23784722	1,28472222	1,23784722

Gambar 5.11 Tampilan halaman perhitungan level 2

5.3.4 Implementasi antarmuka halaman perhitungan level 3

Pada implementasi antarmuka halaman perhitungan level 3 sama seperti halaman implementasi antarmuka level 1 dan level 2, dimana halaman tersebut telah ditentukan nilai parameter – parameter sehingga sistem akan menampilkan perhitungan jarak *uclidiene*, *sequential training*, perhitungan matriks *Hessian*, dan perhitungan kernel. Berikut tampilan implementasi antarmuka perhitungan level 3 dapat dilihat pada Gambar 5.12



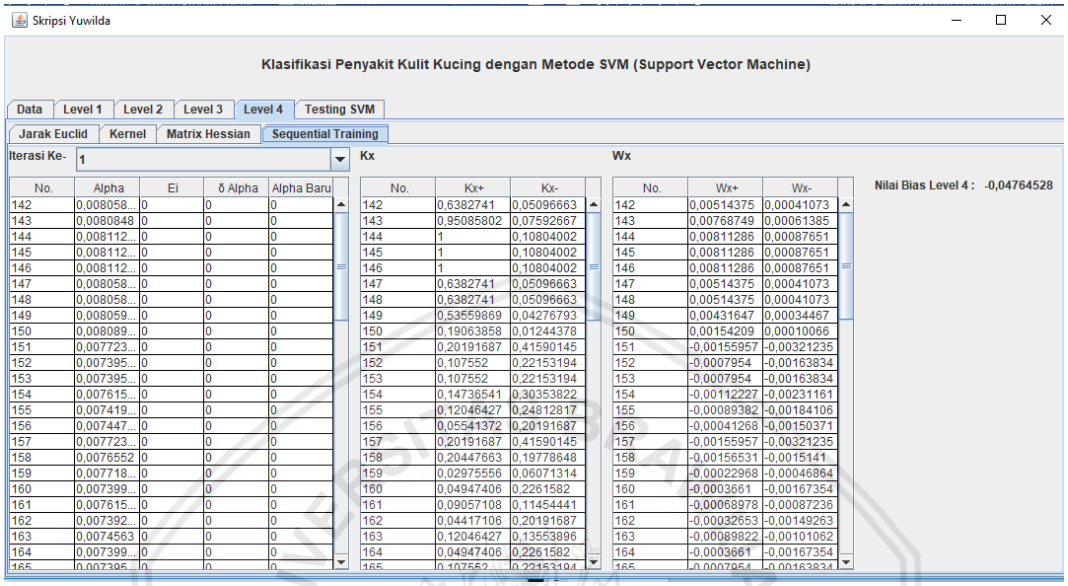
No.	134	135	136	137	138	139	140	141	142	143	144	145	146
134	0	0,390625	0,390625	0,69926698	0,94309414	0,69926698	0,0625	1,04185957	3,29976852	2,90914352	2,56346451	2,56346451	2,56346451
135	0,390625	0	0	0,30864198	1,80246914	0,30864198	0,765625	1,90123457	2,90914352	2,51851852	2,17283951	2,17283951	2,17283951
136	0,390625	0	0	0,30864198	1,80246914	0,30864198	0,765625	1,90123457	2,90914352	2,51851852	2,17283951	2,17283951	2,17283951
137	0,69926698	0,30864198	0,30864198	0	2,2345679	0	1,07426698	2,58024691	3,58815586	3,19753086	2,85185185	2,85185185	2,85185185
138	0,94309414	1,80246914	1,80246914	2,2345679	0	2,2345679	0,81809414	0,04938272	4,63753858	4,24691358	3,90123457	3,90123457	3,90123457
139	0,69926698	0,30864198	0,30864198	0	2,2345679	0	1,07426698	2,58024691	3,58815586	3,19753086	2,85185185	2,85185185	2,85185185
140	0,0625	0,765625	0,765625	1,07426698	0,81809414	1,07426698	0	0,91685957	3,67476852	3,28414352	2,93846451	2,93846451	2,93846451
141	1,04185957	1,90123457	1,90123457	2,58024691	0,04938272	2,58024691	0,91685957	0	4,58815586	4,19753086	3,85185185	3,85185185	3,85185185
142	3,29976852	2,90914352	2,90914352	3,58815586	4,63753858	3,58815586	3,67476852	4,58815586	0	0,390625	0,44000772	0,44000772	0,44000772
143	2,90914352	2,51851852	2,51851852	3,19753086	4,24691358	3,19753086	3,28414352	4,19753086	0,390625	0	0,04938272	0,04938272	0,04938272
144	2,56346451	2,17283951	2,17283951	2,85185185	3,90123457	2,85185185	2,93846451	3,85185185	0,44000772	0,04938272	0	0	0
145	2,56346451	2,17283951	2,17283951	2,85185185	3,90123457	2,85185185	2,93846451	3,85185185	0,44000772	0,04938272	0	0	0
146	2,56346451	2,17283951	2,17283951	2,85185185	3,90123457	2,85185185	2,93846451	3,85185185	0,44000772	0,04938272	0	0	0
147	3,29976852	2,90914352	2,90914352	3,58815586	4,63753858	3,58815586	3,67476852	4,58815586	0	0,390625	0,44000772	0,44000772	0,44000772
148	3,29976852	2,90914352	2,90914352	3,58815586	4,63753858	3,58815586	3,67476852	4,58815586	0	0,390625	0,44000772	0,44000772	0,44000772
149	3,47164352	3,08101852	3,08101852	3,76003086	4,80941358	3,76003086	3,84664352	4,76003086	0,015625	0,5625	0,61188272	0,61188272	0,61188272
150	3,30806327	4,16743827	4,16743827	4,96990741	3,87114198	4,96990741	3,18306327	3,77237654	1,02797068	1,57484568	1,6242284	1,6242284	1,6242284
151	3,76099537	3,37037037	3,37037037	3,67901235	5,17283951	3,67901235	4,13599537	5,27160494	2,30420525	1,91358025	1,56790123	1,56790123	1,56790123
152	4,37827932	3,98765432	3,98765432	4,2962963	5,79012346	4,2962963	4,75327932	5,88888889	2,9214892	2,5308642	2,18518519	2,18518519	2,18518519
153	4,37827932	3,98765432	3,98765432	4,2962963	5,79012346	4,2962963	4,75327932	5,88888889	2,9214892	2,5308642	2,18518519	2,18518519	2,18518519
154	4,06963735	3,67901235	3,67901235	3,98765432	5,48148148	3,98765432	4,44463735	5,58024691	2,61284722	2,22222222	1,87654321	1,87654321	1,87654321
155	3,89679784	3,50617284	3,50617284	3,81481481	5,30864198	3,81481481	4,27179784	5,40740741	2,81037809	2,41975309	2,07407407	2,07407407	2,07407407
156	4,28742284	3,89679784	3,89679784	4,20543981	5,69926698	4,20543981	4,66242284	5,79803241	3,57137346	3,18074846	2,83506944	2,83506944	2,83506944
157	3,76099537	3,37037037	3,37037037	3,67901235	5,17283951	3,67901235	4,13599537	5,27160494	2,30420525	1,91358025	1,56790123	1,56790123	1,56790123

Gambar 5.12 Tampilan halaman perhitungan level 3

5.3.5 Implementasi antarmuka halaman perhitungan level 4

Di halaman implementasi antarmuka menampilkan perhitungan level 4 diamana user dapat melihat proses perhitungan *sequential training* yang

menampilkan nilai alfa, nilai error, nilai delta alfam, dan alfa terbaru. Halaman tersebut juga menampilkan nilai kx dan wx dalam kelas positif dan kelas negatif, tak hanya itu halaman tersebut menampilkan nilai bias dan nilai bias. Berikut tampilan pada level 4 ditunjukin seperti gambar dibawah berikut yaitu Gambar 5.13

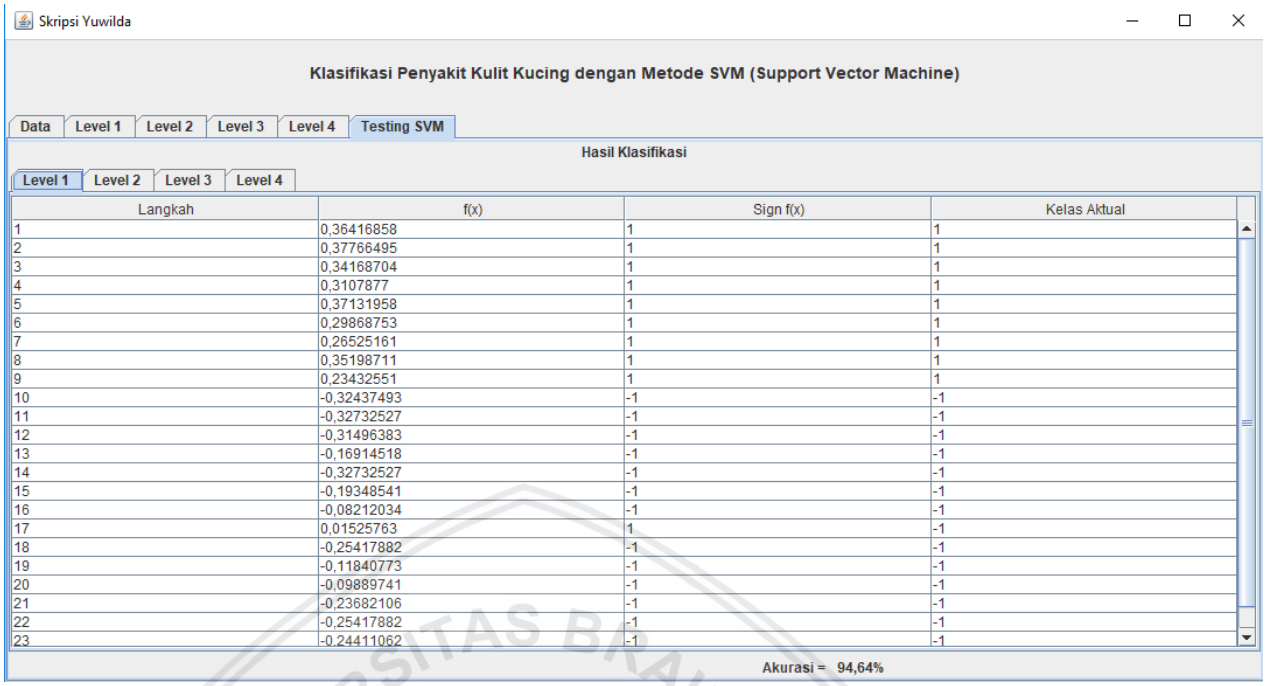


Gambar 5.13 Tampilan halaman perhitungan level 4

Penjelasan tentang Gambar 5.13 menampilkan halaman perhitungan level 3 yang mana tak jauh berbeda dengan tampilan perhitungan level sebelumnya. Hal yang membedakan ialah nilai hasil perhitungan yang ditampilkan perhitungan level 1, level 2, level 3, dan level 4. Hal ini dikarenakan jumlah data yang diproses berbeda.

5.3.6 Implementasi antarmuka halaman testing SVM

Berikut merupakan halaman antarmuka testing SVM yang didalamgnya terdapat hasil akurasi dengan menunjukan hasil akurasi pada testing SVM. Halaman muka ini dapat memperlihatkan dari level 1 sampai dengan level 4 yang dimana terdapat *actual class* dan terdapat nilai akurasi yang didapatkan. Tampilan antarmuka halaman testing SVM Gambar 5.14.



Gambar 5.14 Tampilan antarmuka halaman testing SVM

Gambar 5.14 menjelaskan mengenai tampilan halaman testing SVM yang berisi tampilan hasil klasifikasi dengan actual class di mana hasil klasifikasi SVM sesuai atau tidak dengan actual class beserta hasil akurasi yang diperoleh dari klasifikasi SVM.

BAB 6 PENGUJIAN DAN ANALISIS

Bab ini akan membahas tentang penjabaran dari pegujian dan analisis yang telah dilakukan pada sistem sesuai dengan skenario yang dibuat pada bab sebelumnya. Pengujian dilakukan untuk menentukan parameter dimana bertujuan untuk menentukan parameter terbaik. Pengujian dilakukan untuk mendapatkan nilai pengujian terhadap parameter gamma, pengujian nilai kernel, pengujian parameter parameter complexity, pengujian lambda, pengujian terhadap jumlah iterasi data, dan pengujian terhadap rasio data. Bab pengujian dan analisis ini sebelumnya telah dirancang pada b.

6.1 Sistematika Pengujian

Pada sub bab ini dalam sistematika pegujian akan membahas bagaimana pengujian dalam sitem. Dalam perancangannya sistem ini dibuat untuk mengklasifikasian 5 jenis penyakit dari 240 data dan 14 gejala penyakit kulit kucing. Pengujian pertama untuk menentukan nilai pengujian rasio data yang mana nilai pengujian didapatkan dari perbandingan jumlah banyaknya dataset training dan dataset testing. Selanjutnya sistem akan mencari nilai kernel dimana ada beberapa kernel unutuk membandingkan agar mendapatkan kernel yang terbaik. Setelah mendapatkan kernel dengan nilai terbaik selanjutnya sistem menghitung parameter – parameter pada SVM. Parameter yang dihitung diantaranya adalah gamma, complexity, dan lambda. Setelah itu pegujian yang terakhir adalah menentukan jumlah iterasi agar mendapatkan hasil akurasi terbaik.

6.2 Hasil dan Analisis Pembahasan

Pada sub bab ini akan menjelaskan tentang pemhasan dari hasil dan analisis yang sudah dilakukan. Analisis yang dilakukan pada sistem dengan menggunakan metode *Support Vector Machine* untuk mengklasifikasi lima jenis penyakit dari 14 gejala yang diketahui. Skenario pada pembahasan ini sebelumnya telah dirancang pada bab sebelumnya.

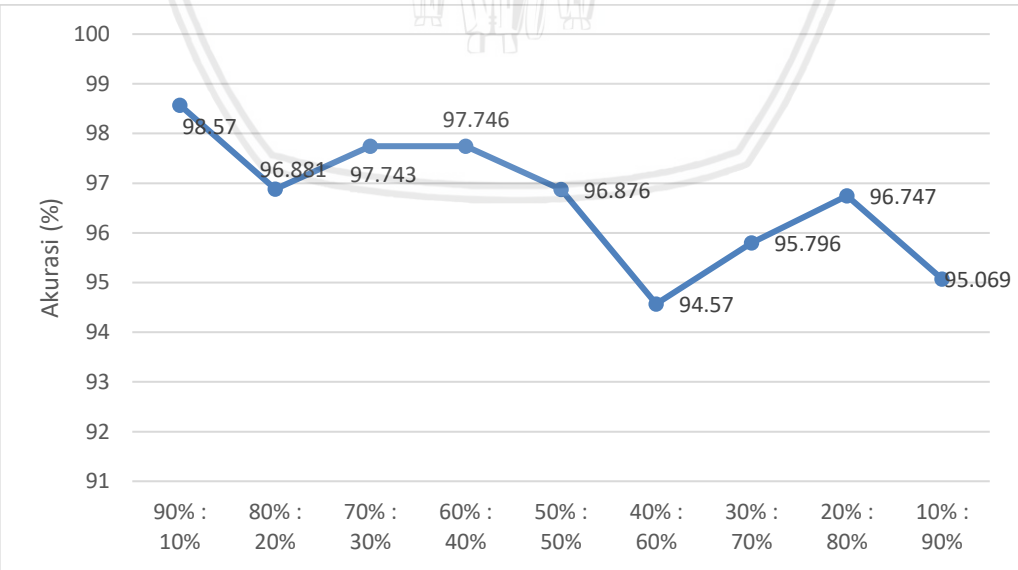
6.2.1 Pengujian Nilai Akurasi Terhadap Rasio Data

Pengujian terhadap rasio dataset sebanyak 240 data yang dibagi menjadi data training dan data testing berdasarkan skenario pengujian data yang sudah ditetapkan yaitu dengan rasio perbandingan jumlah data training dan jumlah data testing yaitu 90%:10%, 80%:20%, 70%:30%, 60%:40%, 50%:50%, 40%:60%, 30%:70%, 20%:80%, 10%:90%. Parameter pada pengujian ini sudah ditentukan dengan parameter yang akan mendapatkan nilai akurasi tertimggi diantaranya yaitu σ *Kernel* RBF = 0.7 $\lambda = 0.00001$ *y* = 0.01 *C* = 10 ε = 0.01 iterasi = 100. Berikut pengujian rasio data yang ditunjukan tabel 6.1

Tabel 6.1 Hasil pengujian perbandingan rasio data

Nilai *Rasio data*	Dataset ke-										Rata-Rata Akurasi
	1	2	3	4	5	6	7	8	9	10	
90:10	100	100	100	100	100	100	98,21	98,21	91,07	98,57	98,57
80:20	83,49	100	98,17	100	99,08	96,33	93,58	99,08	99,08	100	96,881
70:30	98,17	84,15	98,17	98,78	98,78	98,78	98,17	98,17	98,17	97,743	97,743
60:40	98,21	98,21	98,21	98,66	99,11	96,43	97,33	99,11	98,21	93,98	97,746
50:50	100	97,87	92,59	96,1	97,16	96,45	95,74	97,16	97,82	96,876	96,876
40:60	97,63	96,44	91,39	97,03	98,52	84,57	96,44	96,44	89,91	97,33	94,57
30:70	96,98	97,33	96,9	94,96	97,98	83,63	98,99	97,23	96,98	96,98	95,796
20:80	96,24	96,68	96,9	96,46	96,24	96,9	95,35	98,01	97,12	97,57	96,747
10:90	98,23	98,22	99,11	98,23	84,65	84,65	97,44	96,65	96,46	97,05	95,069

Dari Tabel 2.1 dibuatlah grafik hasil pengujian nilai rata-rata akurasi terhadap perbandingan rasio dengan menggunakan parameter yang sudah ditentukan.



Gambar 6.1 Grafik Hasil uji coba rasio data tingkat akurasi

Dari Gambar 6.1 didapatkan dari pengujian skenario perbandingan rasio data yaitu tingkat akurasi tertinggi didapatkan pada perbandingan rasio data 90:10

dengan keakurasian sebesar 98,57%. Hasil akurasi ini dicoba dengan sepuluh kali percobaan yang dilakukan pada masing-masing perbandingan rasio data. Hasil pengujian perbandingan rasio data memiliki nilai rata-rata lebih dari 90%, dapat dilihat nilai keakurasian memang tidak tetap tapi stabil pada nilai akurasi mencapai lebih dari 90%. Pada pengujian ini data yang digunakan pada prosesnya dilakukan secara acak setiap kelasnya.

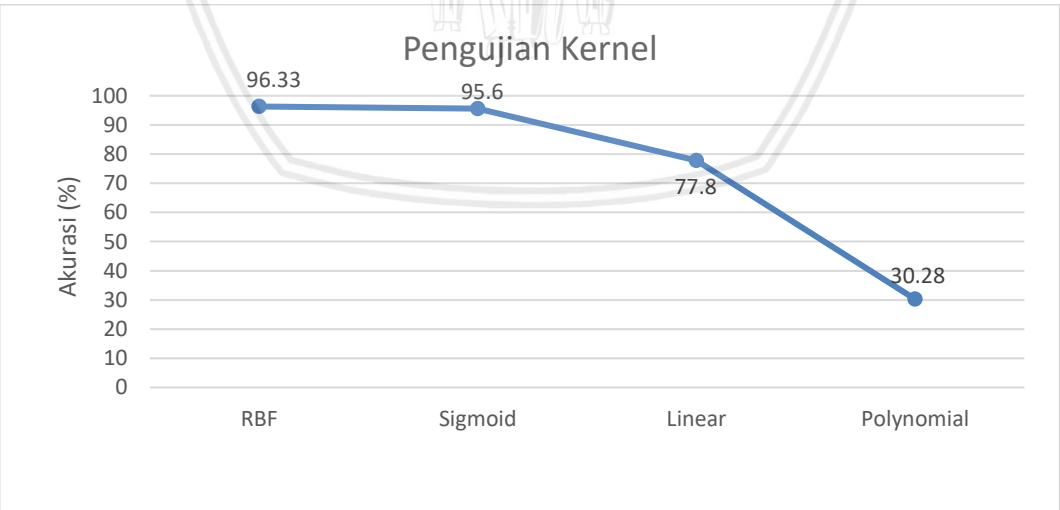
6.2.2 Pengujian Terhadap Jenis Kernel

Selanjutnya dilakukan pengujian pada jenis, antara lain kernel RBF, kernel Sigmoid, kernel linear, kernel polynomial. Masing-masing jenis kernel diuji dengan parameter *sequential training* SVM diantaranya $\sigma = 0.7$ $\lambda = 0.00001$ $y =$ 0.01 $C =$ 10 ε = 0.01 iterasi = 100 dengan perbandingan rasio 90:10. Pengujian dilakukan untuk mengetahui jenis kernel mana yang memiliki nilai akurasi yang paling tinggi.

Tabel 6.2 Tabel hasil pengujian terhadap jenis kernel

Jenis	Dataset ke-										Rata-Rata Akurasi
	1	2	3	4	5	6	7	8	9	10	
RBF	100	100	100	100	100	100	98,21	98,21	91,07	98,57	98,57
Sigmoid	83,49	100	98,17	100	99,08	96,33	93,58	99,08	99,08	100	96,881
Linear	79,82	78,91	77,06	74	78,90	77,06	77,06	74,31	79,82	79,80	77,80s
Polynomial	30,36	30,36	30,36	30,36	30,36	30,36	30,36	30,36	30,36	30,36	30,36

Data pada Tabel 6.2 dibuat grafik pengujian nilai rata-rata akurasi yang didapatkan dari pengujian jenis kernel yang digunakan.



Gambar 6.2 Grafik hasil pengujian Jenis Kernel

Dari Gambar 6.2 disimpulkan bahwa kernel RBF memiliki hasil nilai akurasi tertinggi dengan nilai 98.57% disusul dengan kernel sigmoid sebesar 95.60% selanjutnya kernel linear sebesar 77.80% dan terakhir kernel polynomial sebesar

30.36%. pada Gambar 6.2 akurasi tertinggi yakni kernel RBF karena kernel RBF cocok digunakan dengan tipe data yang digunakan. Kernel RBF dapat mengklasifikasikan data traing sesuai dengan kelas aslinya.

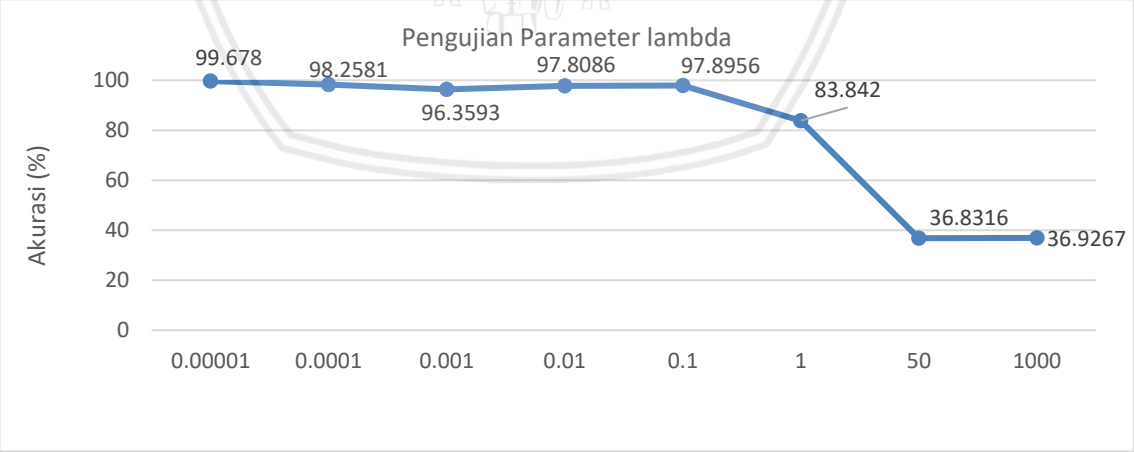
6.2.3 Pengujian Terhadap Parameter λ *(Lambda)*

Pengujian pada parameter *lambda* diantaranya yaitu 0.00001 , 0.0001 , 0.001 , 0.01 , 0.1 , 1 , 50, 100. Pada pengujian ini digunakan parameter yaitu σ *Kernel* RBF = 0.7 *y* = 0.01 *C* = 10 ε = 0.01 iterasi = 100 dengan perbandingan rasio 90%:10%. Nilai parameter yang digunakan nantinya akan diuji dengan tujuan mengetahui apakah nilai paramater lambda pada *sequential training SVM* akan mempengaruhi tingkat keakurasian atau tidak.

Tabel 6.3 Tabel hasil pengujian terhadap parameter lambda

Nilai *Lamda*	Dataset ke-										Rata-Rata
	1	2	3	4	5	6	7	8	9	10	
0,00001	98,21	100	100	100	100	100	98,21	100	100	100	99,678
0,0001	100	100	98,21	96,43	94,64	100	98,21	98,21	100	100	98,2581
0,001	94,64	98,21	98,21	98	100	96,43	83,93	96,43	98,21	96,3	96,3593
0,01	100	98,21	100	98,21	96,43	91,07	98,21	98,21	100	100	97,8086
0,1	100	98,21	96,43	94,6	98,21	98,21	100	100	98,21	98,21	97,8956
1	91,07	91,07	64,64	67,86	89,93	85,71	82,14	87,5	82,14	92,86	83,842
50	30,28	30,28	30,28	30,28	30,28	30,28	30,28	30,28	30,28	30,28	36,8316
100	30,28	30,28	30,28	30,28	30,28	30,28	30,28	30,28	30,28	30,28	36,9267

Dari data yang didapatkan pada Tabel 6.3 dibuatlah grafik pengujian terhadap nilai lambda dengan menggunakan parameter yang sudah ditentukan



Gambar 6.3 Grafik hasil uji coba terhadap parameter lambda

Hasil analisis dari grafik yang ditunjukan pada Gambar 6.3 diketahui bahwa akurasi tertinggi didapatkan dari lambda sebesar 0.00001 dengan tingkat akurasi yaitu 99.678% dissssusul dengan nilai lambda 0.0001 sebesar 98.25%, selanjutnya dengan nilai lamda sebesar 0.001 tingkat keakurasian mencapai 96.35% ,

kemudian nilai lamda sebesar 0.01 memiliki keakurasian sebesar 97,8%. Dari grafik kita dapat melihat grafik nilai rata-rata mengalami penurunan dan stabil ketika nilai lamda lebih besar dari pada 1, itu disebabkan karena nilai lamda yang besar sehingga mengakibatkan proses waktu komputasi pada perhitungan matriks *Hessian* membutuhkan waktu yang yang lebih lama juga. Menurut Vijayakumar & Wu, 1999 proses ini diakibatkan karena nilai lamda dapat membuat sistem lebih lambat untuk dapat mencapai titik temu dan menyebabkan ketidakstabilan pada saat pembelajarannya.

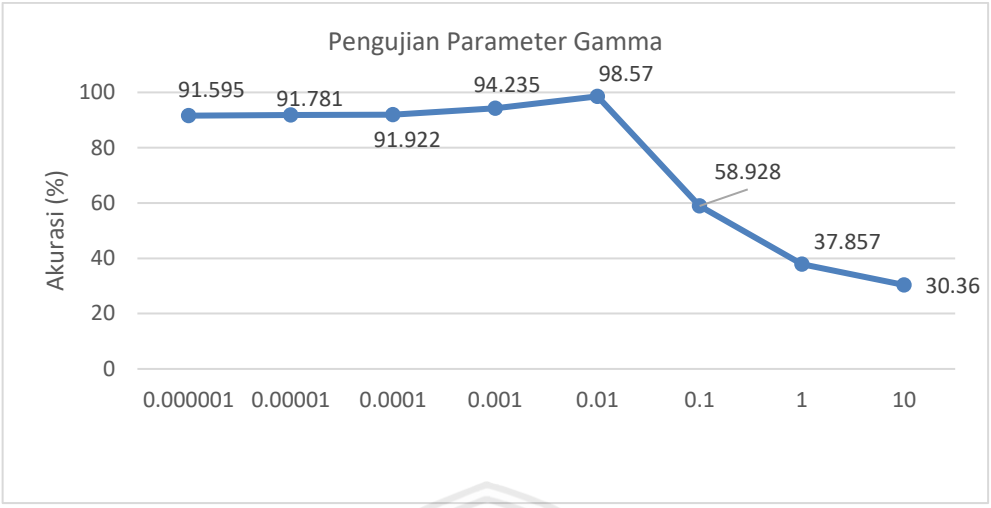
6.2.4 Pengujian Terhadap Parameter *y (gamma)*

Pada pengujian parameter pengujian y *(gamma)* merupakan pengujian dimana parameternya memiliki pengaruh besar dalam menentukan parameter terbaik, berikut nilai parameter y *(gamma)* yang akan diuji pada pengujian ini 0.000001, 0.00001, 0.0001, 0.001, 0.01, 0.1, 1, 10. Parameter y *(gamma)* diuji menggunakan parameter *sequential training SVM* yakni nilai σ *Kernel* RBF = 0.7 $\lambda = 0.0001$ *C* = 10 ε = 0.01 iterasi = 100 dan rasio data 90%:10%. Berikut hasil pengujian nilai gamma ditunjukkan oleh Tabel 6.4.

Tabel 6.4 Tabel hasil pengujian terhadap parameter gamma

Nilai *gamma*	Dataset ke-										Rata-Rata Akurasi
	1	2	3	4	5	6	7	8	9	10	
0,000001	91,07	92,86	91,07	91	91,07	91,07	91,07	91,07	92,81	92,86	91,595
0,00001	92,86	91,07	91,07	94,64	91,07	92,86	91,07	91,07	92,81	89,29	91,781
0,0001	91,07	92,86	92,86	96	92,86	91,07	91,07	91,07	91,07	89,29	91,922
0,001	94,64	96,43	94,64	91,07	94,64	92,86	92,86	94,64	94,14	96,43	94,235
0,01	100	100	100	100	100	100	98,21	98,21	91,07	98,57	98,57
0,1	60,71	60,71	60,71	60,71	60,71	60,71	53,58	58,93	58,93	53,58	58,928
1	30,36	41,07	41,07	41,07	41,07	41,07	41,07	41,07	30,36	30,36	37,857
10	30,36	30,36	30,36	30,36	30,36	30,36	30,36	30,36	30,36	30,36	30,36

Dari hasil data Tabel 6.4 maka dapat dibuatlah grafik hasil akurasi pengujian nilai rata-rata nilai gamma.



Gambar 6.4 Grafik hasil uji coba terhadap parameter gamma

Hasil analisis yang dilakukan dapat dilihat pada Gambar 6.4 dari grafik tersebut diketahui pengujian rata – rata memiliki nilai akurasi parameter gamma yang paling tinggi adalah 98.57% dengan nilai gamma 0,01. Nilai gamma sangat mempengaruhi nilai rata – rata keakurasian hal tersebut dapat dilihat dari grafik pada Gambar 6.4. dari hasil grafik diatas dapat disimpulkan bahwa nilai gamma 0,1 mengalami penurunan dengan nilsai 58.928% hal itu disebabkan karena apabila nilai gamma semakin besar maka mempengaruhi besar hasil nilai *learning rate*. Learning rate adalah nilai proses pelatihan atau pembelajaran. Learning rate berpengaruh dari nilai gamma yang dihasilkan.

6.2.5 Pengujian Terhadap Parameter C (Complexity)

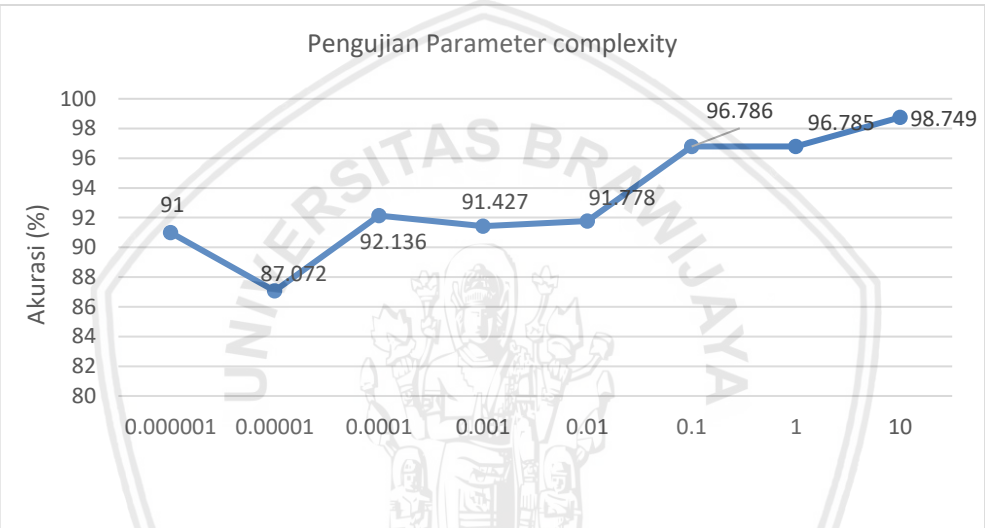
Pengujian terhadap parameter complexity diantaranya menggunakan nilai 0.000001, 0.00001, 0.0001, 0.001, 0.01, 0.1, 1, 10. Masing – masing tiap nilai parameter gamma diuji dengan nilai parameter *sequential training SVM* diantaranya yaitu σ *Kernel* RBF = 0.7 $\lambda = 0.0001$ $y =$ 0.01 ε = 0.01 iterasi = 100 dengan perbandingan rasio 90%:10%. Berikut hasil nilai rata – rata terhadap parameter complexit ditunjukan pada Tabel 6.5.

Tabel 6.5 Tabel hasil pengujian terhadap parameter complexity

Nilai *complexity*	Dataset ke-										Rata-Rata
	1	2	3	4	5	6	7	8	9	10	
0,000001	94,64	92,86	91,07	91	91,07	91,07	89,29	92,86	94,64	91,07	91,957
0,00001	52,86	89,29	89,29	91,07	91,07	91,07	91,07	92,86	91,07	91,07	87,072
0,0001	91,07	94,64	89,29	91	92,86	92,86	92,86	91,07	94,64	91,07	92,136
0,001	91,07	91,07	94,64	91,07	91,07	91,07	91,07	91,07	91,07	91,07	91,427
0,01	92,86	91,07	92,86	91,	91,07	91,07	91,07	91,07	91,07	94,64	91,778

Nilai *complexity*	Dataset ke-										Rata-Rata
	1	2	3	4	5	6	7	8	9	10	
0,1	96,43	96,43	98,21	96,43	96,43	96,43	96,43	98,21	96,43	96,43	96,786
1	96,43	98,21	96,43	96,43	96,43	94,64	96,43	98,21	98,21	96,43	96,785
10	100	100	100	100		100	100	98,21	98,21	91,07	98,749

Dapat disimpulkan pada data Tabel 6.5 dari grafik pengujian terhadap nilai rata-rata complexity. Berikut grafik pengujian nilai rata- rata ditunjukan oleh Gambar 6.5.



Gambar 6.5 Grafik hasil uji coba terhadap parameter Complexity

Analisis yang didapatkan dari grafik pada Gambar 6.5 nilai akurasi tertinggi didapatkan dengan nilai complexity dengan nilai 10 yakni sebesar 98.749%. Pada grafik diatas disimpulkan bahwa nilai rata – rata akurasi complexity semakin meningkat apabila nilai complexity semakin besar, itu disebabkan karena pengujian nilai complexity bertujuan untuk meminimalkan nilai error. Tujuan yang lain ialah memperkecil nilai *slack* variable karena pada saat nilai complexity semakin besar nilai margin hyperlane akan semakin lebar, sehingga penalti yang diberikan ketika nilai complexity semakin besar akan memaksimalkan nilai error pada proses klasifikasi. (Nugroho, et al., 2003)

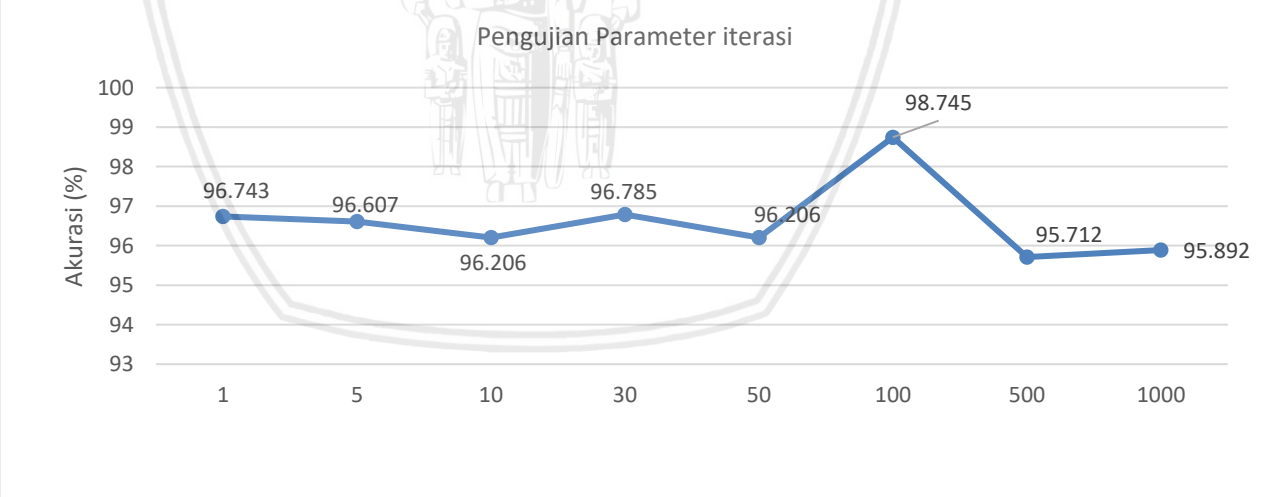
6.2.6 Pengujian terhadap jumlah iterasi

Tahap pengujian terhadap parameter jumlah iterasi diantaranya 1, 5, 10, 30, 50, 100, 500, 1000. Setelah itu nilai parameter jumlah iterasi diuji dengan nilai *sequential training* SVM diantaranya σ *Kernel* RBF = 0.7 $\lambda = 0.0001$ y = 0.01 ε = 0.01 C = 10 dengan perbandingan rasio 90%:10%. Pengujian dilakukan untuk mengetahui parameter terbaik untuk mendapatkan akurasi yang tinggi. Berikut tabel hasil pengujian terhadap jumlah iterasi yang dilihat pada Tabel 6.6.

Tabel 6.6 Tabel hasil pengujian terhadap jumlah iterasi

Nilai Iterasi	Dataset ke-										Rata-rata Akurasi
	1	2	3	4	5	6	7	8	9	10	
1	96,43	96,43	98,21	96	98,21	96,43	96,43	96,43	96,43	96,43	96,743
5	96,43	98,21	96,43	96,43	96,43	98,21	96,43	96,43	94,64	96,43	96,607
10	96,43	98,21	94,64	96	94,64	96,43	96,43	98,21	96,43	94,64	96,206
3s0	96,43	96,43	98,21	96,43	96,43	96,43	96,43	94,64	98,21	98,21	96,785
50	96,43	96,43	96,43	96	94,64	94,64	98,21	96,43	98,21	94,64	96,206
100	100	100	100	100	100	100	100	98,21	98,21	91,07	98,745
500	94,64	94,64	98,21	94,64	96,43	96,43	94,64	94,64	98,21	94,64	95,712
1000	96,43	94,64	94,64	96,43	98,21	96,43	94,64	94,64	96,43	96,43	95,892

Dari data yang ditunjukan oleh Tabel 6.6 dibuatlah grafik hasil rata – rata nilai pengujian jumlah iterasi. Berikut grafik nilai pengujian parameter terhadap jumlah iterasi yang ditunjukan oleh Gambar 6.6.



Gambar 6.6 Grafik hasil uji coba terhadap jumlah iterasi

Hasil analisis yang dari grafik pada Gambar 6.6 menunjukan bahwa nilai akurasi tertinggi didapatkan pada jumlah iterasi bernilai 100 dengan tingkat akurasi sebesar 98.745%. Grafik yang didapatkan dari percobaan pegujian diatas nilai rata-rata yang didapatkan sebesar 90% keatas. Nilai jumlah iterasi berpengaruh terhadap pengaruh pada nilai α *(alpha),* dikaerenakan jumlah iterasi sangat berpengaruh agar nilai alpha yang didapatkan menjadi konvergen.

BAB 7 PENUTUP

Bab penutup ini menjelaskan tentang kesimpulan yang didapatkan dari penelitian klasifikasi penyakit kulit kucing menggunakan metode *Support Vector Machine*.

7.1 Kesimpulan

Dari penelitian yang didapatkan mengenai klasifikasi penyakit kulit kucing dengan algoritma *Support Vector Machine* dapat ditarik kesimpulan bahwa*:*

1. Dapat disimpulkan bahwa dengan menggunakan algoritme *Support Vector Machine* dapat diimplementasikan pada klasifikasi penyakit kuit kucing dengan menggunakan *dataset* sebesar 240 dataset dengan jumlah parameter sebanyak 14. Pada penelitian ini terdapat lima kelas diantaranya kelas scabies, cat flea, abses, dermatitis, dan jamur.
2. Algoritme *Support Vector Machine* memberikan hasil Memberikan hasil terbaik dengan akurasi tertinggi sebesar 98.745% menggunakan nilai parameter pada *sequential training* SVM dengan nilai $\lambda = 0.00001$ *y* = 0.01 *C* = 10 ε = 0.01 iterasi = 100 dan rasio data 90%:10%.
3. Disimpulkan bahwa kernel RBF memiliki hasil nilai akurasi tertinggi dengan nilai 98.57% disusul dengan kernel sigmoid sebesar 95.60% selanjutnya kernel linear sebesar 77.80% dan terakhir kernel polynomial sebesar 30.36%.

7.2 Saran

Sistem klasifikai penyakit kulit kucing dengan menggunakan metode algoritma *Support Vector Machine* ini masih terdapat beberapa kekurangan. Saran agar penelitian ini supaya menjadi lebih baik adalah:

1. Sistem ini sebaiknya dikembangkan dengan menggunakan metode yang berbeda atau mengkombinasikan dengan metode lain agar lebih akurat.
2. Diperlukan data penyakit kulit kucing lebih banyak agar sistem dapat memberikasn keakuratan pada akurasi yang didapatkan

DAFTAR PUSTAKA

Hakim, A.M., 2015. Sistem Pakar Identifikasi Penyakit Kulit Anjing Menggunakan Metode Certainty Factor. STMIK STIKOM Surabaya.

Kurniati , N. et al., 2017. Sistem Pakar Untuk Mendiagnosa Penyakit Kulit Pada Kucing Menggunakan Certainty Factor. *ILKOM Jurnal Ilmiah.*

Kusumaningrum (2012) 'Sistem Pakar Diagnosa Penyakit Kulit Pada Kucing Dengan Metode CF', (5), pp. 11–14

Hartuti, R. S., Adam, M. & Murtina, T., 2014. Kajian Kesejahteraaan Kucing Yang Dipelihara Pada Beberapa Pet Shop Di Wilayah Bekasi, Jawa Barat. *Medika Veterinaria.*

Pitaloka, 2016. Implementasi Metode Naive Bayes Classivier Untuk Diagnosis Klasifikasi Gejala Penyakit Kucing (Studi Kasus Klinik Hewan Universitas Brawijata) Malang.

Way, N., 2014 Cat Bite Abscesses Http://Wwwnicklinwayvet.Com.Au/Wp-Content/Uploads/2014/07/Catbite-Abscess.Pdf>

Palguna (2014) 'Jurnal Sistem Informasi Jurnal Sistem Informasi', 3(1), pp. 140–143.

Anto, S. N., Arief, B. W. & Dwi, H., 2003. *Support Vector Machine -Teori danAplikasinya dalam Bioinformatika1–.* s.l., Kuliah Umum IlmuKomputer.Com

Darsyah, M. Y., November2014. *Klasifikasi Tuberkulosis dengan Pendekatan Metode Support Vector Machine (SVM).* Semarang, Program Studi Statistika Fakultas Matematika dan Ilmu Pengetahuan Alam, Universitas Muhammadiyah.

Puspita , S., Wihandika, R. C. & Muflikah, L., 2018. Klasifikasi Kualitas Susu Sapi Menggunakan Metode *Support Vector Machine* (SVM). *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer,* Volume 2.

Nugroho, A., Witarto, A. & Handoko, D., 2003. *Support Vector Machine-Teori dan Aplikasinya.* s.l., s.n

Puspitasari, A. M., Ratnawati, D. E. & Widodo, A. W., 2018. *Klasifikasi Penyakit Gigi dan Mulut Menggunakan Metode Support Vector Machine.* Malang: Univeritas Brawijaya.

Rachman, 2012. *Perbandingan Klasifikais Tingkat Keganasan Breast Cancer dengan Menggunakan Regresi Logistik Ordinal dan Support Vector Machine.* s.l., s.n.

Munawarah , R., Soesanto, O. & Faisal, M. R., 01 Februari 2016. Penerapan Metode Support Vector Machine Pada Diagnosa Hepatitis. *Kumpulan jurnal, Ilmu Komputer (KLIK),* Volume Volume 04, No. 01 Februarl 2016.

Ali, E. E. E. A. & Feng, W. . Z., 2014. Breast Cancer Classification using Support Vector. *International Journal of Science and Research (IJSR),* Volume Index Copernicus Value (2013): 6.14 |.

Sembiring, K., September 2007. *Penerapan Teknik Support Vector Machine untuk Pendeteksian Intrusi pada Jaringan.* Bogor, S1 Teknik Informatika, Sekalah Teknik Elektro dan Informatika, ITB.

Vijayakumar S, W. S., 1999. Sequential *Support Vector* Classifiers and Regression. *Proc. International Conference on Soft Computing,* Issue (SOCO'99),Genoa, Italy,, pp. pp.610-619,.

Winarko2 & Suwanto, R. E., Oktober 2014. Klasterisasi, Klasifikasi dan Peringkasan Teks Berbahasa Indonesia. *Prosiding Seminar Ilmiah Nasional Komputer dan Sistem Intelijen (KOMMIT 2014),* Vol. 8(ISSN : 2302-3740).

Penyakit Kulit Kucing Yang Umum Ditemui Di Indonesia.

Http://Ternakkucing.Blogspot.Co.Id/2012/10/Penyakit-Kulit-Pada-Kucing-Yang-Umum-Di Html> (Effendi C 2012)

.

DATA SET PENYAKIT KULIT KUCING

No	G1	G2	G3	G4	G5	G6	G7	G8	G9	G10	G11	G12	G13	G14	PENYAKIT
1	9	0	8	8	8	5	0	0	8	0	0	0	0	0	Scabies
2	8	0	8	8	5	5	0	0	7	0	0	0	0	0	Scabies
3	8	0	8	8	5	5	0	0	8	0	5	0	0	0	Scabies
4	8	0	8	8	8	5	0	0	8	0	0	0	0	0	Scabies
5	8	0	8	8	8	5	0	0	8	0	0	0	0	0	Scabies
6	8	0	8	8	8	5	0	0	8	0	0	0	0	0	Scabies
7	5	0	8	8	5	5	0	0	6	0	0	0	0	0	Scabies
8	8	0	8	8	5	5	0	0	8	0	0	0	0	0	Scabies
9	5	0	8	8	8	5	0	0	8	0	0	0	0	0	Scabies
10	9	0	9	9	9	5	8	0	9	0	5	0	0	0	Scabies
11	8	0	8	8	5	8	0	0	8	0	0	0	0	0	Scabies
12	8	0	8	8	5	8	0	0	8	0	0	0	0	0	Scabies
13	8	0	8	8	8	5	0	0	8	0	0	0	0	0	Scabies
14	9	0	8	8	8	5	0	0	8	0	0	0	0	0	Scabies
15	8	0	8	8	9	8	0	0	8	0	0	0	0	0	Scabies
16	8	0	8	6	5	8	0	0	8	0	0	0	0	0	Scabies
17	8	0	8	8	9	8	0	0	8	0	0	0	0	0	Scabies
18	8	0	8	8	8	5	0	0	8	0	0	0	0	0	Scabies
19	5	0	8	8	8	8	5	0	8	0	0	0	0	0	Scabies
20	8	0	8	8	8	5	0	0	8	0	0	0	0	0	Scabies
21	8	0	5	8	8	5	0	0	8	0	0	0	0	0	Scabies
22	5	0	8	8	5	8	0	0	8	0	0	0	0	0	Scabies
23	8	0	9	8	8	5	0	0	9	0	5	0	0	0	Scabies
24	8	0	8	8	9	8	0	0	9	0	5	0	0	0	Scabies
25	9	0	8	8	9	8	8	0	8	0	0	0	0	0	Scabies
26	8	0	8	8	5	5	0	0	8	0	0	0	0	0	Scabies
27	8	0	8	8	5	5	0	0	8	0	0	0	0	0	Scabies
28	8	0	8	8	8	8	0	0	8	0	0	0	0	0	Scabies
29	5	0	8	8	8	8	0	0	8	0	0	0	0	0	Scabies
30	8	0	9	8	9	8	0	0	8	0	0	0	0	0	Scabies
31	8	0	8	8	8	8	0	0	8	0	0	0	0	0	Scabies
32	8	0	8	8	5	8	0	0	8	0	0	0	0	0	Scabies
33	8	0	8	8	8	5	0	0	8	0	0	0	0	0	Scabies
34	8	0	8	8	8	5	0	0	8	0	0	0	0	0	Scabies
35	8	0	8	8	5	8	0	0	8	0	0	0	0	0	Scabies

36	5	0	8	8	5	5	0	0	8	0	0	0	0	0	Scabies
37	8	0	8	8	8	5	0	0	8	0	0	0	0	0	Scabies
38	8	0	8	8	8	5	0	0	8	0	0	0	0	0	Scabies
39	5	0	8	8	8	5	0	0	8	0	0	0	0	0	Scabies
40	5	0	8	8	8	8	0	0	8	0	5	0	0	0	Scabies
41	8	0	8	8	8	8	0	0	8	0	0	0	0	0	Scabies
42	8	0	8	8	8	8	0	0	8	0	0	0	0	0	Scabies
43	8	0	8	8	8	8	0	0	8	0	0	0	0	0	Scabies
44	8	0	8	8	8	8	0	0	8	0	0	0	0	0	Scabies
45	8	0	8	8	5	8	0	0	8	0	0	0	0	0	Scabies
46	8	0	8	8	8	8	0	0	8	0	0	0	0	0	Scabies
47	8	0	8	8	5	8	0	0	8	0	0	0	0	0	Scabies
48	8	0	8	8	8	8	0	0	8	0	0	0	0	0	Scabies
49	8	0	8	8	8	8	0	0	8	0	5	0	0	0	Scabies
50	8	0	8	8	8	8	0	0	8	0	0	0	0	0	Scabies

51	8	0	8	8	8	8	0	0	8	0	0	0	0	0	Scabies
52	8	0	8	7	5	8	0	0	8	0	0	0	0	0	Scabies
53	8	0	8	8	8	8	0	0	8	0	0	0	0	0	Scabies
54	8	0	8	8	8	8	0	0	8	0	0	0	0	0	Scabies
55	8	0	8	8	8	8	0	0	8	0	0	0	0	0	Scabies
56	8	0	8	8	8	8	0	0	8	0	0	0	0	0	Scabies
57	8	0	8	8	5	8	0	0	7	0	0	0	0	0	Scabies
58	8	0	8	8	8	8	0	0	8	0	0	0	0	0	Scabies
59	8	0	8	8	8	8	0	0	8	0	0	0	0	0	Scabies
60	8	0	8	8	8	8	0	0	8	0	0	0	0	0	Scabies
61	8	0	8	8	8	8	0	0	8	0	0	0	0	0	Scabies
62	5	0	8	8	8	8	0	0	8	0	0	0	0	0	Scabies
63	5	0	7	8	6	8	0	0	8	0	5	0	0	0	Scabies
64	5	0	8	8	8	8	0	0	8	0	0	0	0	0	Scabies
65	8	0	8	8	8	8	0	0	8	0	0	0	0	0	Scabies
66	8	0	7	6	8	7	0	0	8	0	0	0	0	0	Scabies
67	9	0	8	9	9	8	5	0	8	0	0	0	0	0	Scabies
68	9	0	8	8	9	8	0	0	8	0	5	0	0	0	Scabies
69	8	0	8	8	8	8	0	0	8	0	0	0	0	0	Scabies
70	5	0	8	8	5	8	0	0	8	0	0	0	0	0	Scabies
71	8	0	8	8	5	5	0	0	8	0	0	0	0	0	Scabies
72	5	0	8	8	5	5	0	0	7	0	0	0	0	0	Scabies
73	8	0	8	8	8	5	0	0	8	0	0	0	0	0	Scabies
74	8	0	7	7	8	8	0	0	8	0	0	0	0	0	Scabies

75	8	0	8	8	8	8	0	0	8	0	0	0	0	0	Scabies
76	8	0	8	8	8	5	0	0	8	0	0	0	0	0	Scabies
77	8	0	8	8	8	5	0	0	8	0	5	0	0	0	Scabies
78	7	0	6	6	8	5	0	0	8	0	0	0	0	0	Scabies
79	8	0	8	8	8	5	0	0	8	0	0	0	0	0	Scabies
80	8	0	8	8	8	5	0	0	8	0	0	0	0	0	Scabies
81	8	0	8	8	8	5	0	0	8	0	0	0	0	0	Scabies
82	8	0	8	8	8	5	0	0	8	0	0	0	0	0	Scabies
83	7	0	8	8	8	5	0	0	8	0	0	0	0	0	Scabies
84	8	0	8	8	8	5	0	0	8	0	0	0	0	0	Scabies
85	8	0	8	8	5	5	0	0	6	0	5	0	0	0	Scabies
86	8	0	8	8	5	5	0	0	8	0	0	0	0	0	Scabies
87	0	8	5	0	5	0	0	0	8	0	0	0	0	8	Cat Flea
88	0	8	5	0	5	5	0	0	8	0	0	0	0	8	Cat Flea
89	0	8	5	0	5	5	0	0	8	0	0	0	0	8	Cat Flea
90	0	7	5	0	5	5	0	0	8	0	0	0	0	6	Cat Flea
91	0	8	5	0	5	5	0	0	8	0	0	0	0	8	Cat Flea
92	0	8	5	0	5	5	0	0	8	0	0	0	0	8	Cat Flea
93	0	8	5	0	5	5	0	0	8	0	0	0	0	8	Cat Flea
94	0	8	5	0	5	5	0	0	8	0	0	0	0	8	Cat Flea
95	0	7	5	0	5	5	0	0	8	0	0	0	0	7	Cat Flea
96	0	8	5	0	5	5	0	0	8	0	0	0	0	8	Cat Flea
97	0	8	5	0	5	5	0	0	8	0	0	0	0	8	Cat Flea
98	0	8	5	0	8	0	0	0	8	0	0	0	0	8	Cat Flea
99	0	8	5	0	8	5	0	0	8	0	0	0	0	8	Cat Flea
100	0	8	5	0	5	0	0	0	8	0	0	0	0	8	Cat Flea
101	0	8	5	0	5	8	0	0	8	0	0	0	0	8	Cat Flea
102	0	9	8	0	8	8	0	0	8	0	0	0	0	8	Cat Flea
103	0	9	8	0	8	8	0	0	0	0	0	0	0	8	Cat Flea
104	0	9	7	0	8	0	0	0	0	0	0	0	0	8	Cat Flea
105	0	9	7	0	8	0	0	0	0	0	0	0	0	7	Cat Flea
106	0	8	8	0	8	0	0	0	0	0	0	0	0	5	Cat Flea
107	0	8	5	0	5	0	0	0	0	0	0	0	0	5	Cat Flea
108	0	8	8	0	8	0	0	0	0	0	0	0	0	5	Cat Flea
109	0	8	8	0	8	0	0	0	0	0	0	0	0	5	Cat Flea
110	0	8	8	0	8	0	0	0	0	0	0	0	0	5	Cat Flea
111	0	8	8	0	8	0	0	0	0	0	0	0	0	5	Cat Flea

112	0	8	8	0	8	0	0	0	0	0	0	0	0	8	Cat Flea
113	0	8	8	0	5	0	0	0	0	0	0	0	0	8	Cat Flea
114	0	8	5	0	5	0	0	0	0	0	0	0	0	5	Cat Flea
115	0	8	8	0	5	0	0	0	0	0	0	0	0	5	Cat Flea
116	0	8	5	0	8	0	0	0	0	0	0	0	0	8	Cat Flea
117	0	8	8	0	8	0	0	0	0	0	0	0	0	5	Cat Flea
118	0	8	8	0	8	0	0	0	0	0	0	0	0	5	Cat Flea
119	0	8	8	0	8	0	0	0	0	0	0	0	0	8	Cat Flea
120	0	8	8	0	5	0	0	0	0	0	0	0	0	5	Cat Flea
121	0	8	5	0	8	0	0	0	0	0	0	0	0	5	Cat Flea
122	0	8	5	0	8	0	0	0	0	0	0	0	0	8	Cat Flea
123	0	8	5	0	8	0	0	0	0	0	0	0	0	8	Cat Flea
124	0	8	8	0	8	0	0	0	0	0	0	0	0	5	Cat Flea
125	0	8	8	0	5	0	0	0	0	0	0	0	0	5	Cat Flea
126	0	8	8	0	8	0	0	0	0	0	0	0	0	5	Cat Flea
127	0	8	8	0	8	0	0	0	0	0	0	0	0	5	Cat Flea
128	0	8	8	0	8	0	0	0	0	0	0	0	0	8	Cat Flea
129	0	8	8	0	8	0	0	0	0	0	0	0	0	8	Cat Flea
130	0	8	5	0	5	0	0	0	0	0	0	0	0	5	Cat Flea
131	0	8	8	0	5	0	0	0	0	0	0	0	0	5	Cat Flea
132	0	8	8	0	5	0	0	0	0	0	0	0	0	5	Cat Flea
133	0	8	8	0	8	0	0	0	0	0	0	0	0	8	Cat Flea
134	0	8	8	0	8	0	0	0	0	0	0	0	0	8	Cat Flea
135	0	8	5	0	5	0	0	0	0	0	0	0	0	5	Cat Flea
136	0	8	8	0	5	0	0	0	0	0	0	0	0	5	Cat Flea
137	0	8	8	0	5	0	0	0	0	0	0	0	0	8	Cat Flea
138	0	8	8	0	5	0	0	0	0	0	0	0	0	5	Cat Flea
139	0	5	8	0	8	0	0	0	0	0	0	0	0	5	Cat Flea
140	0	8	5	0	5	0	0	0	0	0	0	0	0	8	Cat Flea
141	0	8	5	0	5	0	0	0	0	0	0	0	0	8	Cat Flea
142	0	8	8	0	8	0	0	0	0	0	0	0	0	5	Cat Flea
143	0	8	5	0	8	0	0	0	0	0	0	0	0	5	Cat Flea
144	0	8	8	0	5	0	0	0	0	0	0	0	0	8	Cat Flea
145	0	8	5	0	5	0	0	0	0	0	0	0	0	5	Cat Flea
146	0	8	5	0	6	0	0	0	0	0	0	0	0	5	Cat Flea
147	0	8	8	0	8	0	0	0	0	0	0	0	0	5	Cat Flea
148	0	8	6	0	5	0	0	0	0	0	0	0	0	8	Cat Flea
149	0	0	0	0	0	0	0	0	0	0	0	8	0	0	Abses
150	0	0	0	0	0	0	0	0	0	0	0	8	0	0	Abses

151	0	0	0	0	0	5	0	5	0	0	0	8	0	0	Abses
152	0	0	0	0	0	5	0	7	0	0	0	8	0	0	Abses
153	8	0	0	0	0	8	0	8	0	0	0	8	0	0	Abses
154	8	0	0	0	0	6	0	8	0	0	0	8	0	0	Abses
155	0	0	0	0	0	5	0	0	0	0	0	8	0	0	Abses
156	0	0	0	0	0	5	0	0	0	0	0	8	0	0	Abses
157	0	0	0	0	0	5	0	0	0	0	0	8	0	0	Abses
158	0	0	5	0	5	8	0	0	6	0	0	0	0	0	Dermatitis
159	0	0	5	0	5	8	0	0	6	0	0	0	0	0	Dermatitis
160	0	0	5	0	5	8	5	0	8	0	0	0	0	0	Dermatitis
161	0	0	5	0	5	8	5	0	8	0	0	0	0	0	Dermatitis
162	0	0	5	0	5	8	5	0	8	0	0	0	0	0	Dermatitis
163	0	0	5	0	5	8	0	0	6	0	0	0	0	0	Dermatitis
164	0	0	5	0	5	8	5	0	8	0	0	0	0	0	Dermatitis
165	0	0	5	0	5	9	6	8	8	0	0	0	0	0	Dermatitis
166	0	0	5	0	5	8	0	0	8	0	0	0	0	0	Dermatitis
167	0	0	5	0	5	8	6	0	8	0	0	0	0	0	Dermatitis
168	0	0	5	5	5	8	0	0	0	5	5	0	5	0	Jamur
169	0	0	5	5	5	8	5	0	0	5	5	0	5	0	Jamur
170	0	0	5	5	5	8	5	0	0	5	8	0	5	0	Jamur
171	0	0	5	5	8	8	0	0	0	5	8	0	5	0	Jamur
172	0	0	8	8	8	8	0	0	0	8	9	0	5	0	Jamur
173	0	0	5	5	8	8	0	0	0	8	8	0	8	0	Jamur
174	0	0	9	8	9	8	8	0	0	9	9	0	8	0	Jamur
175	0	0	8	8	5	8	0	0	0	5	5	0	0	0	Jamur
176	0	0	8	8	8	8	0	0	0	5	5	0	0	0	Jamur
177	0	0	8	8	8	8	0	0	0	5	5	0	0	0	Jamur
178	0	0	6	8	5	8	0	0	0	5	0	0	0	0	Jamur
179	0	0	8	8	8	5	0	0	0	5	5	0	0	0	Jamur

180	0	0	8	8	8	5	0	0	0	5	5	0	0	0	Jamur
181	0	0	8	8	8	5	0	0	0	5	5	0	0	0	Jamur
182	0	0	8	8	5	5	0	0	0	5	5	0	0	0	Jamur
183	0	0	8	8	8	8	0	0	0	5	5	0	5	0	Jamur
184	0	0	8	8	8	5	5	0	0	0	5	0	5	0	Jamur
185	0	0	8	8	8	5	0	0	0	5	5	0	0	0	Jamur

186	0	0	8	8	8	5	0	0	0	5	5	0	0	0	Jamur
187	0	0	8	8	8	5	0	0	0	5	5	0	0	0	Jamur
188	0	0	8	6	5	0	0	0	0	0	0	0	5	0	Jamur
189	0	0	8	8	8	5	0	0	0	5	5	0	0	0	Jamur
190	0	0	8	8	8	5	0	0	0	5	5	0	5	0	Jamur
191	0	0	8	9	8	5	0	0	0	5	5	0	0	0	Jamur
192	0	0	7	8	6	5	0	0	0	0	5	0	0	0	Jamur
193	0	0	8	8	8	5	0	0	0	5	5	0	0	0	Jamur
194	0	0	8	8	8	5	0	0	0	5	5	0	0	0	Jamur
195	0	0	8	9	8	5	0	0	0	5	5	0	0	0	Jamur
196	0	0	8	8	8	5	0	0	0	0	0	0	0	0	Jamur
197	0	0	8	8	8	5	0	0	0	0	0	0	0	0	Jamur
198	0	0	8	8	8	5	0	0	0	0	5	0	0	0	Jamur
199	0	0	8	8	5	5	0	0	0	5	5	0	0	0	Jamur
200	0	0	8	9	8	5	0	0	0	5	5	0	0	0	Jamur
201	0	0	8	8	8	5	0	0	0	5	5	0	0	0	Jamur
202	0	0	8	8	8	5	0	0	0	0	0	0	0	0	Jamur
203	0	0	8	8	8	5	0	0	0	5	5	0	0	0	Jamur
204	0	0	8	8	8	5	0	0	0	0	0	0	0	0	Jamur
205	0	0	8	8	5	5	0	0	0	5	5	0	0	0	Jamur
206	0	0	8	8	8	5	0	0	0	5	5	0	0	0	Jamur
207	0	0	8	8	8	5	0	0	0	5	5	0	0	0	Jamur
208	0	0	8	9	8	8	5	0	0	8	8	0	5	0	Jamur
209	0	0	8	8	8	5	0	0	0	5	5	0	0	0	Jamur
210	0	0	8	8	8	5	0	0	0	5	5	0	0	0	Jamur
211	0	0	5	8	5	5	0	0	0	8	5	0	5	0	Jamur
212	0	0	8	8	5	5	0	0	0	8	5	0	5	0	Jamur

213	0	0	8	8	8	5	0	0	0	8	5	0	5	0	Jamur
214	0	0	8	8	5	5	0	0	0	8	5	0	0	0	Jamur
215	0	0	8	8	5	5	0	0	0	7	5	0	5	0	Jamur
216	0	0	8	8	5	5	0	0	0	6	5	0	0	0	Jamur

217	0	0	8	8	5	5	0	0	0	8	5	0	5	0	Jamur
218	0	0	5	8	8	5	0	0	0	8	5	0	5	0	Jamur
219	0	0	8	8	5	5	0	0	0	8	5	0	5	0	Jamur
220	0	0	8	8	5	5	0	0	0	8	5	0	5	0	Jamur
221	0	0	8	8	5	5	0	0	0	8	5	0	5	0	Jamur
222	0	0	8	8	5	5	0	0	0	8	5	0	5	0	Jamur
223	0	0	5	8	5	5	0	0	0	8	5	0	5	0	Jamur
224	0	0	8	8	5	5	0	0	0	8	5	0	5	0	Jamur
225	0	0	8	8	5	5	0	0	0	8	5	0	5	0	Jamur
226	0	0	8	8	5	5	0	0	0	7	5	0	5	0	Jamur
227	0	0	5	8	8	5	0	0	0	8	5	0	5	0	Jamur
228	0	0	8	8	8	5	0	0	0	8	5	0	5	0	Jamur
229	0	0	8	8	8	5	0	0	0	8	5	0	5	0	Jamur
230	0	0	8	8	8	5	0	0	0	8	5	0	5	0	Jamur
231	0	0	5	8	5	5	0	0	0	8	5	0	5	0	Jamur
232	0	0	8	8	5	5	0	0	0	8	5	0	5	0	Jamur
233	0	0	8	8	5	5	0	0	0	8	5	0	5	0	Jamur
234	0	0	8	8	8	5	0	0	0	8	5	0	5	0	Jamur
235	0	0	5	8	5	5	0	0	0	8	5	0	5	0	Jamur
236	0	0	8	8	5	5	0	0	0	6	5	0	5	0	Jamur
237	0	0	8	8	8	5	0	0	0	8	5	0	5	0	Jamur
238	0	0	5	8	5	5	0	0	0	8	5	0	5	0	Jamur
239	0	0	5	8	8	5	0	0	0	9	8	0	8	0	Jamur
240	0	0	8	8	5	5	0	0	0	9	8	0	8	0	Jamur