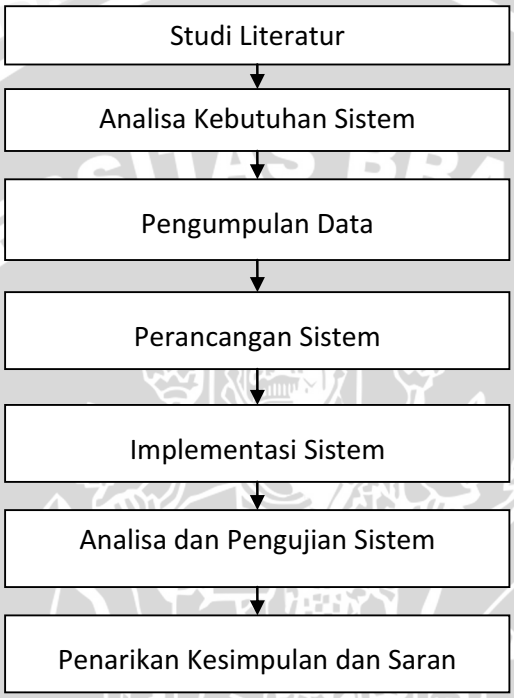
BAB 3 METODOLOGI

Pada bab ini akan dibahas metode-metode yang akan digunakan dalam pembuatan penelitian ini. Metodologi peelitian ini dilakukan melalui beberapa tahapan yaitu melakukan wawancara kepada pakar, pengumpulan data, analisis kebutuhan, perancangan sistem, implementasi sistem, pengujian sistem, dan pengambilan kesimpulan.



Gambar 3.1 Diagram Alir Penelitian.

Pada Gambar 3.1 di atas dapat diketahui diagram alir penelitian ini dimana setiap langkahnya memiliki peran masing-masing dan saling berkaitan. Berikut penjelasan tentang langkah-langkah dari alur penelitian.

1. Melakukan studi literatur mengenai pengklasifikasian irama detak jantung berdasarkan hasil pemeriksaan EKG (Elektrokardiografi), algoritma *Binary Decision Tree - Support Vector Machine*, dan literatur lain yang berhubungan dengan metode yang digunakan.
2. Melakukan analisa kebutuhan sistem untuk mengetahui secara keseluruhan kebutuhan yang diperlukan dalam membangun implementasi sistem meliputi kebutuhan *software*, *hardware*, data, fungsional dan non-fungsional.
3. Melakukan pengumpulan data. Data yang digunakan dalam penelitian ini merupakan data yang didapat dari *database MIT BIH Arrhythmia*. Data yang didapat berupa data mengenai EKG (Elektrokardiografi).
4. Melakukan perancangan sistem, menghitung proses manual dari algoritma, dan mendesain antarmuka atau tampilan dari sistem. Sistem dirancang untuk mengklasifikasi data menjadi empat kelas yaitu Normal, *PVC Bigeminy*, *Ventricular Tachycardia*, dan *Atrial fibrillation*. Perhitungan manual dilakukan

agar mengetahui bentuk sederhana dari algoritma yang dijalankan pada perangkat lunak. Pembuatan antarmuka memberikan kemudahan dan efisiensi waktu dalam melakukan perancangan.

5. Melakukan implementasi sistem dalam bentuk perangkat lunak yang dapat mengklasifikasi data kondisi detak jantung dari hasil EKG (Elektrokardiografi) menggunakan algoritma *Binary Decision Tree* dan *Support Vector Machine*.
6. Melakukan analisa dan pengujian sehingga dapat diketahui akurasi yang dihasilkan.
7. Menarik kesimpulan dan saran yang dapat digunakan sebagai referensi dan pengembangan pada penelitian yang selanjutnya.

3.1 Studi Literatur

Studi literatur dilakukan dengan cara mengumpulkan dan mempelajari literatur-literatur yang berkaitan dengan sistem ini. Literatur tersebut berkaitan dengan klasifikasi, hasil irama detak jantung, hasil pemeriksaan EKG (Elektrokardiografi), dan *binary decision tree-support vector machine*. Sumber literatur didapat dari jurnal, internet, laporan penelitian, dan pakar yang berkaitan dengan jantung.

3.2 Analisa Kebutuhan Sistem

Analisa kebutuhan bertujuan untuk mengetahui secara keseluruhan kebutuhan yang diperlukan dalam membangun implementasi sistem. Secara keseluran kebutuhan yang digunakan dalam implementasi penelitian ini adalah sebagai berikut :

1. Kebutuhan *hardware*, meliputi : Komputer /*PC* atau laptop.
2. Kebutuhan *software*, meliputi :
   - *Microsoft windows* sebagai sistem operasi.
   - Bahasa Pemrograman *Java*.
   - *Tool* yang digunakan yaitu *Netbeans IDE*.
3. Kebutuhan data, meliputi : Data diambil dari *database MIT BIH Arrhythmia*.
4. Kebutuhan fungsional dari sistem ini dapat di lihat pada Tabel 3.1 di bawah ini.

Tabel 3.1 Kebutuhan Fungsional Sistem.

No.	Nama Fungsi	Deskripsi
1	Menampilkan proses *input* dari data EKG untuk di proses.	Sistem bisa menampilkan proses *input* dari paramater-parameter berdasarkan data EKG sehingga bisa diproses untuk pengklasifikasian.
2	Menampilkan hasil klasifikasi dari kondisi detak jantung.	Menampilkan hasil klasifikasi dari kondisi detak jantung pasien pada hasil EKG (Elektrokardiografi) yang terdiri dari empat kelas yaitu, Normal, *PVC Bigeminy*, *Ventricular Tachycardia*, dan *Atrial fibrillation*.
3	Menampilkan proses	Sistem juga bisa menampilkan proses

	perhitungan menggunakan metode *BDT-SVM*.	perhitungan yang dilakukan oleh sistem berdasarkan algortima yaitu *BDT-SVM*.
4	Menampilkan gambar dari data EKG.	Sistem bisa menampilkan hasil akhir tidak hanya hasil klasifikasi saja tetapi juga menampilkan gambar EKG dari data uji yang dimasukkan oleh *user*.

5. Kebutuhan non-fungsional dari sistem ini dapat dilihat pada Tabel 3.2 dibawah ini.

Tabel 3.2 Kebutuhan Non-Fungsional Sistem.

No.	Nama Fungsi	Deskripsi
1	*User Interface*	Halaman yang mudah digunakan oleh pengguna dengan *user interface* yang baik.
2	Informasi yang lengkap	Sistem ini diharapkan bisa memberikan informasi yang lengkap kepada *user* (jenis kelainan jantung, gambar EKG) serta keakuratan hasil klasifikasi kondisi detak jantung.

3.3 Pengumpulan Data

Data yang dibutuhkan untuk penelitian ini adalah data untuk klasifikasi kondisi detak jantung dari hasil EKG (Elektrokardiografi). Data yang digunakan merupakan data yang didapat dari *Database MIT BIH Arrhythmia*. Untuk pengambilan parameter *input* diambil dari https://www.physionet.org/physiobank/database/html/mitdbdir/records.htm#205, untuk pengambilan gambar EKG dari http://www.physionet.org/lightwave/, dan untuk pengambilan nilai dari parameter *input* dari http://www.physionet.org/cgi-bin/atm/ATM. Data yang didapat berupa data mengenai EKG (Elektrokardiografi). Data tersebut terdiri dari empat kelas yaitu Normal, *Ventricular Bigeminy*, *Ventricular Tachycardia*, dan *Atrial fibrillation.* Jumlah data yaitu sebanyak 140 data dimana terdapat 35 data untuk setiap kelas.

3.4 Perancangan Sistem

Sistem yang dibuat bertujuan untuk mengklasifikasikan data kondisi detak jantung dari hasil EKG (Elektrokardiografi) menjadi empat kelas yaitu Normal, *Ventricular Bigeminy*, *Ventricular Tachycardia*, dan *Atrial fibrillation* dengan menggunakan algortima *Binary Decision Tree* dan *Support Vector Machine*. Data hasil EKG tersebut akan dibagi menjadi dua yaitu data *training*/latih dan data *testing*/uji. Perangkat lunak ini akan melakukan normalisasi pada data terlebih

dahulu dan selanjutnya akan dilakukan proses *training* menggunakan algortima *Binary Decision Tree* dan *Support Vector Machine*.

3.5 Implementasi Sistem

Implementasi dilakukan dengan membuat sistem berbasis *desktop* sehingga bahasa pemrograman yang digunakan adalah bahasa *java* dengan fitur-fitur dibawah ini.

1. Implementasi program : implementasi menggunakan bahasa pemrograman *Java*.
2. Pembuatan *user interface* dengan bantuan *adobe photoshop* dan desain sistem dengan menggunakan GUI pada Java.
3. Implementasi basis data : implementasi ini menggunakan *file.csv* sebagai penyimpanan data dan terdapat *database* yang digunakan dalam proses perhitungan nilai bobot *support vector* dan evaluasi untuk akurasi sistem.
4. Penerapan algoritma *Binary Decision Tree* dan *Support Vector Machine* dalam pembuatan sistem.

3.6 Analisa dan Pengujian Sistem

Pada tahap ini dilakukan pengujian sistem untuk menunjukkan bahwa perangkat lunak telah mampu bekerja sesuai dengan spesifikasi dari kebutuhan yang telah dituliskan. Proses pengujian dilakukan dengan menguji untuk membandingkan data hasil pengolahan perhitungan manual dengan perhitungan yang dilakukan oleh sistem yaitu pengujian akurasi. Selain itu pengujian juga dilakukan untuk melihat akurasi dari parameter *input* dan data pemeriksaan EKG.

3.7 Penarikan Kesimpulan dan Saran

Pengambilan kesimpulan dilakukan setelah semua tahapan perancangan, implementasi sistem yang dibuat selesai dilaksanakan dan didasarkan pada kesesuaian antara teori dengan penerapan. Kesimpulan dibuat untuk memberikan jawaban terhadap rumusan masalah tentang bagaimana proses perhitungan atau komputasi dari algoritma *BDT-SVM* pada kondisi detak jantuk berdasarkan hasil EKG (Elektrokardiografi) dan bagaimana tingkat akurasi dari sistem. Pembuatan saran merupakan tahapan akhir dalam penyusunan penelitian ini. Saran ditujukan untuk memperbaiki berbagai kesalahan yang terjadi selama penulisan dan dalam proses penerapan algoritma.

BAB 4 PERANCANGAN

Bab ini menjelasakan formulasi permasalahan, langkah-langkah algoritma *Binary Decision Tree Support Vector Machine (BDT-SVM)*. Langkah-langkah penyelesaian pengklasifikasian detak jantung berdasarkan hasil pemeriksaan Elektrokardiogram (EKG) menggunakan algoritma *Binary Decision Tree Support Vector Machine*, perhitungan manual, dan perancangan antarmuka sistem.

4.1 Formulasi Permasalahan

Permasalahan yang diselesaikan adalah klasifikasi data detak jantung yang terdiri dari empat kelas yaitu normal, *atrial fibrilation, pvc bigeminy, dan ventricular tachycardia*. Nilai yang dimasukkan pengguna yaitu nilai dari hasil pemeriksaan EKG untuk pemeriksaan dengan durasi waktu sekitar 10 detik. Untuk parameter terdiri dari dua pemeriksaan yaitu *MLII* dan *VI* dimana setiap pemeriksaan tersebut terdiri dari 10 detik yang akan menjadi fitur dari penelitian ini. Dari penelitian ini menggunakan fitur yaitu 7202 (20 detik yang terdiri dari pemeriksaan *MLII* dan *VI*), 3601 (10 detik dari setiap pemeriksaan *MLII* atau *VI*), dan 2161 (6 detik dari setiap pemeriksaan *MLII* dan *VI*). Dari nilai masukan tersebut akan diklasifikasikan ke dalam empat kelas tersebut.

Masukan sistem antara lain data detak jantung dengan format *.csv* dan parameter perhitungan yaitu data nilai dari dua sisi pemeriksaan yaitu *MLII* dan *VI*. Kemudian masuk ke tahap klasifikasi dimana terdiri dari 2 tahap. Tahap pertama adalah proses membentuk pohon biner dengan algoritma *Binary Decision Tree*. Tahap kedua adalah proses klasifikasi dengan algoritma *SVM*. Jumlah Level yang terdapat pada perhitungan dengan *SVM* didapatkan dari proses penyusunan pohon biner dengan konsep *BDT*. Keluaran dari hasil tersebut adalah hasil pengklasifikasian data hasil EKG.

Adapun jumlah sampel yang digunakan untuk perhitungan manual data detak jantung berdasarkan hasil pemeriksaan Elektrokardiogram yang digunakan pada penelitian ini berjumlah 140 data. Di bawah ini pada Tabel 4.1 merupakan data sampel yang digunakan.

Tabel 4.1 Data Detak Jantung Hasil EKG.

Data Ke-	EKG			Kelas	Kode Kelas
	Waktu (hh:mm:ss.mmm)	*MLII*	*VI*		
1	'17:35.000'	-0.25	-0.06	Normal	1
	'17:35.003'	-0.245	-0.06		
	'17:35.006'	-0.225	-0.075		
	17:35.008'	-0.235	-0.07		
	'17:35.011'	-0.245	-0.075		
2	'1:42.000'	-0.165	-0.01	Normal	1

	'1:42.003'	-0.175	-0.025		
	'1:42.006'	-0.18	-0.035		
	'1:42.008'	-0.18	-0.02		
	'1:42.011'	-0.18	-0.015		
3	'0:40.000'	-0.31	0.125	Normal	1
	'0:40.003'	-0.28	0.145		
	'0:40.006'	-0.275	0.125		
	'0:40.008'	-0.27	0.125		
	'0:40.011'	-0.28	0.125		
4	'26:09.000'	-0.485	-0.72	Normal	1
	'26:09.003'	-0.46	-0.715		
	'26:09.006'	-0.45	-0.73		
	'26:09.008'	-0.455	-0.73		
	'26:09.011'	-0.465	-0.74		
5	'16:20.000'	-0.185	-0.01	*Atrial Fibrilation*	2
	'16:20.003'	-0.175	-0.005		
	'16:20.006'	-0.16	0		
	'16:20.008'	-0.155	0.01		
	'16:20.011'	-0.15	-0.01		
6	'22:47.000'	-0.155	0.36	*Atrial Fibrilation*	2
	'22:47.003'	-0.185	0.34		
	'22:47.006'	-0.225	0.31		
	'22:47.008'	-0.245	0.305		
	'22:47.011'	-0.265	0.295		
7	'22:27.000'	0.11	0.1	*Atrial Fibrilation*	2
	'22:27.003'	0.125	0.1		
	'22:27.006'	0.115	0.095		
	'22:27.008'	0.115	0.09		
	'22:27.011'	0.135	0.095		
8	'17:32.000'	-0.39	0.145	*Atrial Fibrilation*	2
	'17:32.003'	-0.38	0.12		
	'17:32.006'	-0.395	0.105		
	'17:32.008'	-0.385	0.08		

	'17:32.011'	-0.375	0.065		
9	'12:27.000'	0.175	0.065	*Atrial Fibrilation*	2
	'12:27.003'	0.18	0.105		
	'12:27.006'	0.195	0.125		
	'12:27.008'	0.19	0.14		
	'12:27.011'	0.2	0.16		
10	'0:00.000'	-0.215	0.095	*PVC Bigeminy*	3
	'0:00.003'	-0.215	0.095		
	'0:00.006'	-0.215	0.095		
	'0:00.008'	-0.215	0.095		
	'0:00.011'	-0.215	0.095		
11	'2:41.000'	0.285	-0.11	*PVC Bigeminy*	3
	'2:41.003'	0.295	-0.09		
	'2:41.006'	0.305	-0.08		
	'2:41.008'	0.305	-0.095		
	'2:41.011'	0.305	-0.105		
12	'0:20.000'	-0.54	0.11	*PVC Bigeminy*	3
	'0:20.003'	-0.515	0.095		
	'0:20.006'	-0.5	0.075		
	'0:20.008'	-0.495	0.055		
	'0:20.011'	-0.49	0.055		
13	'24:18.000'	-0.25	-0.4	*Ventricular Tachycardia*	4
	'24:18.003'	-0.14	-0.42		
	'24:18.006'	-0.005	-0.44		
	'24:18.008'	0.16	-0.47		
	'24:18.011'	0.355	-0.52		
14	'17:30.000'	0.05	0.695	*Ventricular Tachycardia*	4
	'17:30.003'	0.125	0.655		
	'17:30.006'	0.195	0.615		
	'17:30.008'	0.255	0.56		
	'17:30.011'	0.32	0.515		
15	'5:38.000'	-0.565	0.675	*Ventricular Tachycardia*	4
	'5:38.003'	-0.59	0.705		

	'5:38.006'	-0.58	0.735		
	'5:38.008'	-0.58	0.745		
	'5:38.011'	-0.59	0.735		
16	'7:05.000'	0.13	0.26	*Ventricular Tachycardia*	4
	'7:05.003'	0.12	0.265		
	'7:05.006'	0.115	0.28		
	'7:05.008'	0.135	0.28		
	'7:05.011'	0.175	0.3		

Sumber : *MIT-BIH* Aritmia (2010).

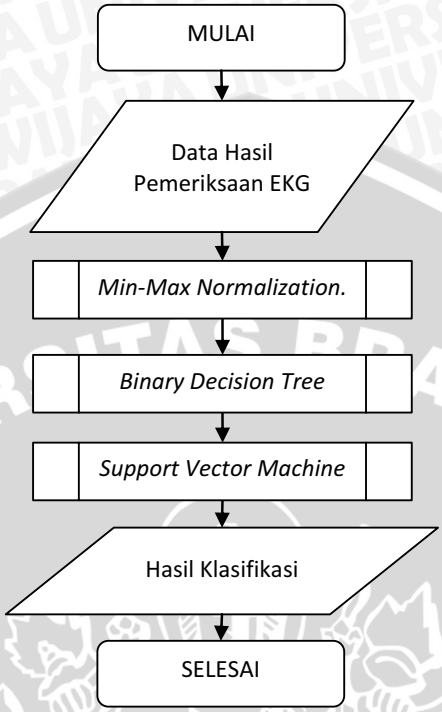
Dari data-data detak jantung di atas telah dikonsultasikan ke pakar yaitu dokter spesialis jantung dan pembuluh darah untuk validasi data yang didapatkan di *MIT-BIH*.

4.2 Langkah-Langkah Algoritma *BDT-SVM*

Siklus algoritma *Binary Decision Tee-Support Vector Machine* merupakan urutan penyelesaian masalah menggunakan teknik klasifikasi. Untuk proses klasifikasi dengan menggunakan algoritma *Support Vector Machine* yang merupakan algoritma untuk mencari *hyperplane* pemisah terbaik atau yang paling optimal. Kelemahan *Support Vector Machine* yaitu hanya bisa dengan *binary class* maka digabungkan dengan *Binary Decision Tree* agar bisa *multi-class*. Algoritma ini merupakan algoritma yang baik dilakukan karena memiliki keuntungan yaitu dapat menghitung dengan dimensi data yang tinggi atau banyak, selain itu juga memiliki arsitektur perhitungan yang efisien dan memiliki akurasi klasifikasi yang tinggi. Proses normalisasi data menggunakan metode *Min-Max Normalization,* proses penentuan jarak setiap kelas dengan menggunakan *Euclidean Distance* dan penentuan pusat data / *data center* menggunakan *Gravity Center.*

Diagram alir proses sistem klasifikasi data detak jantung hasil pemeriksaan Elektrokardiografi (EKG) secara keseluruhan ditunjukkan pada Gambar 4.1 di bawah ini. Pada gambar tersebut dijelaskan bagaimana cara kerja secara umum sistem tersebut. *Input* data diperoleh dari data hasil detak jantung dari hasil Pemeriksaan Elektrokardiografi (EKG) tahun 2010. Data tersebut telah divalidasi oleh pakar yaitu Dokter Spesialis Jantung dan Pembuluh Darah. Data yang telah dimasukkan akan di normalisasi menggunakan metode *Min-Max Normalization*. kemudian akan menuju ke proses pembentukan kelas dengan metode *Binary Decision Tree*, lalu untuk proses klasifikasi dengan menggunakan metode *Support Vector Machine (SVM)* dari proses *training* data hingga *testing* data. Jika kaidah telah sesuai maka akan mendapatkan hasil klasifikasi. *Output* yang dihasilkan dari sistem ini adalah pasien yang mengalami kondisi detak jantung yang Normal, kondisi pasien terkena Aritmia yang terdiri dari 3 yaitu *PVC Bigeminy*, *Ventricular*

Tachycardia, dan *Atrial Fibrilation*. Untuk lebih jelasnya proses yang telah dijelaskan telah digambarkan pada Gambar 4.1 di bawah ini.



Gambar 4.1 Diagram Alir Proses Berjalannya SIstem.

4.2.1 Proses Normalisasi Fitur

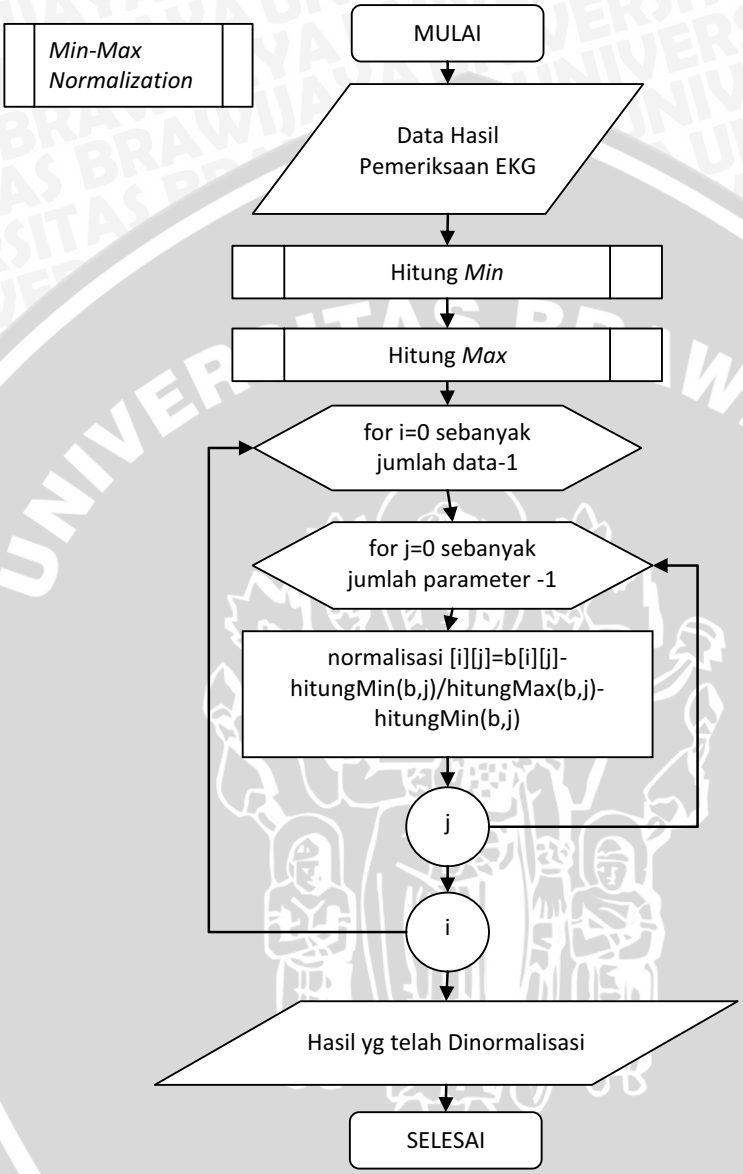
Normalisasi data merupakan proses transformasi data atau mengubah data agar menjadi lebih kecil dengan rentang antara 0 sampai dengan 1. Proses ini dilakukan agar nilai antar fitur memiliki selisih yang tidak jauh atau kisaran nilai yang kecil sehingga mendapatkan kualitas data yang baik. Pada penelitian ini, proses normalisasi data menggunakan metode *Min-Max Normalization*. Diagram alir proses normalisasi data ditunjukkan pada Gambar 4.2 di bawah ini.

Langkah-langkah untuk menghitung normalisasi setiap fitur berdasarkan diagram alir di bawah yaitu sebagai berikut.

1. Sistem menerima masukan berupa data hasil rekam jantung EKG yang terdiri dari gabungan antara data latih dan data uji.
2. Mencari nilai minimal dan maksimal dari data yang ada pada semua parameter berdasarkan parameter. Untuk proses mencari nilai minimal dan maksimal akan dijelaskan lebih mendalam pada sub bab di bawah ini.
3. Perulangan sebanyak jumlah data dan jumlah parameter untuk menuju ke proses menghitung nilai normalisasi pada setiap data untuk setiap parameter yang ada berdasarkan persamaan pada Persamaan 2.1. Nilai pada parameter *b[i][j]* merupakan data ke-*X* sesuai dengan baris ke-*i* dan kolom ke-*j*. Kemudian dikurangi dengan hasil dari *min.* Setelah itu akan dibagi dengan

nilai *range* yang didapat dari pengurangan antara nilai maksimal dengan nilai minimal.

4. Sistem akan menampilkan hasil yang telah dinormalisasi.



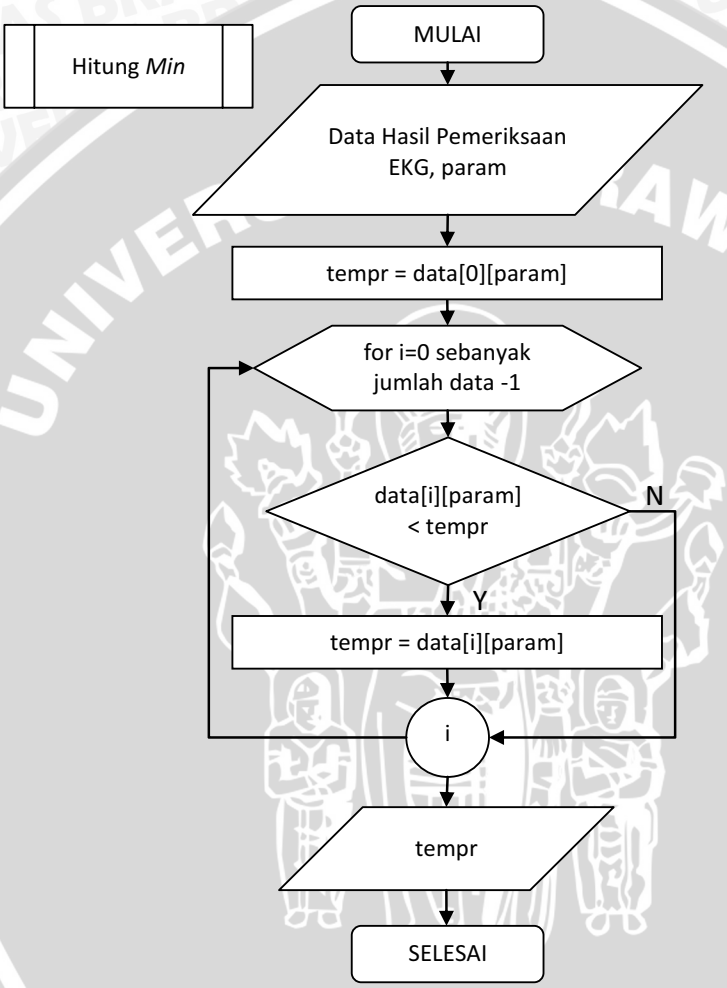
Gambar 4.2 Diagram Alir Proses Normalisasi.

4.2.1.1. Proses Hitung *Min*

Proses hitung *min* adalah suatu proses untuk mencari nilai minimal atau nilai yang terkecil dari setiap fitur atau parameter berdasarkan masukan. Nilai minimal ini yang digunakan dalam proses selanjutnya untuk melakukan normalisasi data. Diagram alir proses hitung *min* data ditunjukkan pada Gambar 4.3 di bawah ini. Langkah-langkah untuk mencari nilai *min* berdasarkan fitur atau parameter *ya*itu sebagai berikut.

1. Sistem menerima masukan berupa data hasil rekam jantung EKG.

2. Inisialisasi variabel *tempr = data[0][param]*. Berarti dari data itu dilihat parameter nya berdasarkan *param* dan baris = 0 berarti bahwa yang diubah atau nilai yang disimpan adalah nilai *param* sebagai nilai awal.
3. Perulangan berdasarkan banyaknya jumlah data dan terdapat percabangan jika data pada baris ke-*i* dan kolom ke-*param* lebih kecil dari nilai *tempr* maka nilai *tempr = data[i][param]* dan jika tidak akan keluar dari perulangan.
4. Menampilkan hasil *min*.



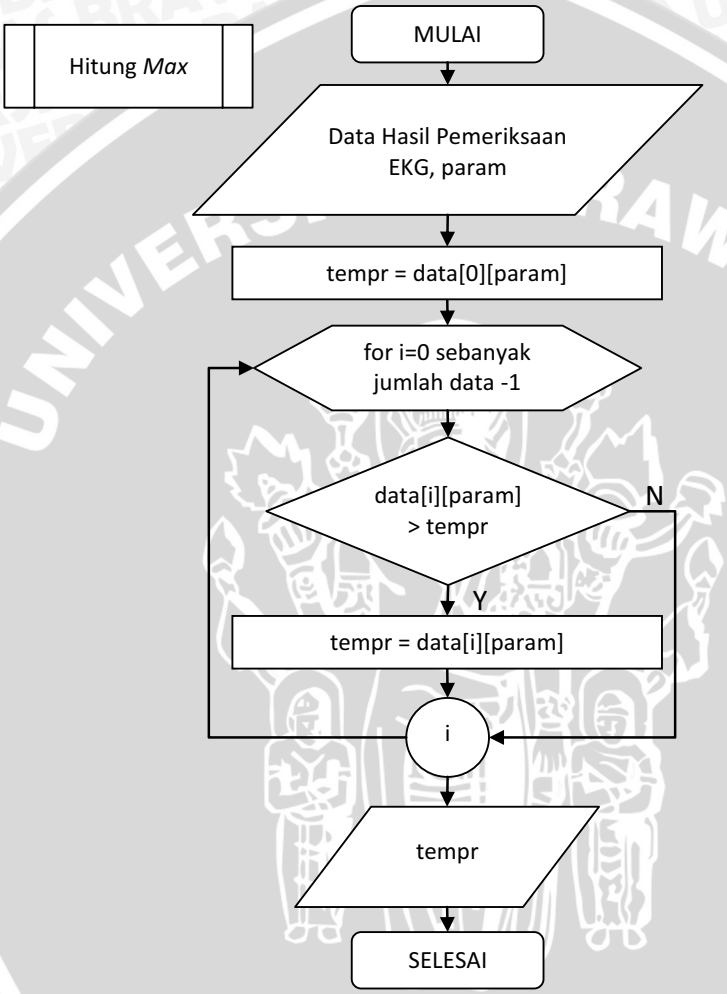
Gambar 4.3 Diagram Alir Proses Hitung *Min*.

4.2.1.2. Proses Hitung *Max*

Proses hitung *max* adalah suatu proses untuk mencari nilai maksimal atau nilai yang terbesar dari setiap fitur atau parameter berdasarkan masukan. Nilai maksimal ini yang digunakan dalam proses selanjutnya untuk melakukan normalisasi data. Diagram alir proses hitung *max* data ditunjukkan pada Gambar 4.4 di bawah ini. Langkah-langkah untuk mencari nilai *min* berdasarkan fitur atau parameter *ya*itu sebagai berikut.

1. Sistem menerima masukan berupa data hasil rekam jantung EKG.

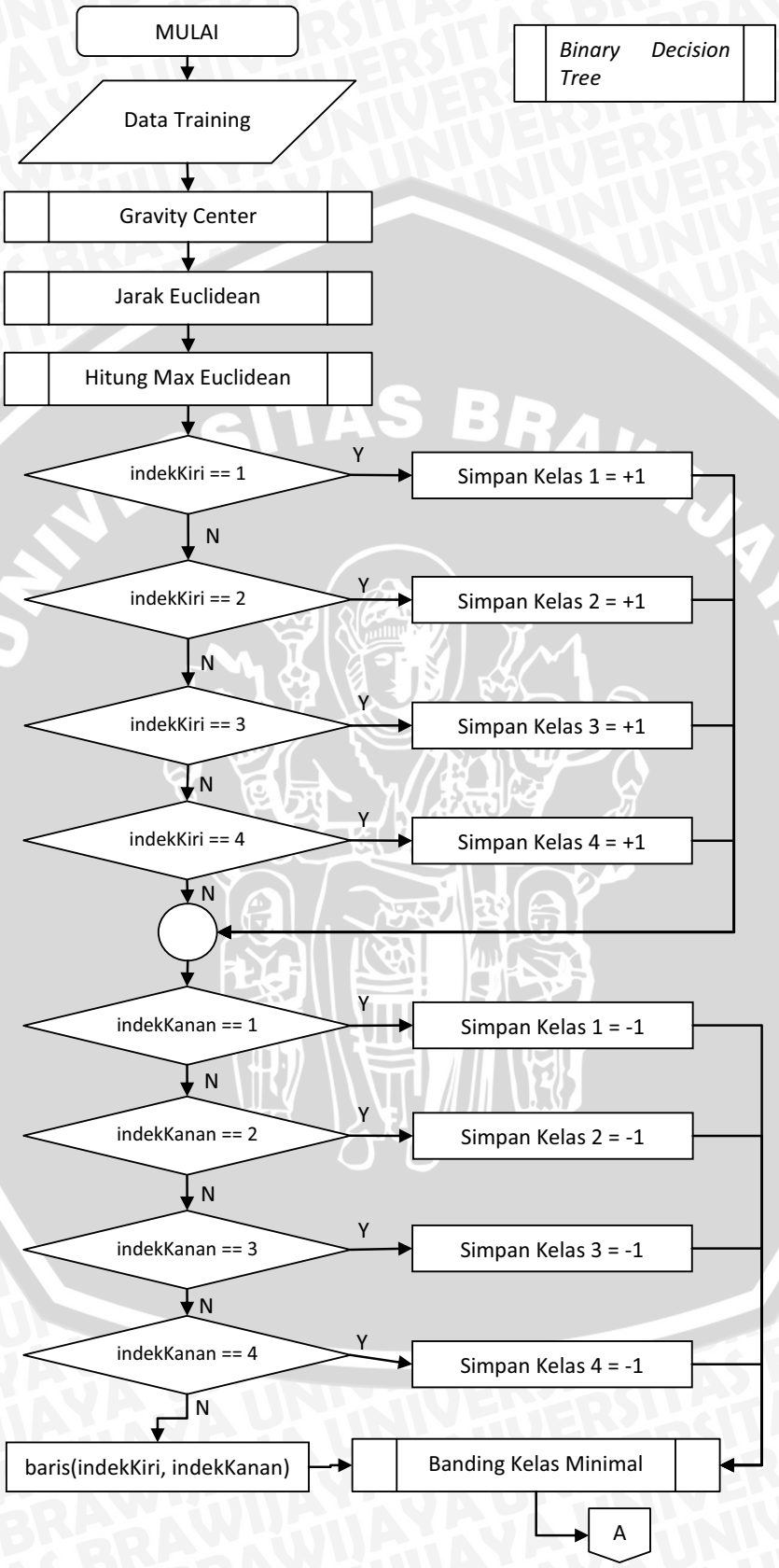
2. Inisialisasi variabel *tempr = data[0][param]*. Berarti dari data itu dilihat parameter nya berdasarkan *param* dan baris = 0 berarti bahwa yang diubah atau nilai yang disimpan adalah nilai *param* sebagai nilai awal.
3. Perulangan berdasarkan banyaknya jumlah data dan terdapat percabangan jika data pada baris ke-*i* dan kolom ke-*param* lebih besar dari nilai *tempr* maka nilai *tempr = data[i][param]* dan jika tidak akan keluar dari perulangan.
4. Menampilkan hasil *max.*

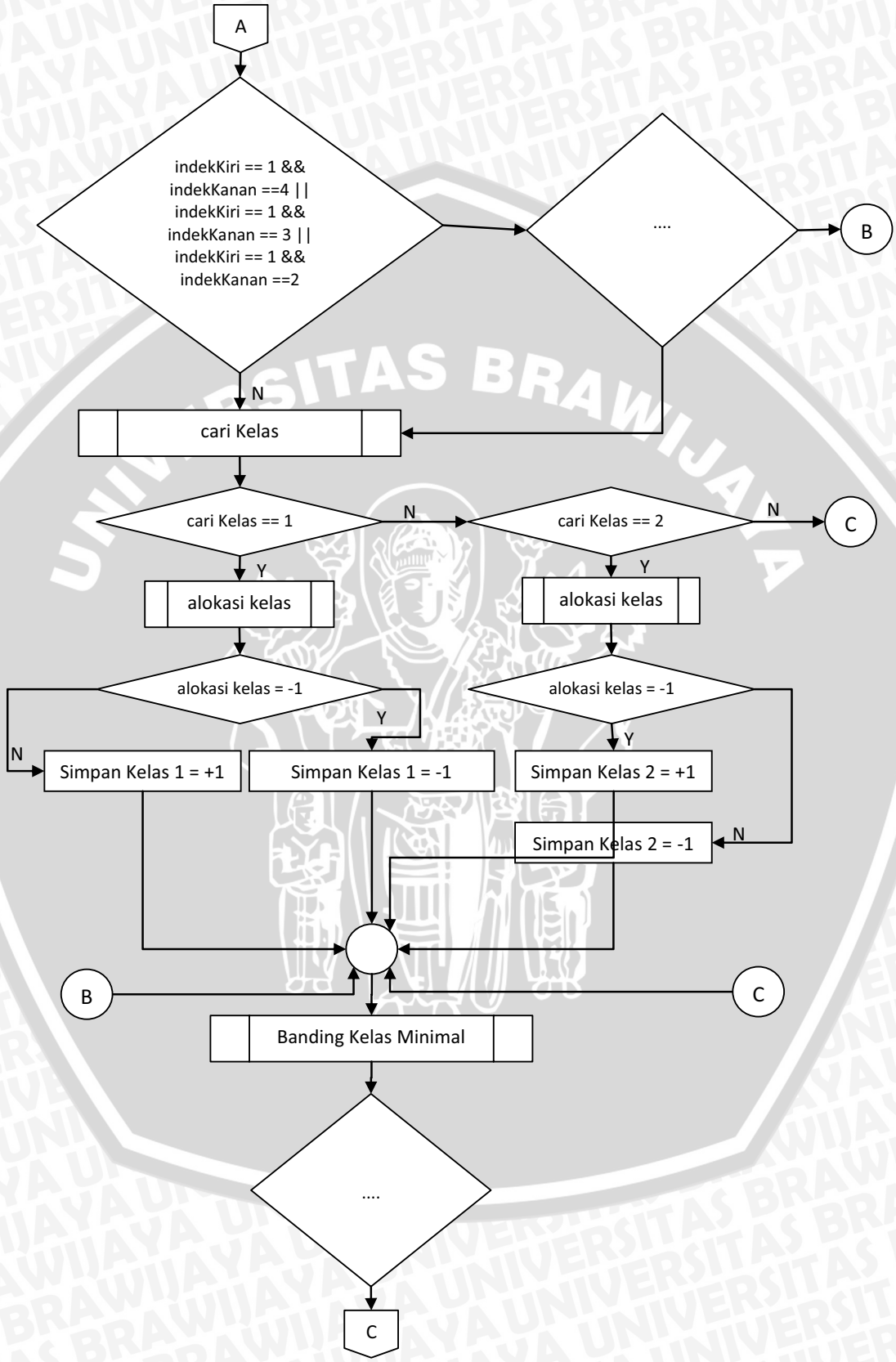


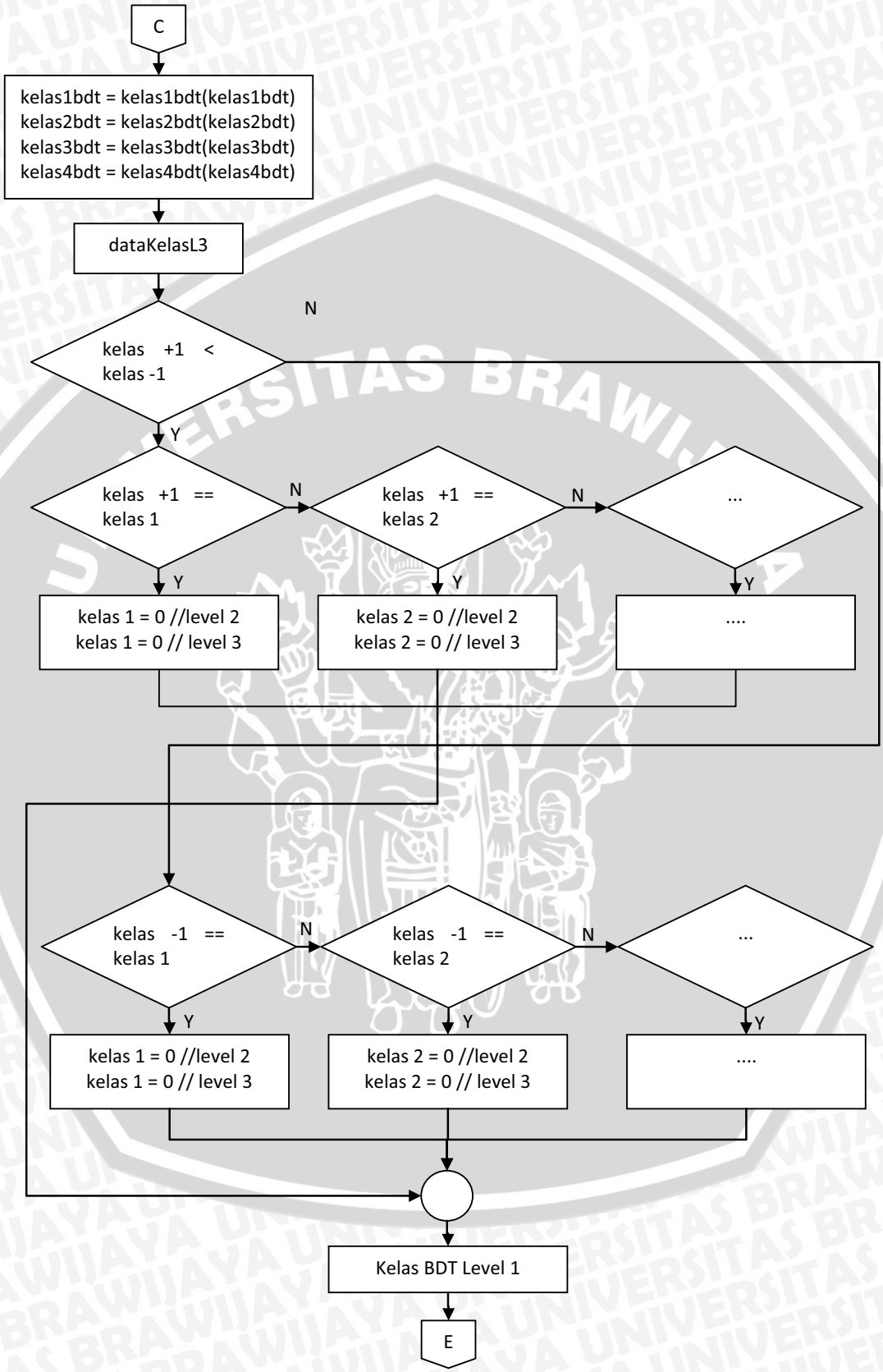
Gambar 4.4 Diagram Alir Proses Hitung *Max*.

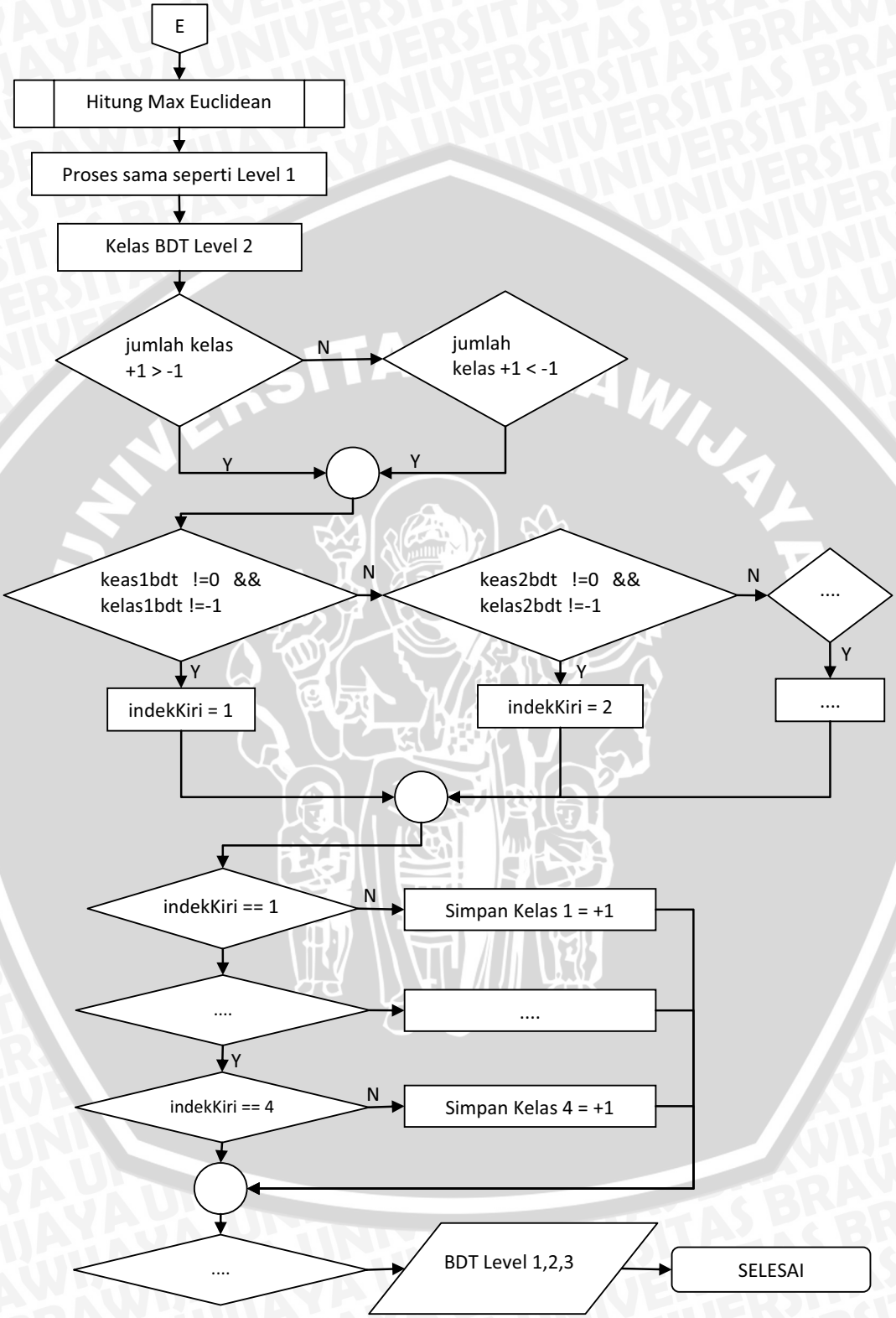
4.2.2 Proses *Binary Decision Tree*

Proses *Binary Decision Tree* adalah proses yang menggambarkan tahap-tahap perhitungan hingga mendapatkan hasil *output* yang merupakan hasil penentuan kelas. Untuk *flow chart* dapat dilihat pada Gambar 4.5 di bawah ini. Proses adalah dengan mencari nilai titik tengah kemudian akan menghitung jarak euclidean dan dicari jarak maksimumnya. Dari jarak maksimum tersebut akan membagi kelas menjadi 2 kategori yaitu positif maupun negatif. Kemudian dari *node* yang masih bisa diturunkan akan dicari nilai minimal dari jarak euclidean dan mencari nilai tersebut berada di kelas 1, 2, 3, atau 4 hingga level terakhir.


MULAI
Data Training
Gravity Center
Jarak Euclidean
Hitung Max Euclidean
indekKiri == 1
Y
Simpan Kelas 1 = +1
N
indekKiri == 2
Y
Simpan Kelas 2 = +1
N
indekKiri == 3
Y
Simpan Kelas 3 = +1
N
indekKiri == 4
Y
Simpan Kelas 4 = +1
N
indekKanan == 1
Y
Simpan Kelas 1 = -1
N
indekKanan == 2
Y
Simpan Kelas 2 = -1
N
indekKanan == 3
Y
Simpan Kelas 3 = -1
N
indekKanan == 4
Y
Simpan Kelas 4 = -1
N
baris(indekKiri, indekKanan)
Banding Kelas Minimal
A
Binary Decision Tree


A
indekKiri == 1 &&
indekKanan ==4 ||
indekKiri == 1 &&
indekKanan == 3 ||
indekKiri == 1 &&
indekKanan ==2
....
B
N
cari Kelas
cari Kelas == 1
N
cari Kelas == 2
N
C
Y
alokasi kelas
Y
alokasi kelas
alokasi kelas = -1
alokasi kelas = -1
Y
Y
N
Simpan Kelas 1 = +1
Simpan Kelas 1 = -1
Simpan Kelas 2 = +1
Simpan Kelas 2 = -1
N
B
C
Banding Kelas Minimal
....
C


C
kelas1bdt = kelas1bdt(kelas1bdt)
kelas2bdt = kelas2bdt(kelas2bdt)
kelas3bdt = kelas3bdt(kelas3bdt)
kelas4bdt = kelas4bdt(kelas4bdt)
dataKelasL3
N
kelas +1 < kelas -1
Y
kelas +1 == kelas 1
N
kelas +1 == kelas 2
N
...
Y
Y
Y
kelas 1 = 0 //level 2
kelas 1 = 0 // level 3
kelas 2 = 0 //level 2
kelas 2 = 0 // level 3
....
kelas -1 == kelas 1
N
kelas -1 == kelas 2
N
...
Y
Y
Y
kelas 1 = 0 //level 2
kelas 1 = 0 // level 3
kelas 2 = 0 //level 2
kelas 2 = 0 // level 3
....
Kelas BDT Level 1
E



Gambar 4.5 Diagram Alir Proses *Binary Decision Tree*.

Untuk melihat *flow chart* pada proses *binary decision tree* ini secara keseluruhan bisa dilihat pada Lampiran E. Dimana tahapan dalam proses penentuan hasil penentuan data uji ini yaitu sebagai berikut.

1. Karena terdapat empat kelas, maka terdapat 3 Level perhitungan.
2. Sistem akan memberikan inputan data *training* yang telah dinormalisasi.
3. Sistem akan menghitung *Gravity Center,* kemudian *Jarak Euclidean,* dan kemudian akan mencari nilai maksimal dari jarak euclidean.
4. Menuju ke percabangan jika hasil *max euclidean* terdapat pada *indekKiri == 1* maka akan menyimpan hasil kelas 1 = +1 hingga pada *indekKiri == 4*. Begitu pula sebaliknya pada *indekKanan* yang akan menuju ke kelas -1.
5. Kemudian akan membandingkan kelas dengan jarak euclidean minimal dari jarak *euclidean* yang telah didapatkan.
6. Kemudian akan menuju ke percabangan dengan kondisi jika *indekKiri == 1* atau 2 atau 3 atau 4 sama halnya dengan *indekKanan*, maka akan menjalankan fungsi cari kelas untuk mencari kelas dari jarak euclidean minimal.
7. Kemudian akan menempatkan apakah alokasi nya memasuki kelas positif atau kelas negatif. Lalu kemudian akan membandingkan kelas minimal lagi dan mencari kelas nya dan alokasinya hingga mencapai simpul paling bawah dari level BDT.
8. Hasilnya akan dimasukkan ke parameter untuk setiap kelas sehingga terdapat 4 parameter yaitu *kelas1bdt*, *kelas2bdt*, *kelas3bdt*, dan *kelas4bdt*. Kemudian akan digabungkan menjadi satu parameter yaitu *dataKelasL3*. *dataKelasL3* ini adalah parameter yang menyimpan hasil kelas dari BDT untuk level 1.
9. Langkah ini akan terus berulang sampai *node* sudah tidak memiliki anak lagi yaitu hingga level 3.

4.2.2.1. Proses Gravity Center

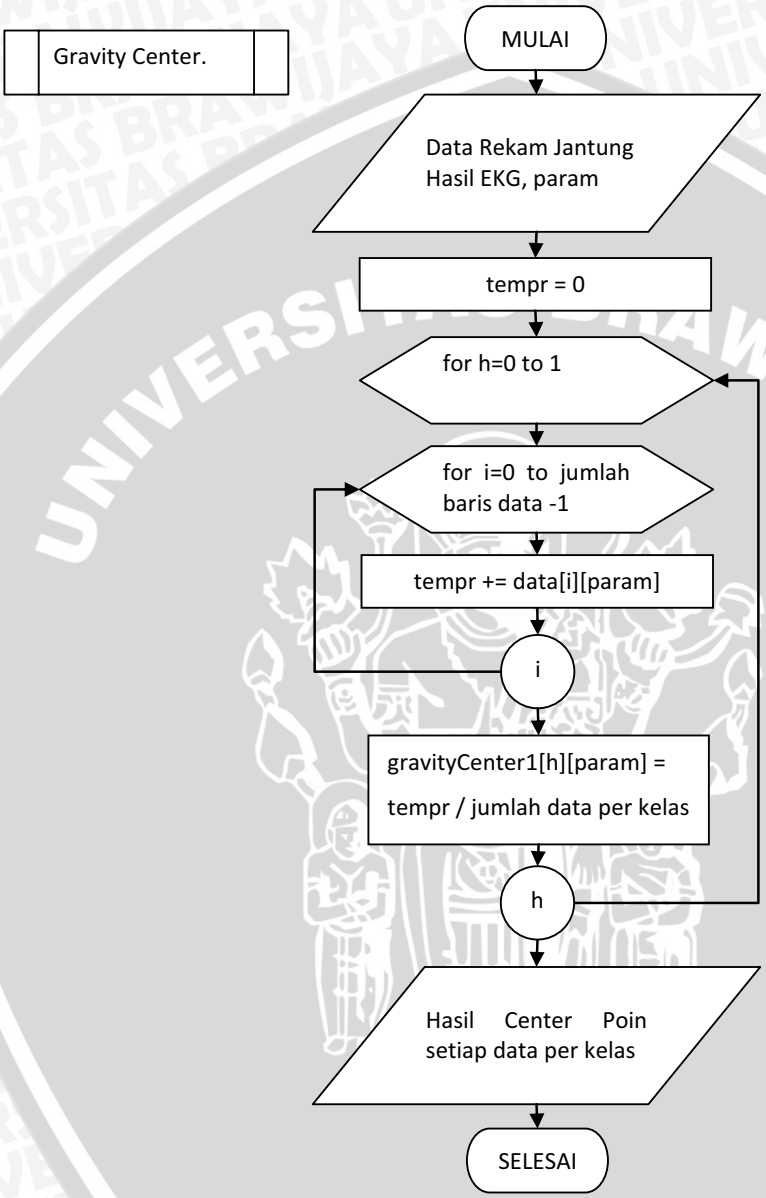
Gravity center adalah titik pusat dari setiap kelas untuk semua data. Untuk mendapatkan titik pusat dari setiap kelas, dapat dihitung dengan mencari rata-rata setiap parameter atau fitur dari masing-masing data pada setiap kelas. Persamaan untuk menghitung gravity center dapat dilihat pada Persamaan 2.20 pada bab sebelumnya.

Langkah-langkah untuk menghitung nilai titik tengah untuk setiap fitur atau *gravity center* berdasarkan diagram alir di bawah ini yaitu sebagai berikut.

1. Sistem menerima masukan berupa data hasil rekam jantung EKG.
2. Kemudian akan menginisialisasi parameter *tempr = 0*.
3. Perulangan yang pertama yaitu *h=0* hingga *h=1* sebagai jumlah baris dari hasil *gravity center* yang berjumlah 1 untuk setiap kelas. Sehingga matriks akan menjadi 4 baris untuk 4 kelas.
4. Kemudian perulangan yang kedua hingga banyaknya jumlah data. Kemudian akan menjumlahkan nilai dari setiap baris.
5. Sistem akan menghitung nilai rata-rata data setiap fitur pada setiap kelas yang ada menggunakan Persamaan 2.18.

6. Keluaran sistem adalah hasil *gravity center* atau titik tengah data pada setiap kelas untuk setiap parameter atau fitur.

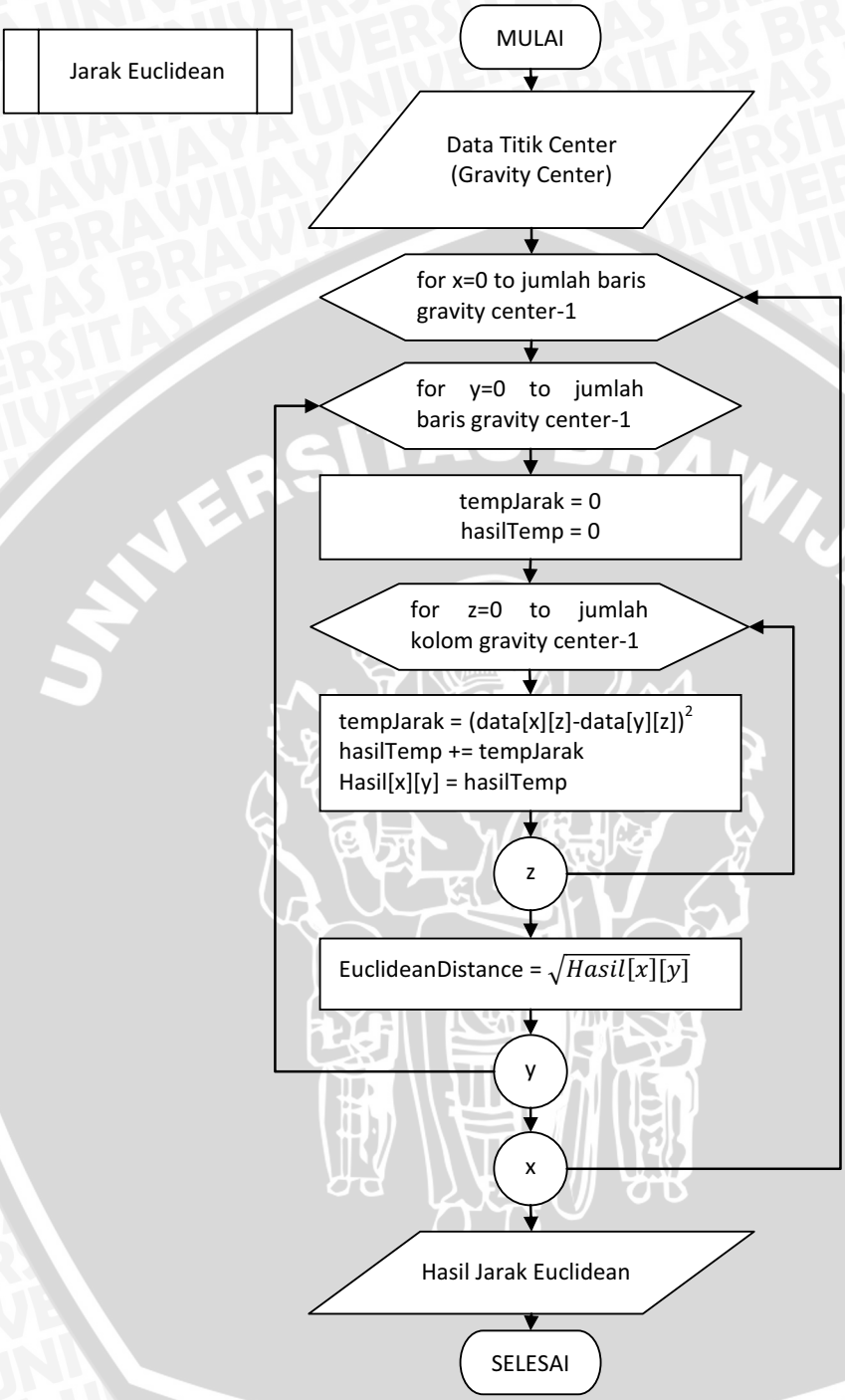
Diagram alir proses mendapatkan *gravity center* untuk kelas ke-1 bisa dilihat pada Gambar 4.6 di bawah ini.



Gambar 4.6 Diagram Alir Proses Gravity Center.

4.2.2.2. Proses Jarak *Euclidean*

Jarak *Euclidean* adalah salah satu jarak yang sering digunakan. Jarak ini berfungsi untuk mencari jarak dari titik-titik data dan mendapatkan jarak terdekat maupun terjauh dari data-data tersebut. Dalam perhitungan ini, jarak ini berguna untuk membentuk *decision tree* dan pembentukan kelas. Diagram alir proses mendapatkan jarak *euclidean* dilihat pada Gambar 4.7 di bawah ini.



Gambar 4.7 Diagram Alir Proses Jarak Euclidean.

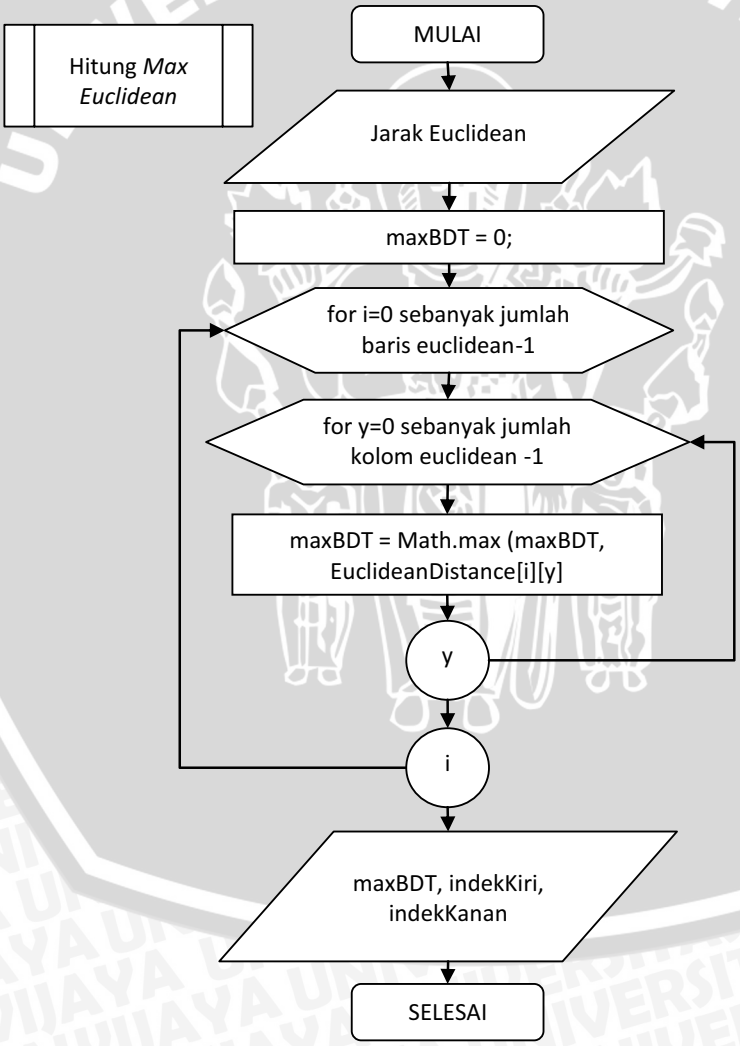
Langkah-langkah untuk menghitung jarak *euclidean* berdasarkan diagram alir di bawah ini yaitu sebagai berikut.

1. Sistem menerima masukan berupa hasil matriks dari *Gravity Center*.
2. Perulangan pertama yaitu perulangan hingga mencapai baris jumlah dari *gravity center* sama halnya sengan perulangan yang kedua.

3. Untuk perulangan yang ketiga yaitu hingga jumlah kolom atau parameter pada *gravity center*.
4. Inisialisasi awal yaitu nilai *tempr = 0* dan *hasilTemp = 0*.
5. Sistem akan menghitung setiap baris dan kolom kemudian mengurangkan setiap data dari *gravity center* tersebut dan hasilnya akan dipangkatkan 2.
6. Kemudian hasil dari proses ke-2 akan diakarkan untuk memperoleh hasil akhir.
7. Keluaran sistem adalah hasil jarak *euclidean* dalam bentuk matriks berdimensi jumlah kelas.

4.2.2.3. Proses Hitung *Max Euclidean*

Proses untuk mencari nilai maksimal dari jarak *euclidean.* Diagram alir proses mendapatkan nilai maksimal dari jarak *euclidean* dilihat pada Gambar 4.8 di bawah ini.



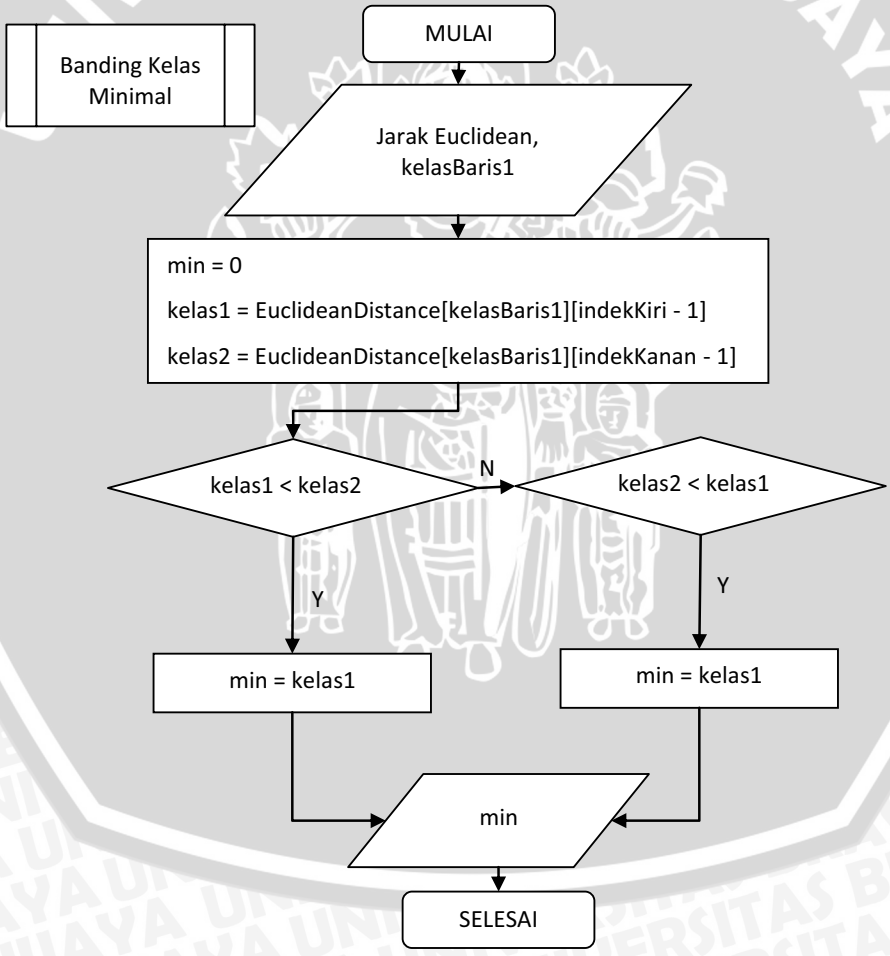
Gambar 4.8 Diagram Alir Proses Hitung *Max Euclidean*.

Langkah-langkah untuk menghitung *max* jarak *euclidean* berdasarkan diagram alir di atas yaitu sebagai berikut.

1. Sistem menerima masukan berupa matriks jarak euclidean. Inisialisasi awal parameter *maxBDT = 0*.
2. Perulangan pertama yaitu perulangan hingga mencapai baris jumlah dari jarak euclidean dan perulangan kedua higga jumlah kolom pada jarak euclidean.
3. Kemudian akan mencari nilai maksimal dari jarak euclidean.
4. Sistem akan menampilkan nilai maksimal jarak euclidean, beserta dengan *indekKanan* dan *indekKiri*.

4.2.2.4. *Proses* Banding Kelas Minimal

Proses untuk membandingkan nilai minimal dari kelas untuk mendapatkan penempatan kelas pada *node* di BDT. Diagram alir dapat dilihat pada Gambar 4.9 di bawah ini.



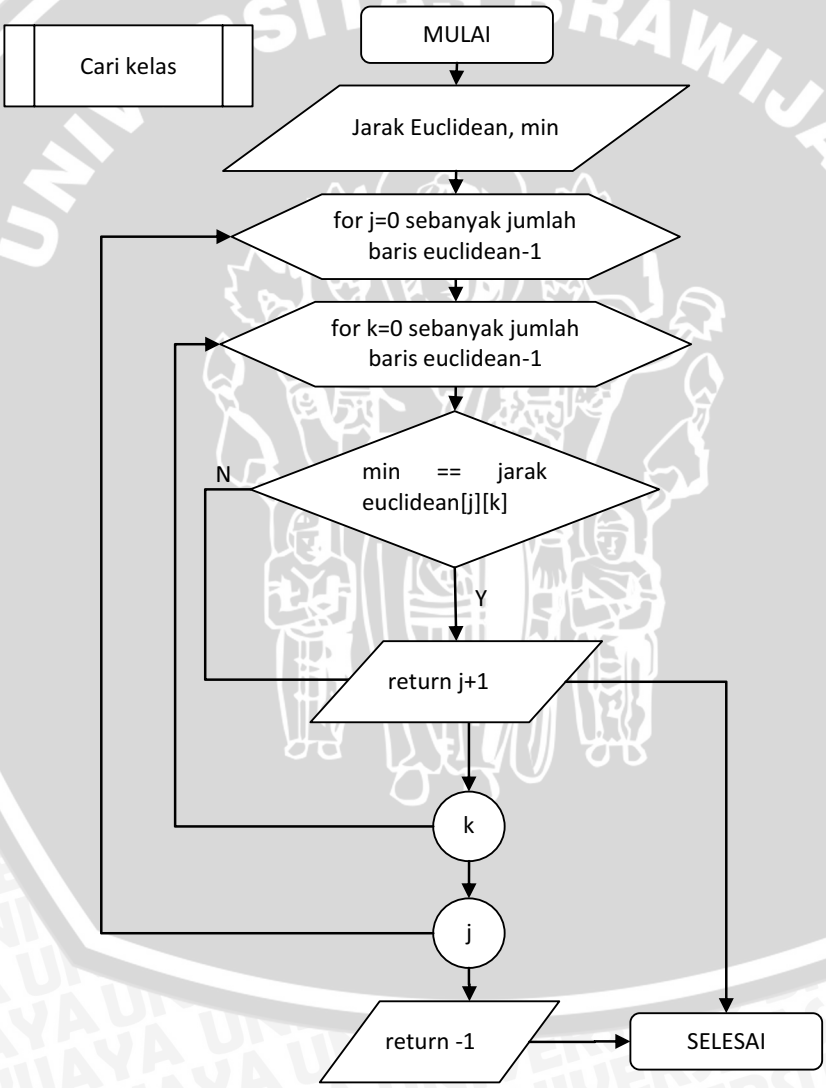
Gambar 4.9 Diagram Alir Proses Banding Kelas Minimal.

Langkah-langkah untuk membandingkan kelas minimal dari jarak *euclidean* berdasarkan diagram alir di atas yaitu sebagai berikut.

1. Sistem menerima masukan berupa matriks jarak euclidean dan *kelasBaris1* yang merupakan nilai pada baris keberapa kelas. Inisialisasi awal parameter min *= 0*.
2. Inisialisasi *kelas1* dan *kelas2* untuk mengetahui dimana nilai yang ingin dibandingkan.
3. Percabangan untuk membandingkan apakah kelas 1 atau kelas 2 memiliki nilai yang paling kecil. Kemudian disimpan di variabel *min.*
4. Sistem akan menampilkan nilai minimal dari jarak euclidean.

4.2.2.5. *Proses* Cari Kelas

Proses untuk mencari kelas dari nilai min yang telah didapatkan. Diagram alir proses mencari kelas dapat dilihat pada Gambar 4.10 di bawah ini.



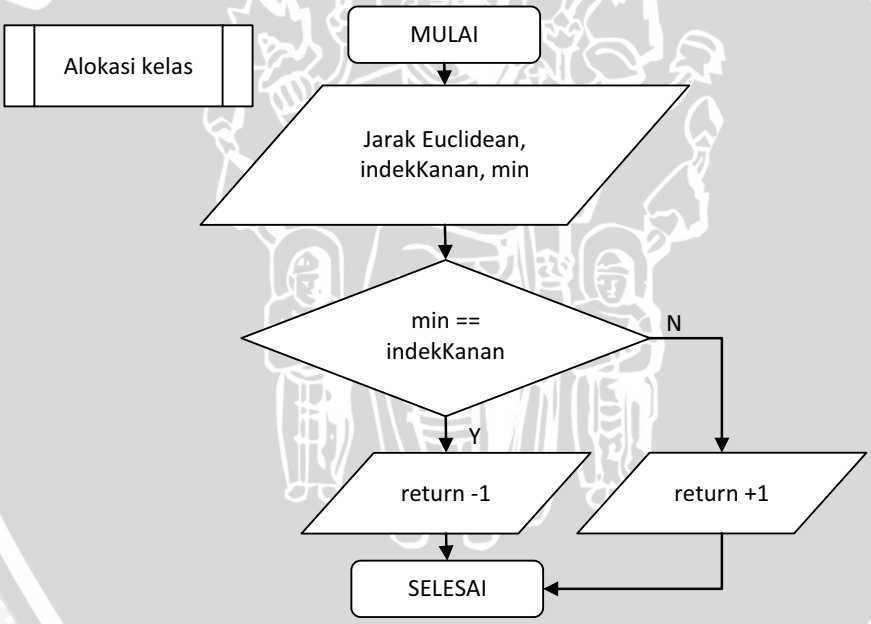
Gambar 4.10 Diagram Alir Proses Cari Kelas.

Langkah-langkah untuk mencari letak indek dari nilai min berdasarkan diagram alir di atas yaitu sebagai berikut.

1. Sistem menerima masukan berupa matriks jarak euclidean dan min yang didapat dari *flowchart* pada Gambar 4.9.
2. Perulangan untuk mencari nilai *min* sampai baris ke jarak euclidean. Kemudian perulanagn kedua untuk mencari nilai min sampai kolom ke jarak euclidean.
3. Percabangan untuk mengetahui apakah nilai yang dicari ketemu. Jika nilai *min* berhasil ketemu maka akan menjumlahkan ke-*j + 1* untuk mengambil jumlah indek dimana nilai *min* berada. Kemudian akan menampilkan hasil dari nilai *min* tersebut. Tetapi Jika nilai *min* tidak berhasil ditemukan maka akan menampilkan -1 yang berarti indek = 0 ke sistem dan akan keluar dari perulangan.

4.2.2.6. Proses Alokasi Kelas

Proses Alokasi kelas adalah suatu proses yang digunakan untuk menentukan apakah suatu data akan memasuki kelas positif atau kelas negatif. Dimana dengan kondisi yaitu jika nilai *min* berada pada sub pohon kanan atau *indekKanan* maka akan masuk ke kelas negatif (-1) sedangkan jika berada di sub pohon kiri atau *indekKiri* maka akan masuk ke kelas positif (+1). Diagram alir yang menjelaskan proses alokasi kelas ini dapat dilihat pada Gambar 4.11 di bawah ini.

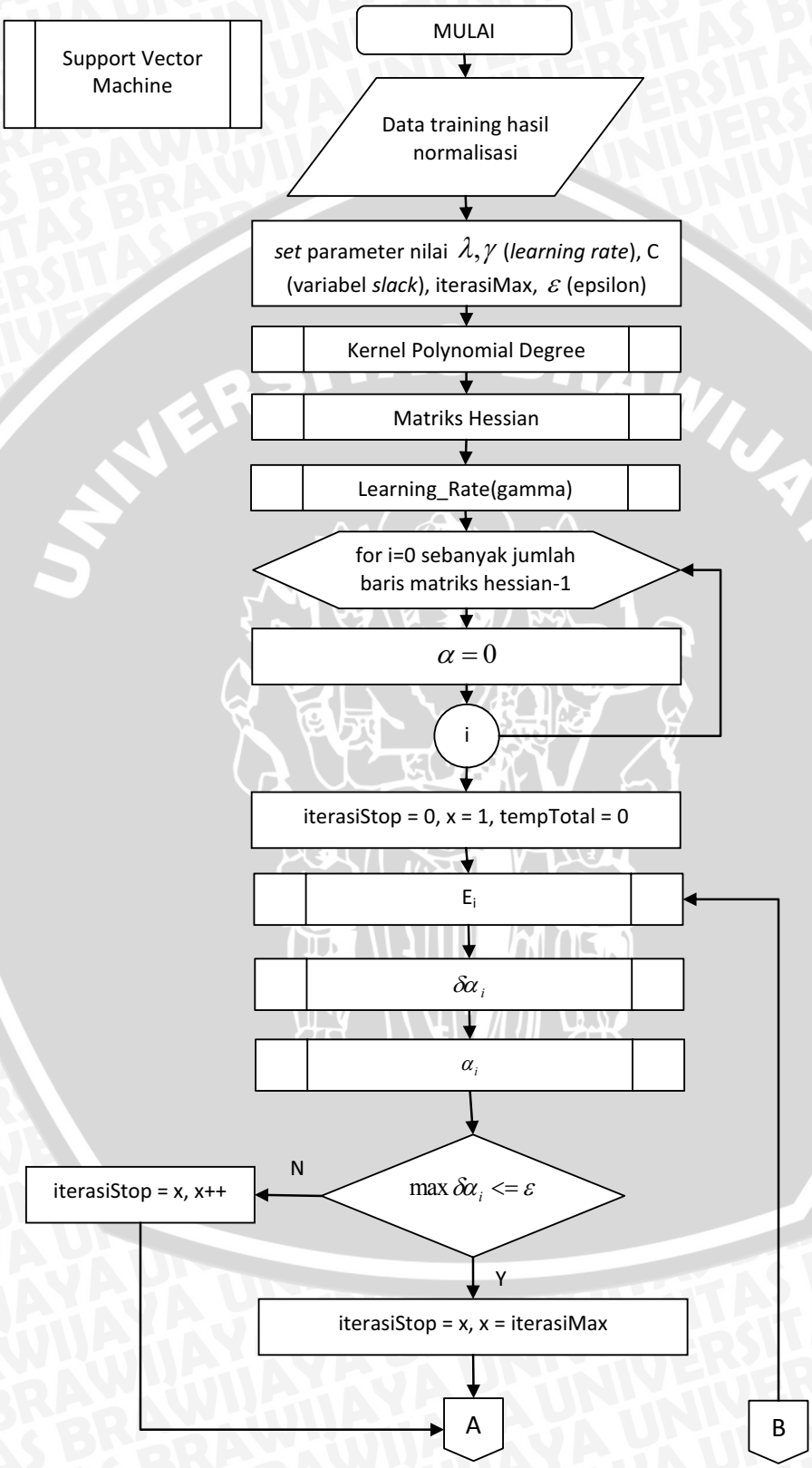


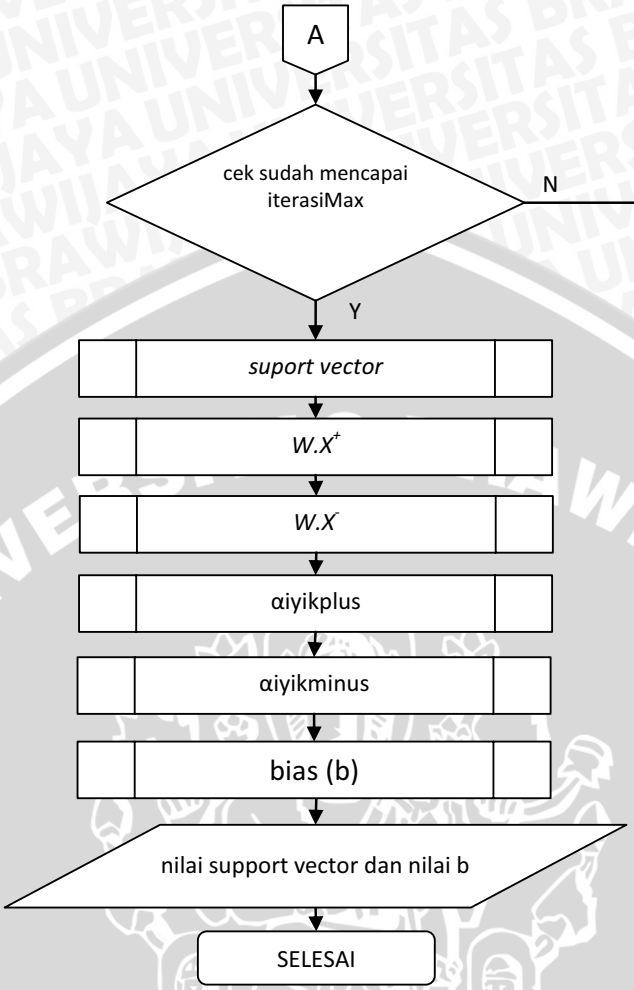
Gambar 4.11 Diagram Alir Proses Alokasi Kelas.

Langkah-langkah untuk mencari letak indek dari nilai min berdasarkan diagram alir di atas yaitu sebagai berikut.

1. Sistem menerima masukan berupa matriks jarak euclidean, nilai *indekKanan* dan *min*.
2. Jika nilai *min == indekKanan* atau nilai *min* berada pada *indekKanan* maka otomatis akan masuk ke kelas negatif. Sebaliknya jika tidak sesuai dengan *indekKanan* maka akan masuk ke kelas positif.

4.2.3 Proses *Support Vector Machine*





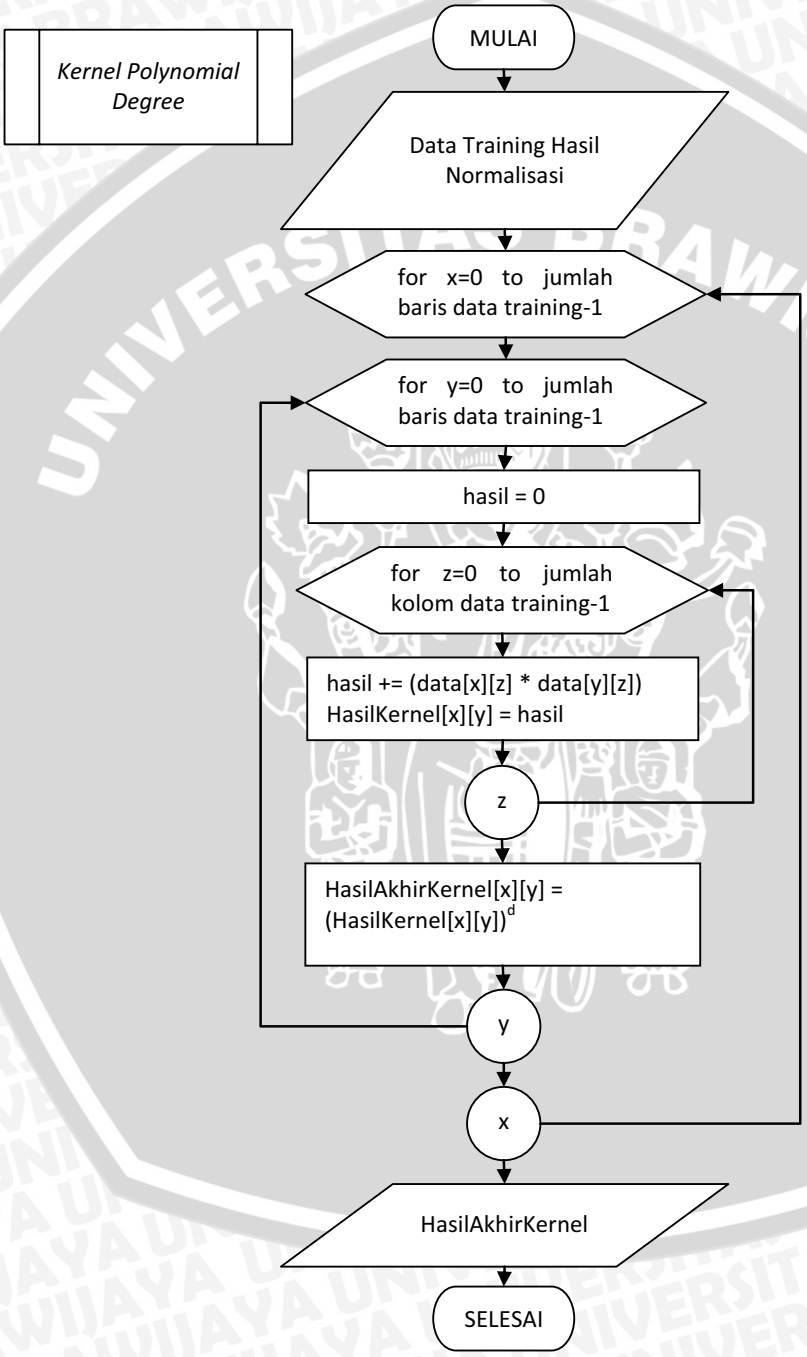
Gambar 4.12 Diagram Alir Proses *Support Vector Machine*.

Untuk mendapatkan *hyperplane* yang optimal dapat menggunakan algoritma *Sequential Training SVM*. Langkah-langkah untuk menghitung *SVM* berdasarkan diagram alir pada Gambar 4.12 di atas yaitu sebagai berikut.

1. Sistem menerima masukan berupa data hasil EKG yang telah di normalisasi.
2. Inisialisasi parameter nilai λ, γ (*learning rate*), *C* (variabel *slack*), *iterasiMax*, ε (epsilon), dan mengatur α .
3. Inisialisasi variabel *iterasiStop = 0*, *x = 1* (*x* adalah definisi awal iterasi), *tempTotal = 0*.
4. Menghitung *kernel polynomial degree d* untuk setiap data. Kemudian menghitung matriks *hessian* berdasarkan Persamaan 2.14., Hitung nilai *Learning Rate*.
5. Lakukan iterasi dengan menghitung nilai *Ei* berdasarkan Persamaan 2.15, hitung nilai $\delta\alpha_i$ berdasarkan Persamaan 2.16, dan hitung nilai α_i berdasarkan Persamaan 2.17 hingga mencapai iterasi maksimal atau *max* $\delta\alpha_i < \varepsilon$.

6. Mendapatkan nilai support vector dari hasil α_i pada iterasi yang terakhir kemudian hitung nilai *W.X*$^+$ dan *W.X*$^-$, *αiyikplus* atau *αiyikminus* dan nilai bias atau *b* berdasarkan Persamaan 2.8. Sistem akan menampilkan hasil nilai support vector dan nilai *b*.

4.2.3.1. Proses *Kernel Polynomial Degree*

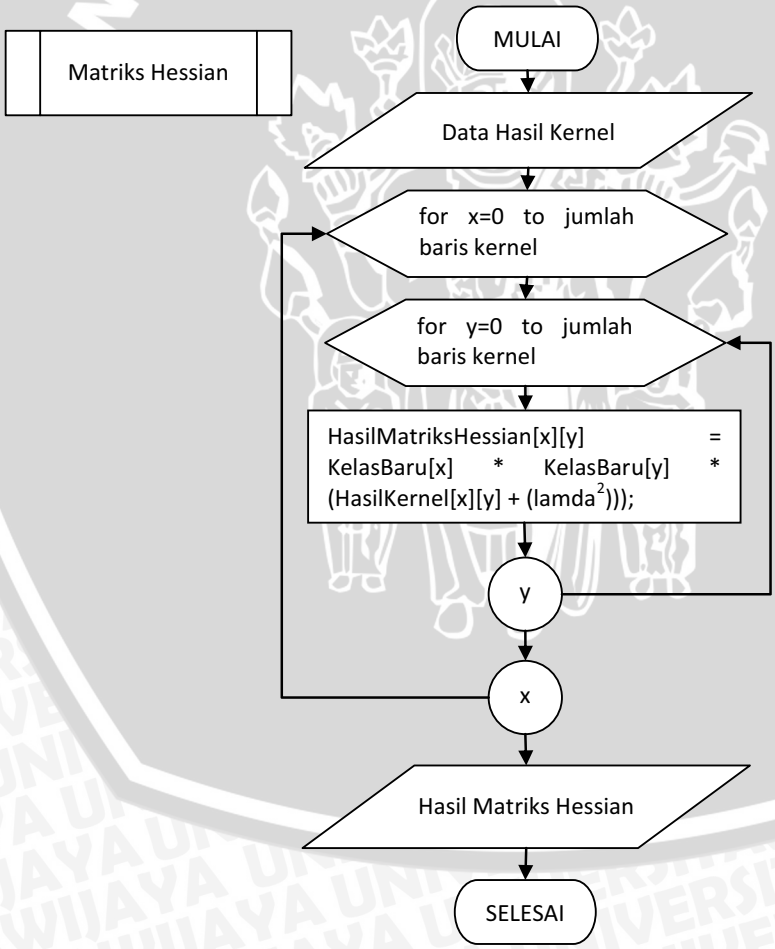


Gambar 4.13 Diagram Alir Proses *Kernel Polynomial Degree*.

Proses menghitung kernel terdapat pada Gambar 4.13 di atas. Pada tahap perhitungan *kernel* langkah yang dilakukan adalah mengambil data latih hasil normalisasi yang kemudian data tersebut diolah menggunakan *kernel Polynomial Degree d*. Kernel ini digunakan karena titik-titik data yang tidak linear sehingga digunakan kernel. Sehingga perhitungan SVM bisa menjadi *non-linear*. Langkah-langkah untuk menghitung *Kernel Polynomial Degree d* berdasarkan diagram alir di atas yaitu sebagai berikut.

1. Sistem menerima masukan berupa data *training* hasil normalisasi.
2. Perulangan pertama sampai jumlah baris pada data *training*.
3. Perulangan kedua sampai jumlah baris pada data *training*.
4. Inisialisasi parameter *hasil = 0*.
5. Perulangan ketiga sampai jumlah kolom pada data *training*.
6. Menghitung *kernel polynomial degree d* untuk setiap data.
7. Mendapatkan nilai *kernel polynomial degree* dengan matriks sebanyak jumlah baris pada data *training*. Sistem akan menampilkan hasil nilai kernel.

4.2.3.2. Proses *Matriks Hessian*

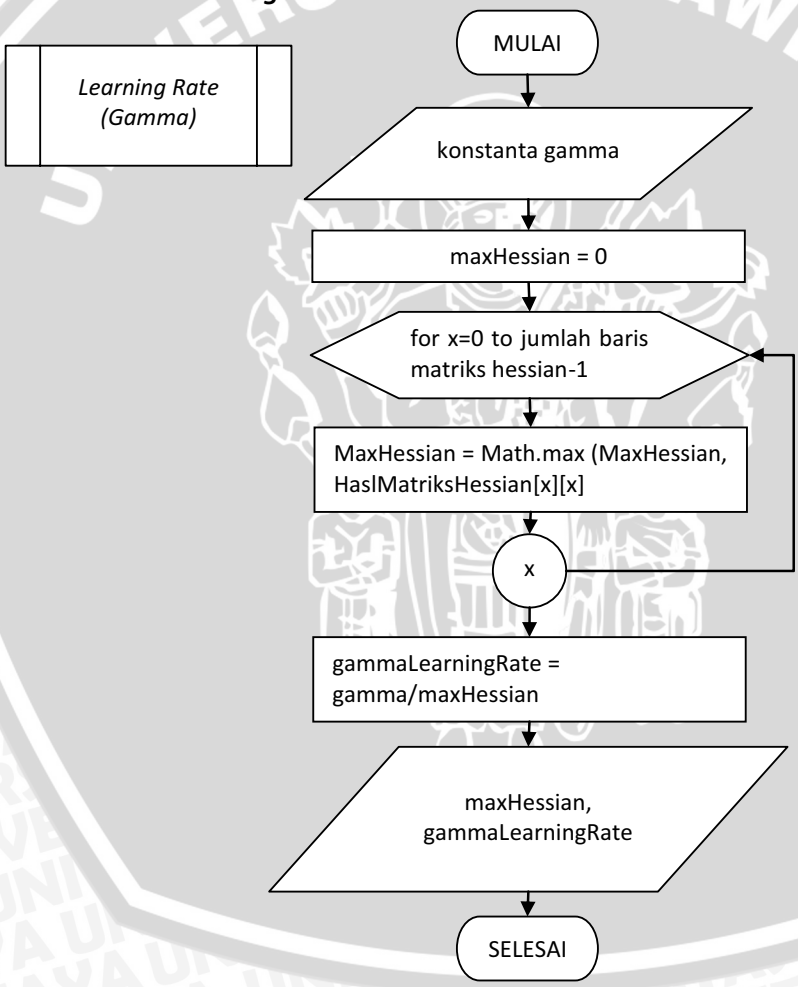


Gambar 4.14 Diagram Alir Proses *Matriks Hessian*.

Proses menghitung *matriks hessian* terdapat pada Gambar 4.14 di atas. Pada tahap perhitungan *matriks hessian* langkah yang dilakukan adalah mengambil data dari hasil perhitungan kernel. Langkah-langkah untuk menghitung *matriks hessian* berdasarkan diagram alir di atas yaitu sebagai berikut.

1. Sistem menerima masukan berupa data hasil kernel.
2. Perulangan pertama sampai jumlah baris pada data kernel.
3. Perulangan kedua sampai jumlah baris pada data kernel.
4. Menghitung Matriks Hessian berdasarkan Persamaan 2.14 untuk setiap data.
5. Mendapatkan nilai matriks hessian dengan matriks sebanyak jumlah baris pada data kernel.
6. Sistem akan menampilkan hasil matriks hessian.

4.2.3.3. Proses *Learning Rate*



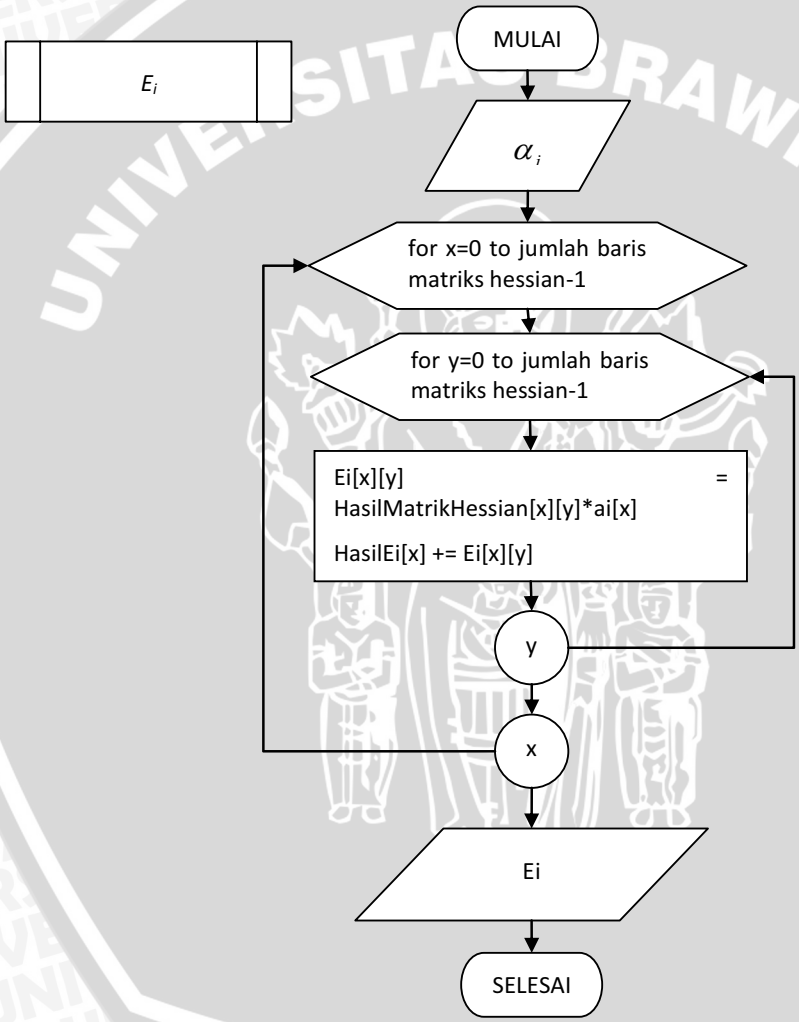
Gambar 4.15 Diagram Alir Proses *Learning Rate*.

Proses menghitung *learning rate* terdapat pada Gambar 4.15 di atas. Pada tahap perhitungan *learning rate* langkah yang dilakukan adalah mengambil

nilai konstanta *gamma*. Langkah-langkah untuk menghitung *learning rate* berdasarkan diagram alir di atas yaitu sebagai berikut.

1. Sistem menerima masukan berupa nilai konstanta *gamma*.
2. Inisialisasi variabel MaxHessian = 0.
3. Perulangan pertama sampai jumlah baris pada *matriks hessian*.
4. Mencari nilai $max_{\{i\}}d_{ii}$ atau nilai maksimal dari diagonal *Matriks Hessian*.
5. Menghitung nilai *learning rate* dengan membagi konstanta *gamma* dengan *maxHessian.*
6. Sistem akan menampilkan hasil *maxHessian* dan *learning rate*.

4.2.3.4. Proses E_i



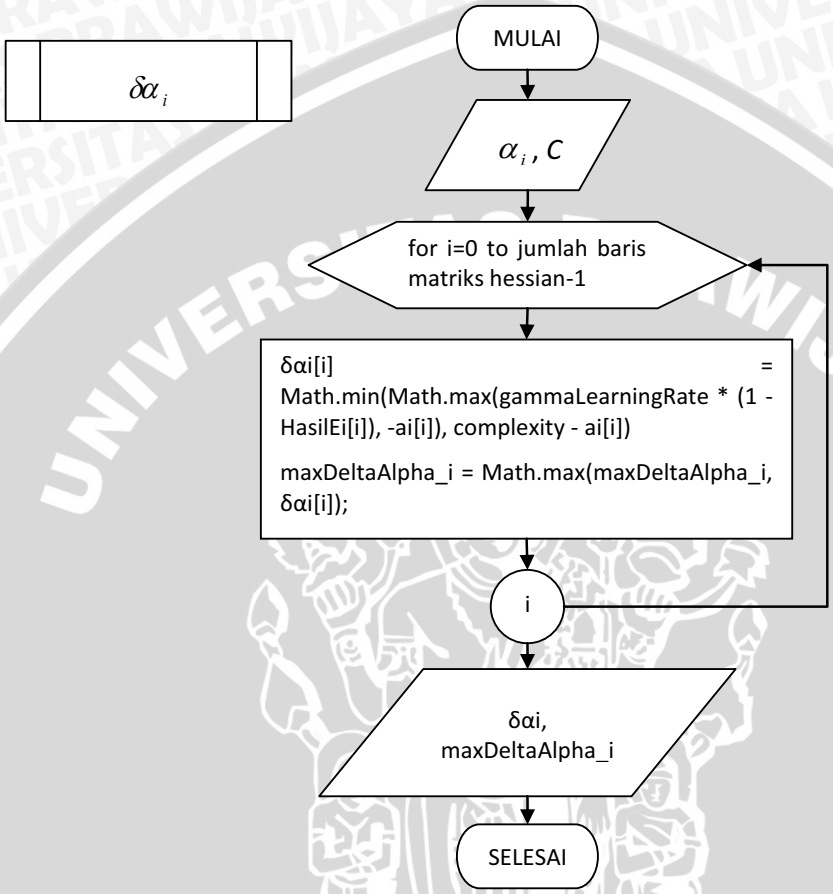
Gambar 4.16 Diagram Alir Proses E_i.

Proses menghitung nilai *Error* ke-*i* (E_i) terdapat pada Gambar 4.16 di atas. Pada tahap perhitungan E_i langkah yang dilakukan adalah mengambil nilai *alpha*. Langkah-langkah untuk menghitung E_i berdasarkan diagram alir di atas yaitu sebagai berikut.

1. Sistem menerima masukan berupa nilai *alpha*.

2. Perulangan pertama sampai jumlah baris pada matriks *hessian*. Perulangan kedua sampai jumlah baris pada matriks *hessian*.
3. Menghitung E_i berdasarkan pada Persamaan 2.14. Sistem akan menampilkan hasil E_i.

4.2.3.5. Proses $\delta\alpha_i$



Gambar 4.17 Diagram Alir Proses $\delta\alpha_i$.

Proses menghitung nilai $\delta\alpha_i$ terdapat pada Gambar 4.17 di atas. Langkah-langkah untuk menghitung $\delta\alpha_i$ berdasarkan diagram alir di atas yaitu sebagai berikut.

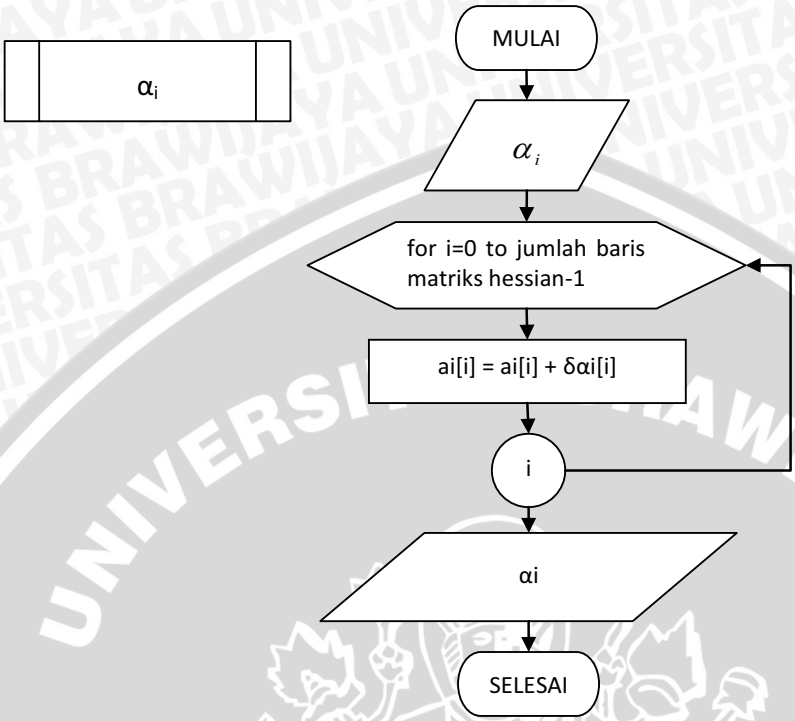
1. Sistem menerima masukan berupa nilai *alpha* dan *complexity (C)*.
2. Perulangan pertama sampai jumlah baris pada matriks *hessian*.
3. Menghitung $\delta\alpha_i$ berdasarkan pada Persamaan 2.16.
4. Sistem akan menampilkan hasil $\delta\alpha_i$ dan $Max\delta\alpha_i$.

4.2.3.6. Proses α_i

Proses menghitung nilai α_i terdapat pada Gambar 4.18 di bawah. Langkah-langkah untuk menghitung α_i berdasarkan diagram alir di bawah yaitu sebagai berikut.

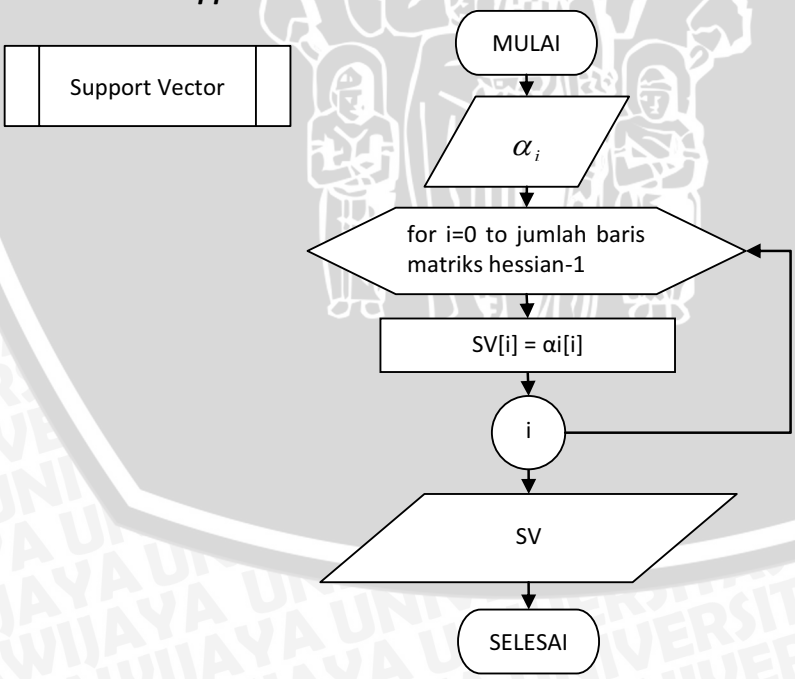
1. Sistem menerima masukan berupa nilai *alpha*. Perulangan pertama sampai jumlah baris pada matriks *hessian*.

2. Menghitung α_i berdasarkan pada Persamaan 2.17. Sistem akan menampilkan hasil α_i.



Gambar 4.18 Diagram Alir Proses α_i.

4.2.3.7. Proses *Support Vector*

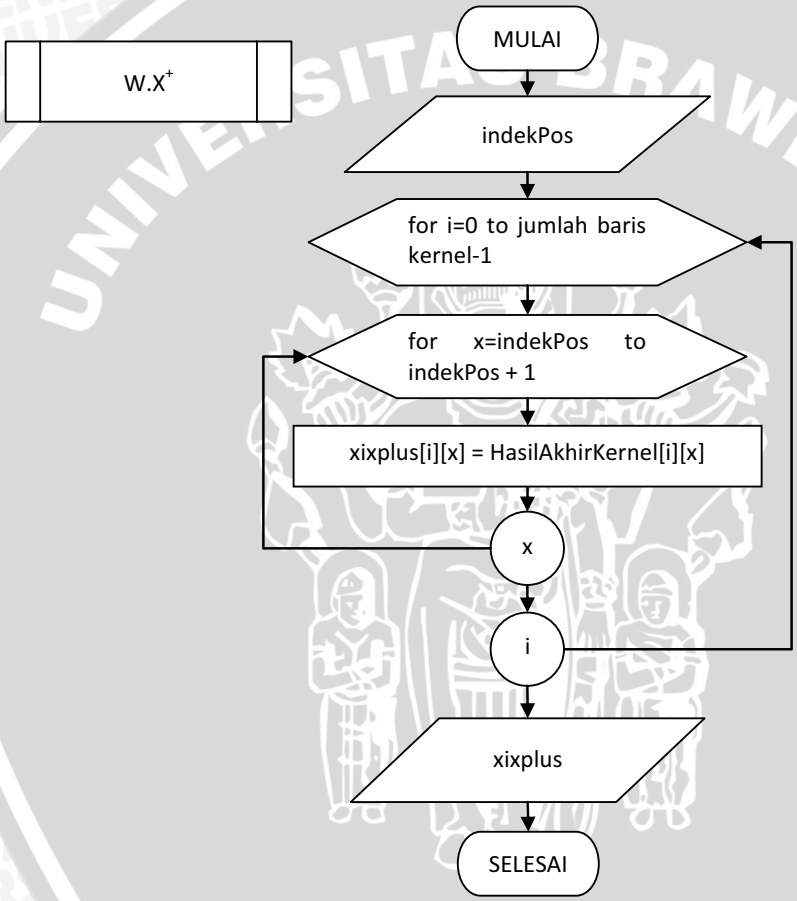


Gambar 4.19 Diagram Alir Proses *Support Vector*.

Proses mendapatkan nilai *support vector* terdapat pada Gambar 4.19 di atas. Pada tahap pengambilan nilai *support vector* adalah nilai α_i pada iterasi terakhir. Langkah-langkah untuk mengambil nilai *support vector* berdasarkan diagram alir di atas yaitu sebagai berikut.

1. Sistem menerima masukan berupa nilai *alpha*.
2. Perulangan pertama sampai jumlah baris pada *matriks hessian*.
3. Mengambil nilai alpha pada iterasi terakhir untuk dijadikan nilai *support vector*.
4. Sistem akan menampilkan hasil α_i.

4.2.3.8. Proses $W.X^+$



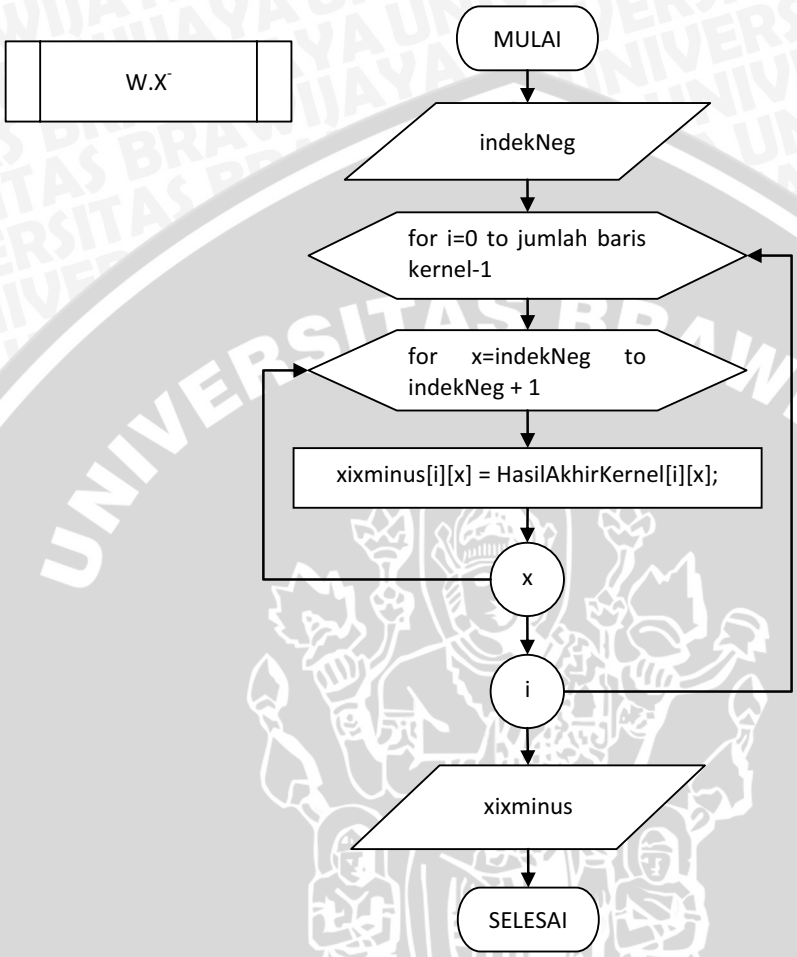
Gambar 4.20 Diagram Alir Proses $W.X^+$.

Proses mendapatkan nilai $W.X^+$ terdapat pada Gambar 4.20 di atas. Pada tahap pengambilan nilai $W.X^+$ adalah *indekPos* yang merupakan nilai dari indek dari *max Alpha* dengan bobot positif. Langkah-langkah untuk mengambil nilai $W.X^+$ berdasarkan diagram alir di atas yaitu sebagai berikut.

1. Sistem menerima masukan berupa nilai *indekPos*.
2. Perulangan pertama sampai jumlah baris pada kernel.
3. Perulangan kedua dimulai dari *indekPos* hingga *indekPos + 1*. Untuk mendapatkan nilai bobot positif. Sehingga *indekPos* untuk mengetahui nilai *max alpha*.

4. Mengambil nilai kernel sebagai bobot positif yang optimal.
5. Sistem akan menampilkan hasil *W.X*$^{+}$.

4.2.3.9. Proses *W.X*$^{-}$

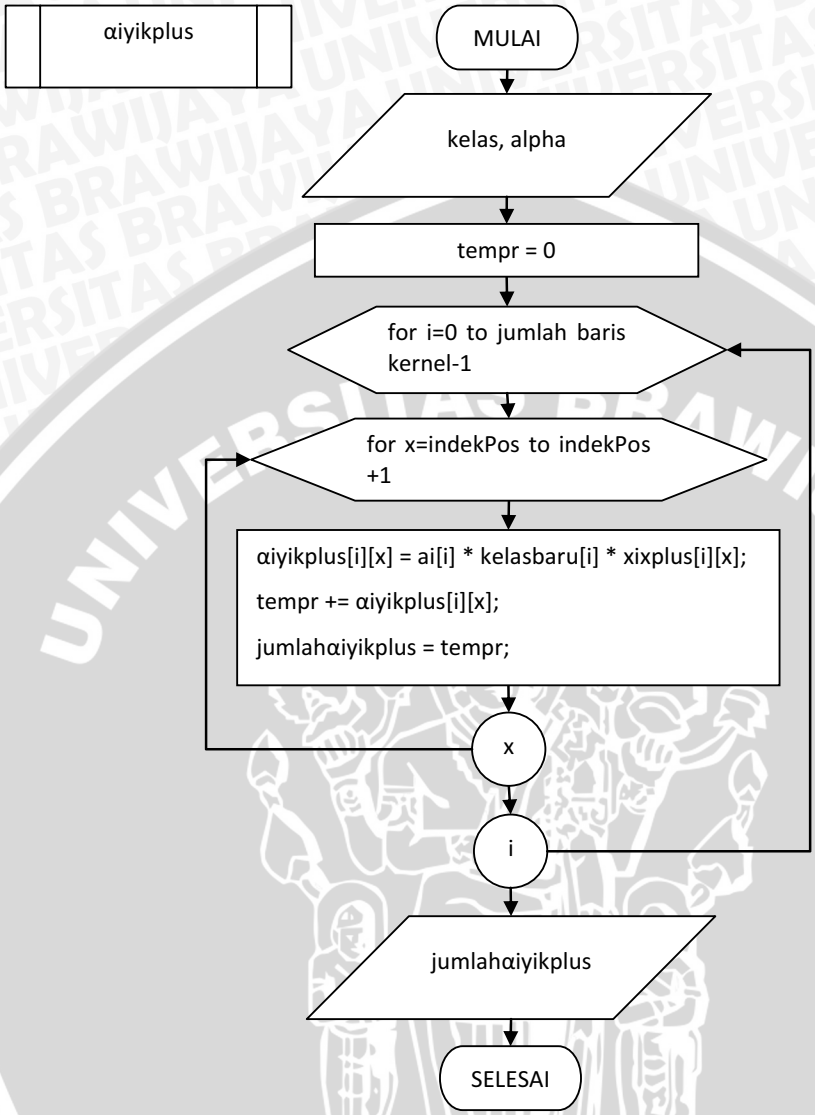


Gambar 4.21 Diagram Alir Proses *W.X*$^{-}$.

Proses mendapatkan nilai *W.X*$^{-}$ terdapat pada Gambar 4.21 di atas. Pada tahap pengambilan nilai *W.X*$^{-}$ adalah *indekNeg* yang merupakan nilai dari indek dari *max Alpha* dengan bobot negatif. Langkah-langkah untuk mengambil nilai *W.X*$^{-}$ berdasarkan diagram alir di atas yaitu sebagai berikut.

1. Sistem menerima masukan berupa nilai *indekNeg*.
2. Perulangan pertama sampai jumlah baris pada kernel.
3. Perulangan kedua dimulai dari *indekNeg* hingga *indekNeg +1*. Untuk mendapatkan nilai bobot negatif. Sehingga *indekPos* untuk mengetahui nilai *max alpha*. *indekNeg* adalah suatu variabel yang digunakan untuk menyimpan nilai dari indek dari nilai *alpha* tertinggi berada pada data ke berapa. Sehingga berdasarkan indek tersebut yang nantinya akan dihitung bobot dari *support vector*.
4. Mengambil nilai kernel sebagai bobot negatif yang optimal.
5. Sistem akan menampilkan hasil *W.X*$^{-}$.

4.2.3.10. Proses *αiyikplus*

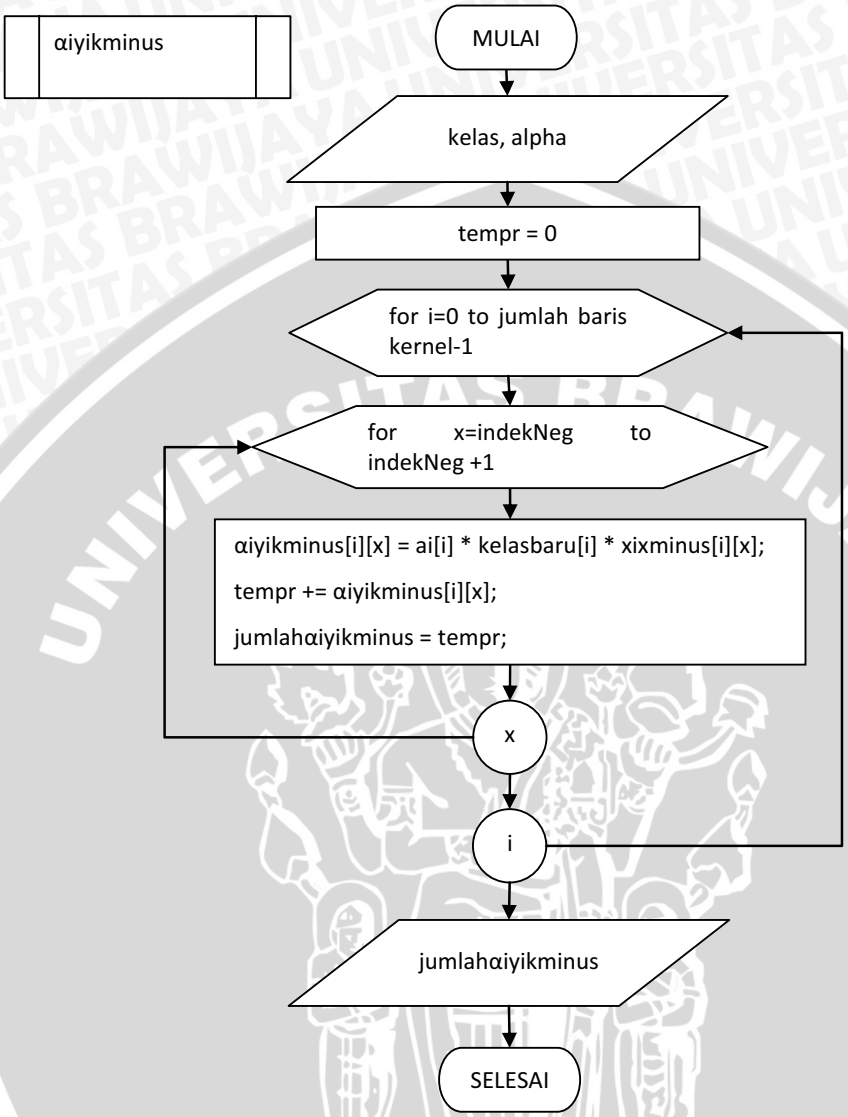


Gambar 4.22 Diagram Alir Proses *jumlahαiyikplus.*

Proses mendapatkan nilai *jumlahαiyikplus* terdapat pada Gambar 4.22 di atas. Langkah-langkah untuk mengambil nilai *jumlahαiyikplus* berdasarkan diagram alir di atas yaitu sebagai berikut.

1. Masukan sistem berupa kelas yang didapatkan dari hasil *Binary Decision Tree* dan nilai *alpha.*
2. Insialisasi awal *tempr = 0*. Variabel itu merupakan variabel yang digunakan untuk menyimpan nilai perhitungan ini yaitu *jumlahαiyikplus.*
3. Menghitung *jumlahαiyikplus* berdasarkan Persamaan 2.13. Dimana akan mengalikan nilai dari *alpha* kemudian dengan kelas *BDT* dan hasil dari *xixplus.* Hasil tersebut disimpan di variabel *tempr* dan disimpan lagi di variabel *jumlahαiyikplus.*
4. Sistem akan menampilkan hasil *jumlahαiyikplus.*

4.2.3.11. Proses *αiyikminus*

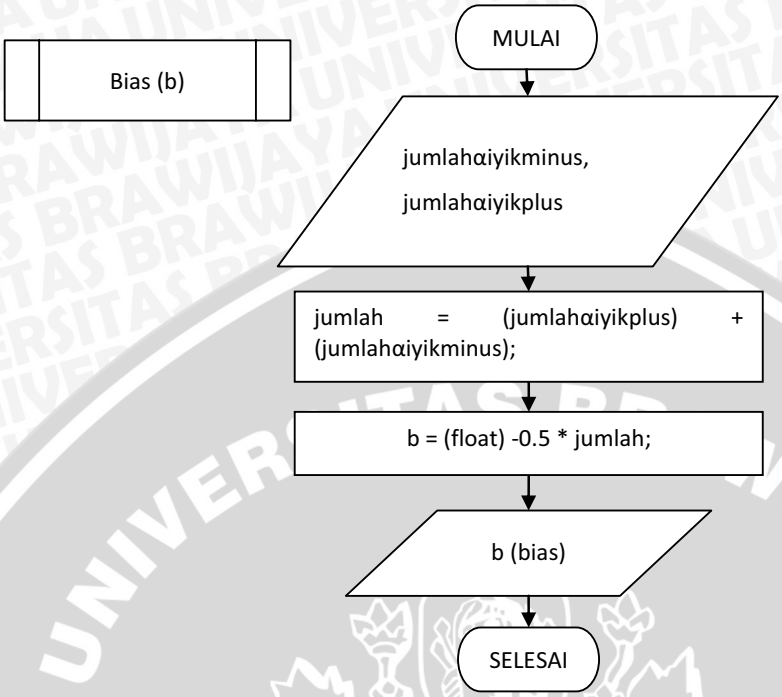


Gambar 4.23 Diagram Alir Proses jumlahαiyikminus.

Proses mendapatkan nilai *jumlahαiyikminus* terdapat pada Gambar 4.23 di atas. Langkah-langkah untuk mengambil nilai *jumlahαiyikminus* berdasarkan diagram alir di atas yaitu sebagai berikut.

1. Masukan sistem berupa kelas yang didapatkan dari hasil *Binary Decision Tree* dan nilai *alpha.*
2. Insialisasi awal *tempr = 0*. Variabel itu merupakan variabel yang digunakan untuk menyimpan nilai perhitungan ini yaitu *jumlahαiyikminus.*
3. Menghitung *jumlahαiyikminus* berdasarkan Persamaan 2.13. Dimana akan mengalikan nilai dari *alpha* kemudian dengan kelas *BDT* dan hasil dari *xixminus.* Hasil tersebut disimpan di variabel *tempr* dan disimpan lagi di variabel *jumlahαiyikminus.*
4. Sistem akan menampilkan hasil *jumlahαiyikminus.*

4.2.3.12. Proses Bias (b)



Gambar 4.24 Diagram Alir Proses *b (bias).*

Proses mendapatkan nilai *b (bias)* terdapat pada Gambar 4.24 di atas. Pada tahap pengambilan nilai *b (bias)* adalah nilai jumlahαiyikminus dan *jumlahαiyikplus*. Langkah-langkah untuk mengambil nilai *b (bias)* berdasarkan diagram alir di atas yaitu sebagai berikut.

1. Sistem menerima nilai *jumlahαiyikminus* dan *jumlahαiyikplus*.
2. Menghitung jumlah dari *jumlahαiyikminus* dan *jumlahαiyikplus*.
3. Menghitung nilai *b* dengan *-0.5 * jumlah*.
4. Sistem akan menampilkan hasil b.

4.3 Langkah-Langkah Penyelesaian Masalah Algoritma *BDT-SVM*

Langkah-langkah algoritma *BDT-SVM* yang telah diuraikan pada Sub bab 4.2 selanjutnya disederhanakan menjadi perhitungan manual untuk memudahkan pemahaman tentang penyelesaian klasifikasi detak jantung dengan hasil pemeriksaan EKG sebelum diimplementasikan ke dalam kode program. Siklus penyelesaian masalah diawali dengan perhitungan *gravity center*, jarak euclidean, pembentukan *tree* dengan menggunakan metode *BDT*, normalisasi data menggunakan metode *Min-Max Normalization*, perhitungan klasifikasi data dengan metode *SVM.*

Perhitungan manual pada penelitian ini menggunakan sampel data hasil rekam EKG sebanyak 16 *records* untuk data *training* dan 4 *records* untuk data *testing*. Sehingga terdapat 20 total data sampel yang digunakan. Lalu mengambil 5 fitur awal berdasarkan waktu untuk setiap pemeriksaan sehingga terdapat 10 fitur untuk dua pemeriksaan.

4.3.1 Data *Training* dan Data *Testing*

Berikut ini adalah data-data yang digunakan dimana terdapat 16 data sampel sebagai data *training* dan 4 data sampel sebagai data *testing* pada Tabel 4.2 dan 4.3.

Tabel 4.2 Data *Training*.

Pasien ke-	MLII					VI					Kelas
1	-0.25	-0.245	-0.225	-0.235	-0.245	-0.06	-0.06	-0.075	-0.07	-0.075	1
2	-0.165	-0.175	-0.18	-0.18	-0.18	-0.01	-0.025	-0.035	-0.02	-0.015	1
3	-0.31	-0.28	-0.275	-0.27	-0.28	0.125	0.145	0.125	0.125	0.125	1
4	-0..485	-0.46	-0.45	-0.455	-0.465	-0.72	-0.715	-0.73	-0.73	-0.74	1
5	-0.185	-0.175	-0.16	-0.155	-0.15	-0.01	-0.005	0	0.01	-0.01	2
6	-0.155	-0.185	-0.225	-0.245	-0.265	0.36	0.34	0.31	0.305	0.295	2
7	0.11	0.125	0.115	0.115	0.135	0.1	0.1	0.095	0.09	0.095	2
8	-0.39	-0.38	-0.395	-0.385	-0.375	0.145	0.12	0.105	0.08	0.065	2
9	0.175	0.18	0.195	0.19	0.2	0.065	0.105	0.125	0.14	0.16	3
10	-0.215	-0.215	-0.215	-0.215	-0.215	0.095	0.095	0.095	0.095	0.095	3
11	0.285	0.295	0.305	0.305	0.305	-0.11	-0.09	-0.08	-0.095	-0.105	3
12	-0.54	-0.515	-0.5	-0.495	-0.49	0.11	0.095	0.075	0.055	0.055	3
13	-0.25	-0.14	-0.005	0.16	0.355	-0.4	-0.42	-0.44	-0.47	-0.52	4
14	0.05	0.125	0.195	0.255	0.32	0.695	0.655	0.615	0.56	0.515	4
15	-0.565	-0.59	-0.58	-0.58	-0.59	0.675	0.705	0.735	0.745	0.735	4
16	0.13	0.12	0.115	0.135	0.175	0.26	0.265	0.28	0.28	0.3	4

Tabel 4.3 di atas merupakan tabel yang menampilkan data *training* dari sampel data yang akan digunakan untuk proses BDT-SVM.

Tabel 4.3 Data *Testing*.

Pasien ke-	MLII					VI					Kelas
17	-0.285	-0.29	-0.315	-0.315	-0.3	0.135	0.045	-0.095	-0.265	-0.43	1
18	-0.59	-0.605	-0.64	-0.67	-0.705	0.08	0.07	0.035	0.01	-0.01	2
19	-0.105	-0.095	-0.095	-0.105	-0.11	-0.065	-0.055	-0.05	-0.04	-0.06	3
20	-0.08	-0.09	-0.09	-0.08	-0.1	0.3	0.295	0.325	0.35	0.35	4

Tabel 4.3 di atas merupakan tabel yang menampilkan data *training* dari sampel data yang akan digunakan untuk proses BDT-SVM.

4.3.2 Perhitungan *Gravity Center*

Langkah yang dilakukan pertama yaitu menghitung *gravity center* atau titik pusat data atau *center point* berdasarkan masing-masing kelas. Fungsi untuk menghitung rata-rata *gravity center* dapat dilihat pada Persamaan 2.18. Dimana akan mencari nilai rata-rata dari setiap parameter dan dihitung berdasarkan

kelas yang sama. Sehingga untuk hasil dari *gravity center* yaitu terdiri dari empat (4) hasil gravity center. Hasil perhitungan rata-rata *gravity center* masing-masing kelas terdapat pada Tabel 4.4, 4.5, 4.6, dan 4.7.

Tabel 4.4 Rata-Rata *Gravity Center* Kelas 1.

Pasien ke-	MLII					VI					Kelas
1	-0.25	-0.245	-0.225	-0.235	-0.245	-0.06	-0.06	-0.075	-0.07	-0.075	1
2	-0.165	-0.175	-0.18	-0.18	-0.18	-0.01	-0.025	-0.035	-0.02	-0.015	1
3	-0.31	-0.28	-0.275	-0.27	-0.28	0.125	0.145	0.125	0.125	0.125	1
4	-0.485	-0.46	-0.45	-0.455	-0.465	-0.72	-0.715	-0.73	-0.73	-0.74	1
Rata2	**-0.3025**	**-0.29**	**-0.2825**	**-0.285**	**-0.2925**	**-0.1663**	**-0.1638**	**-0.1788**	**-0.1738**	**-0.1763**	

Tabel 4.5 Rata-Rata *Gravity Center* Kelas 2.

Pasien ke-	MLII					VI					Kelas
5	-0.185	-0.175	-0.16	-0.155	-0.15	-0.01	-0.005	0	0.01	-0.01	2
6	-0.155	-0.185	-0.225	-0.245	-0.265	0.36	0.34	0.31	0.305	0.295	2
7	0.11	0.125	0.115	0.115	0.135	0.1	0.1	0.095	0.09	0.095	2
8	-0.39	-0.38	-0.395	-0.385	-0.375	0.145	0.12	0.105	0.08	0.065	2
Rata2	**-0.155**	**-0.1538**	**-0.1663**	**-0.1675**	**-0.1638**	**0.1488**	**0.1388**	**0.1275**	**0.1213**	**0.1113**	

Tabel 4.6 Rata-Rata *Gravity Center* Kelas 3.

Pasien ke-	MLII					VI					Kelas
9	0.175	0.18	0.195	0.19	0.2	0.065	0.105	0.125	0.14	0.16	3
10	-0.215	-0.215	-0.215	-0.215	-0.215	0.095	0.095	0.095	0.095	0.095	3
11	0.285	0.295	0.305	0.305	0.305	-0.11	-0.09	-0.08	-0.095	-0.105	3
12	-0.54	-0.515	-0.5	-0.495	-0.49	0.11	0.095	0.075	0.055	0.055	3
Rata2	**-0.0738**	**-0.0638**	**-0.0538**	**-0.0538**	**-0.05**	**0.04**	**0.0513**	**0.0538**	**0.0488**	**0.0513**	

Tabel 4.7 Rata-Rata *Gravity Center* Kelas 4.

Pasien ke-	MLII					VI					Kelas
13	-0.25	-0.14	-0.005	0.16	0.355	-0.4	-0.42	-0.44	-0.47	-0.52	4
14	0.05	0.125	0.195	0.255	0.32	0.695	0.655	0.615	0.56	0.515	4
15	-0.565	-0.59	-0.58	-0.58	-0.59	0.675	0.705	0.735	0.745	0.735	4
16	0.13	0.12	0.115	0.135	0.175	0.26	0.265	0.28	0.28	0.3	4
Rata2	**-0.1588**	**-0.1213**	**-0.0688**	**-0.0075**	**0.065**	**0.3075**	**0.3013**	**0.2975**	**0.2788**	**0.2575**	

4.3.3 Perhitungan *Euclidean Distance*

Setelah melakukan perhitungan *Gravity Center*, langkah selanjutnya adalah menghitung jarak antar kelas. Perhitungan jarak ini dengan menggunakan jarak euclidean dengan menggunakan Persamaan 2.21. Di bawah ini merupakan cara untuk menghitung jarak euclidean.

$$d_{1,1} = \sqrt{\begin{matrix}(-0.3025-(-0.3025))^2+(-0.29-(-0.29))^2+(-0.2825-(-0.2825))^2 \\ +....+(-0.17625-(-0.17625))^2\end{matrix}} = 0$$

$$d_{1,2} = \sqrt{\begin{matrix}(-0.3025-(-0.155))^2+(-0.29-(-0.16625))^2+(-0.2825- \\ (-0.1675))^2+....+(-0.17625-0.11125)^2\end{matrix}} = 0.73377$$

.....

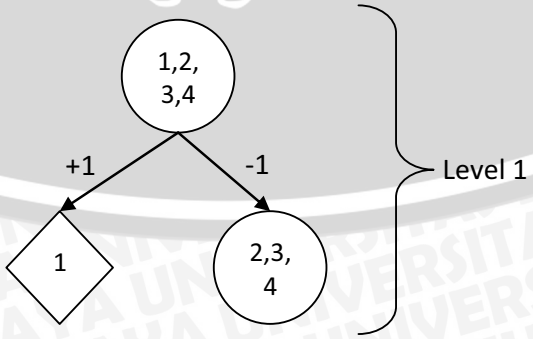
$$d_{4,4} = \sqrt{\begin{matrix}(-0.15875-(-0.15875))^2+(-0.12125-(-0.12125))^2+(-0.06875- \\ (-0.06875))^2+....+(0.2575-0.2575)^2\end{matrix}} = 0.$$

Berdasarkan hasil di atas, maka didapatkan hasil perhitungan matriks *Euclidean Distance* terlihat pada Tabel 4.8 berikut ini.

Tabel 4.8 Matriks Euclidean Distance.

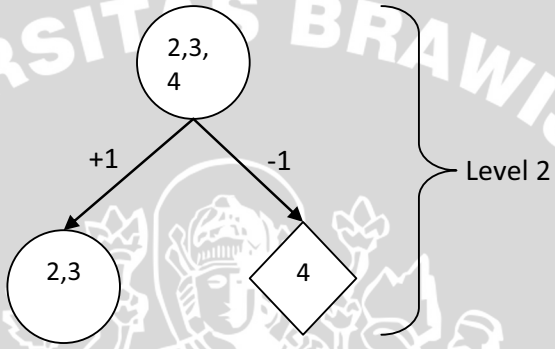
	C1	C2	C3	C4
C1	0	0.73377	0.715687	1.166197
C2	0.73377	0	0.294981	0.463903
C3	0.715687	0.294981	0	0.561275
C4	1.166197	0.463903	0.561275	0

Setelah mendapatkan jarak antar kelas, langkah berikutnya adalah menyusun pohon biner. Langkah pertama adalah mencari nilai terbesar dari Matriks Euclidean diatas. Dimana nilai terbesarnya yaitu 1.166197 dan terletak antara Kelas 1 (*C1*) dan Kelas 4 (*C4*), Sehingga Kelas 1 mengarah ke kiri (positif) dan kelas 4 mengarah ke kanan (negatif). Kemudian kelas yang tersisa adalah Kelas 2 (*C2*) dan Kelas 3 (*C3*). Dicari jarak terdekatnya antara Kelas 1 dan Kelas 4. Kelas 2 mengarah ke kanan karena nilai lebih mendekati ke kelas 4 dibandingkan kelas 1 yaitu 0.463903 < 0.73377 (*C4 < C1*). Kelas 3 mengarah ke kanan karena nilai lebih mendekati ke kelas 4 dibandingkan ke kelas 1 yaitu 0.561275 < 0.715687 (*C4 < C1*). Langkah ini telah menyelesaikan *SVM* Level 1 pada Gambar 4.25 di bawah ini. Jika pada perhitungan *f(x) testing*, data uji dikelompokkan ke kelas positif maka secara otomatis, data tersebut akan masuk ke kelas 1. Sedangkan jika terdapat data uji yang masuk ke kelas negatif maka prosesnya akan diturunkan lagi ke level berikutnya yaitu Level 2.



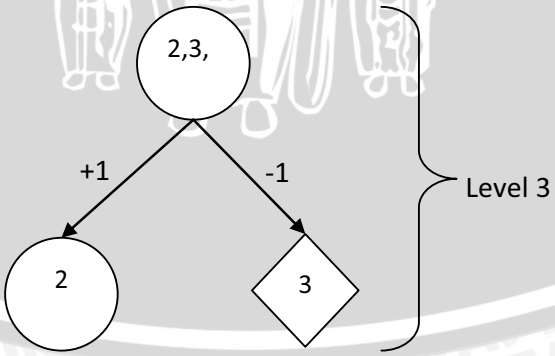
Gambar 4.25 BDT Level 1.

Untuk Level 2, dicari lagi jarak terdekat antara kelas 2, 3, dan 4. Dimana nilai terbesarnya yaitu 0.56128 berdasarkan dari jarak euclidean di atas dan terletak antara Kelas 3 (*C3*) dan Kelas 4 (*C4*). Sehingga Kelas 3 mengarah ke kiri dan Kelas 4 mengarah ke kanan. Tersisa Kelas 2 dan dicari jarak terdekat antara Kelas 3 dan Kelas 4. Kelas 2 mengarah ke kiri karena nilai lebih mendekati ke kelas 3 daripada ke kelas 4 yaitu 0.294981 < 0.463903 (*C2 < C4*) atau kelas 2 lebih kecil dibandingkan dengan kelas 4 dan bisa dilihat pada Gambar 4.26 di bawah ini. Jika pada perhitungan *f(x) testing*, data uji dikelompokkan ke kelas negatif maka secara otomatis, data tersebut akan masuk ke kelas 4. Sedangkan jika terdapat data uji yang masuk ke kelas positif maka prosesnya akan diturunkan lagi ke level berikutnya yaitu Level 3.



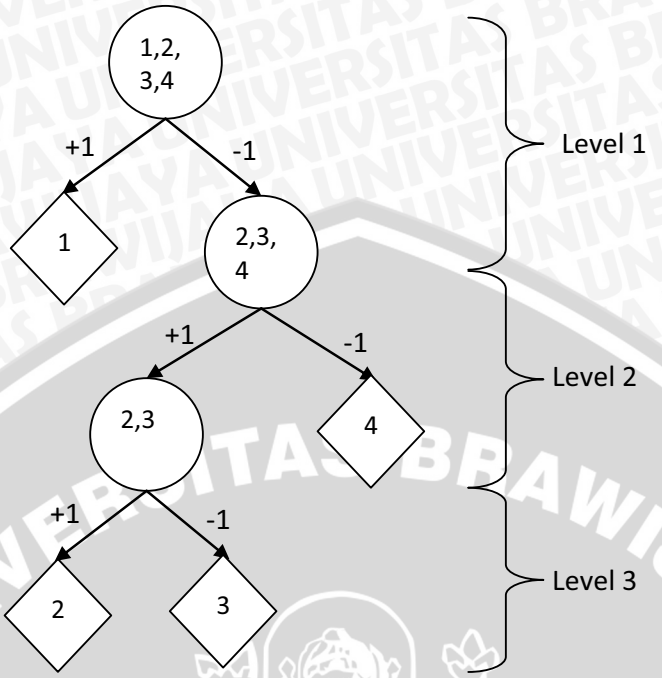
Gambar 4.26 BDT Level 2.

Untuk level 3, karena hanya tersisa dua kelas yaitu Kelas 2 (*C2*) dan Kelas 3 (*C3*), maka Kelas 2 akan langsung mengarah ke kiri dan Kelas 3 akan mengarah ke kanan. Langkah ini telah menyelesaikan SVM Level 3. Sehingga jika pada level sebelumnya masih terdapat level yang bisa diturunkan lagi maka akan langsung membagi ke kelas yang lebih kecil ke arah kiri dan kelas yang lebih besar ke arah kanan. Hasil untuk *binary decision tree* pada level 3 ini dapat dilihat pada Gambar 4.27 di bawah ini.



Gambar 4.27 BDT Level 2.

Ilustrasi pohon biner untuk Level 1 hingga Level 3 dapat dilihat pada Gambar 4.28 di bawah ini. Yang merupakan gabungan dari hasil *Binary Decision Tree* dari level ke-1, level ke-2, hingga level ke-3.



Gambar 4.28 *Binary Decision Tree* atau Pohon Biner.

4.3.4 Perhitungan Normalisasi Data

Proses selanjutnya adalah melakukan normalisasi pada setiap nilai dari data *training* dan *testing* yang bertujuan untuk menyeragamkan nilai fitur dalam rentang nilai tertentu. Untuk persamaan dapat dilihat pada bab sebelumnya yaitu pada Persamaan 2.1. Parameter yang digunakan adalah nilai minimal dan maksimal untuk setiap fitur. Di bawah ini merupakan contoh perhitungan.

$X^* = \frac{-0.25-(-0.59)}{0.285-(-0.59)} = 0.388571$ untuk parameter /fitur ke-1 pada data ke-1.,

$X^* = \frac{-0.245-(-0.605)}{0.295-(-0.605)} = 0.4$ untuk parameter /fitur ke-2 pada data ke-1.

$X^* = \frac{-0.225-(-0.64)}{0.305-(-0.64)} = 0.439153$ untuk parameter /fitur ke-3 pada data ke-1.

$X^* = \frac{-0.235-(-0.67)}{0.305-(-0.67)} = 0.446154$ untuk parameter /fitur ke-4 pada data ke-1.

....

$X^* = \frac{0.28-(-0.73)}{0.745-(-0.73)} = 0.684746$ untuk parameter /fitur ke-9 pada data ke-16.

$X^* = \frac{0.3-(-0.74)}{0.735-(-0.74)} = 0.705085$ untuk parameter /fitur ke-10 pada data ke-16.

Hasil normalisasi nilai fitur terdapat pada Tabel 4.9.

UNIVERSITAS BRAWIJAYA repository.ub.ac

Tabel 4.9 Data Setelah di Normalisasi.

Pasien ke-	MLII					VI					Kelas
1	0.3886	0.4	0.4392	0.4462	0.434	0.4664	0.4613	0.4471	0.4475	0.4508	1
2	0.4857	0.4778	0.4868	0.5026	0.4953	0.5018	0.4859	0.4744	0.4814	0.4915	1
3	0.32	0.3611	0.3862	0.4103	0.4009	0.5972	0.6056	0.5836	0.5797	0.5864	1
4	0.12	0.1611	0.2011	0.2205	0.2264	0	0	0	0	0	1
5	0.4629	0.4778	0.5079	0.5282	0.5236	0.5018	0.5	0.4983	0.5017	0.4949	2
6	0.4971	0.4667	0.4392	0.4359	0.4151	0.7633	0.743	0.7099	0.7017	0.7017	2
7	0.8	0.8111	0.7989	0.8051	0.7925	0.5795	0.5739	0.5631	0.5559	0.5661	2
8	0.2286	0.25	0.2593	0.2923	0.3113	0.6113	0.588	0.57	0.5492	0.5458	2
9	0.8743	0.8722	0.8836	0.8821	0.8538	0.5548	0.5775	0.5836	0.5898	0.6102	3
10	0.4286	0.4333	0.4497	0.4667	0.4623	0.576	0.5704	0.5631	0.5593	0.5661	3
11	1	1	1	1	0.9528	0.4311	0.4401	0.4437	0.4305	0.4305	3
12	0.0571	0.1	0.1481	0.1795	0.2028	0.5866	0.5704	0.5495	0.5322	0.539	3
13	0.3886	0.5167	0.672	0.8513	1	0.2261	0.2077	0.198	0.1763	0.1492	4
14	0.7314	0.8111	0.8836	0.9487	0.967	1	0.9648	0.9181	0.8746	0.8508	4
15	0.0286	0.0167	0.0635	0.0923	0.1085	0.9859	1	1	1	1	4
16	0.8229	0.8056	0.7989	0.8256	0.8302	0.6926	0.6901	0.6894	0.6847	0.7051	4
17	0.3486	0.35	0.3439	0.3641	0.3821	0.6042	0.5352	0.4334	0.3153	0.2102	1
18	0	0	0	0	0	0.5654	0.5528	0.5222	0.5017	0.4949	2
19	0.5543	0.5667	0.5767	0.5795	0.5613	0.4629	0.4648	0.4642	0.4678	0.461	3
20	0.5829	0.5722	0.582	0.6051	0.5708	0.7208	0.7113	0.7201	0.7322	0.739	4

4.3.5 *Training* Data

Setelah menyelesaikan semua tahapan diatas, selanjutnya adalah melakukan *training* data. Proses *training* dimulai pada Level 1. Dalam melakukan *training* data diperlukan inisialisasi parameter antara lain $\lambda = 0.5$, $\gamma = 0.01$, $C = 1$, $\varepsilon = 0.00001$, $\alpha_i = 0$, $Threshold = 0$, dan $IterasiMax = 2$. Data yang digunakan yaitu pada Tabel 4.9 pada baris 1-16 di bawah ini.

Training Level 1

Pertama menghitung matriks *kernel*. *Kernel* yang digunakan adalah *kernel polynomial d* dan *d = 2*. Fungsi untuk menghitung matriks *kernel* ini adalah $K(x,y) = (x.y)^2$. Untuk proses perhitungan bisa dilihat di bawah ini.

$$K(1,1) = \begin{pmatrix} (0.388571*0.388571)+(0.4*0.4)+(0.43915344* \\ 0.43915344)+(0.44615385*0.44615385)+ \\ (0.433962264*0.433962264)+(0.466431*0.466431) \\ +(0.461268*0.461268)+(0.447099*0.447099)+ \\ (0.447458*0.447458)+(0.450847458*0.450847458)\_ \end{pmatrix}^2 = 3.705336$$

$$K(1,2) = \begin{pmatrix} (0.388571*0.485714)+(0.4*0.477778)+(0.43915344 \\ *0.48677249)+(0.44615385*0.5025641)+(0.433962264 \\ *0.495283019)+(0.466431*0.501767)+(0.461268* \\ 0.485915)+(0.447099*0.474403)+(0.447458* \\ 0.481356)+(0.450847458*0.491525424) \end{pmatrix}^2 = 4.57977$$

....

$$K(16,16) = \begin{pmatrix} (0.822857143*0.822857143)+(0.805555556* \\ 0.805555556)+(0.798941799*0.798941799)+ \\ (0.825641026*0.825641026)+(0.830188679* \\ 0.830188679)+(0.692579505*0.692579505)+ \\ (0.690140845*0.690140845)+(0.689419795* \\ 0.689419795)+(0.684745763*0.684745763)+ \\ (0.705084746*0.705084746) \end{pmatrix}^2 = 32.86155$$

Hasil dari perhitungan matriks *kernel* ini dapat dilihat pada Tabel 4.10 di bawah ini.

Tabel 4.10 Hasil Perhitungan Matriks Kernel Polynomial Degree 2.

K(x,y)	1	2	3	4	....	16
1	3.705336	4.579761	4.571226	8.880791	....	10.859345
2	4.57977	5.689446	5.563960	11.187664	....	13.584152
3	4.571226	5.563960	6.026968	10.125667	....	12.807161
4	0.156821	0.20802	0.12636806	0.03264197	....	0.576373

5	4.804783	5.958451	5.84567284	0.22058662	**....**	14.22002
....	....	....	....	....	....	....
16	10.85934	13.58415	12.80716	27.438092	**....**	32.86155

Kemudian menghitung Matriks Hessian dengan rumus $D_{ij} = y_i y_j (K(x_i, x_j) + \lambda^2)$. Untuk proses perhitungan bisa dilihat di bawah ini.

$$D_{11} = 1*1*(3.705336 + 0.5^2) = 3.955336$$

$$D_{21} = 1*1*(4.57977 + 0.5^2) = 4.82977$$

$$D_{22} = 1*1*(5.689446 + 0.5^2) = 5.939446$$

....,dst

Hasil perhitungan matriks hessian bisa dilihat pada Tabel 4.11 di bawah ini.

Tabel 4.11 Hasil Perhitungan Matriks Hessian.

D_{ij}	1	2	3	4	....	16
1	3.9553	4.8298	4.8212	0.4068	....	-11.109
2	4.8298	5.9394	5.814	0.458	....	-13.834
3	4.8212	5.814	6.277	0.3764	....	-13.057
4	0.4068	0.458	0.3764	0.2826	....	-0.8264
5	-5.055	-6.208	-6.0957	-0.471	....	14.47
6	-6.978	-8.466	-9.1351	-0.421	....	19.141
7	-9.131	-11.44	-10.376	-0.804	....	27.688
8	-3.751	-4.466	-5.0922	-0.315	....	9.7326
....	....	....	....	....	....	....
16	-11.11	-13.83	-13.057	-0.826	....	33.1116

Nilai γ atau *learning rate* diperoleh dari rumus $\gamma = \frac{0.01}{\max_i D_{ij}} = \frac{0.01}{65.39558898} = 0.000152916$. Setelah mendapatkan nilai γ, selanjutnya menghitung nilai $E_i = \sum_{j=1}^{n} \alpha_j D_{ij}$. Untuk proses perhitungan bisa dilihat dibawah ini dan hasil perhitungan bisa dilihat pada Tabel 4.12.

$$E_1 = \begin{pmatrix} (3.955336*0) + (4.82976999*0) + (4.821225731*0) + \\ (9.130799968*0) + ... + (-11.10934487*0) \end{pmatrix} = 0 ....., \text{dst.}$$

Tabel 4.12 Hasil Perhitungan E_i.

$\alpha_j D_{ij}$	1	2	3	4	5	6	7	8	....	16	E_i
1	0	0	0	0	0	0	0	0	....	0	0
2	0	0	0	0	0	0	0	0	....	0	0
3	0	0	0	0	0	0	0	0	....	0	0
4	0	0	0	0	0	0	0	0	....	0	0
5	0	0	0	0	0	0	0	0	....	0	0
....	....	....	....	....	....	....	....	....	....	....	....
16	0	0	0	0	0	0	0	0	....	0	0

Hasil perhitungan $\delta\alpha_i = \min\{\max[\gamma(1-E_i), -\alpha_i], C-\alpha_i\}$ dilihat pada Tabel 4.13 dan untuk proses perhitungan bisa dilihat di bawah ini.

$\delta\alpha_1 = MIN(MAX(0.000153*(1-0), -0), 1-0) = 0.000152916$

$\delta\alpha_2 = MIN(MAX(0.000153*(1-0), -0), 1-0) = 0.000152916$

....

$\delta\alpha_{16} = MIN(MAX(0.000153*(1-0), -0), 1-0) = 0.000152916$

Tabel 4.13 Hasil Perhitungan $\delta\alpha_i$.

	1	2	3	.....	16
$\delta\alpha_i$	0.000152916	0.000152916	0.000152916	.....	0.000152916

Kemudian hitung nilai $\alpha_i = \alpha_i + \delta\alpha_i$ dapat dilihat pada Tabel 4.14 di bawah ini dan untuk proses perhitungan juga bisa dilihat pada persamaan di bawah ini.

$\alpha_1 = 0 + 0.000152916 = 0.000152916$

$\alpha_2 = 0 + 0.000152916 = 0.000152916$

....

$\alpha_{16} = 0 + 0.000152916 = 0.000152916$

Tabel 4.14 Hasil Perhitungan α_i.

	1	2	3	.....	16
α_i	0.000152916	0.000152916	0.000152916	.....	0.000152916

Kemudian dilakukan pengecekan nilai maksimum $\delta\alpha_i$, diperoleh nilai maksimum adalah 0.000152916. Karena nilai $\delta\alpha_i > \varepsilon$, maka dilakukan iterasi 1.

Iterasi 1

Untuk proses perhitungan bisa dilihat di bawah ini dan hasil perhitungan bisa dilihat pada Tabel 4.15.

$$E_1 = \begin{pmatrix} (0.000152916*3.955336343) + (0.000152916*4.82976999) \\ + (0.000152916*4.821225731) + ... + (0.000152916* \\ - 6.012044047) \end{pmatrix} = -11.10934487$$

....,dst.

Tabel 4.15 Hasil Perhitungan E_i.

$\alpha_j D_{ij}$	1	2	3	....	16	E_i
1	0.000604832	0.000738547	0.00073724	....	-0.001698791	-0.011552982
2	0.000738547	0.000908233	0.000889045	....	-0.002115456	-0.014273551
3	0.00073724	0.000889045	0.000959846	....	-0.001996643	-0.013980279
4	6.22093E-05	7.00383E-05	5.75525E-05	....	-0.000126365	-0.000902707
5	-0.00077295	-0.000949368	-0.000932123	....	0.00221269	0.014956097
....	....	....	....	....	....	....
16	-0.00169879	-0.002115456	-0.001996643	....	0.00506327	0.033760925

Hasil perhitungan $\delta\alpha_i = \min\{\max[\gamma(1-E_i), -\alpha_i], C-\alpha_i\}$ dilihat pada Tabel 4.16 dan untuk proses perhitungan bisa dilihat di bawah ini.

$\delta\alpha_1 = MIN(MAX(0.000153 * (1-(-0.0116), -0.000153), 1-0.000153) = 0.000155$

$\delta\alpha_2 = MIN(MAX(0.000153 * (1-(-0.0143), -0.000153), 1-0.000153) = 0.000155$

....

$\delta\alpha_{16} = MIN(MAX(0.000153 * (1-0.03376), -0.000153), 1-0.000153) = 0.000148$

Tabel 4.16 Hasil Perhitungan $\delta\alpha_i$.

	1	2	3	....	16
$\delta\alpha_i$	0.000154682	0.000155098	0.000155053	....	0.000147753

Kemudian hitung nilai $\alpha_i = \alpha_i + \delta\alpha_i$ dapat dilihat pada Tabel 4.17 di bawah ini dan untuk proses perhitungan juga bisa dilihat pada persamaan di bawah ini.

$\alpha_1 = 0.000154682 + 0.000152916 = 0.000307598$

$\alpha_2 = 0.000155098 + 0.000152916 = 0.000308014$

....

$\alpha_{16} = 0.000147753 + 0.000152916 = 0.000300668$

Tabel 4.17 Hasil Perhitungan α_i.

	1	2	3	....	16
α_i	0.000307598	0.000308014	0.000307969	....	0.00030068

Kemudian dilakukan pengecekan nilai maksimum $\delta\alpha_i$, diperoleh nilai maksimum adalah 0.000155098. Karena nilai $\delta\alpha_i > \varepsilon$, maka dilakukan iterasi 2. Iterasi dihentikan karena batas maksimum iterasi telah tercapai yaitu 1. Selanjutnya didapatkan nilai *Support Vector* yaitu nilai $\alpha_i >$ *Threshold* yang ditentukan diawal. Karena semua nilai $\alpha_i > 0$, maka semua termasuk dalam *Support Vector*. Nilai *Support Vector* dapat dilihat pada Tabel 4.18 di bawah ini.

Tabel 4.18 Nilai *Support Vector* Level 1.

	1	2	3	....	16
Ai	0.000307598	0.000308014	0.000307969	....	0.00030068
Label	1	1	1	....	-1

Setelah mendapatkan nilai *Support Vector*, langkah berikutnya adalah menghitung nilai *b*. Nilai *b* adalah nilai bias atau nilai pergeseran *hyperplane* yang terbentuk. Nilai *b* dihitung dengan rumus.

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

Dibawah ini merupakan hasil perhitungan matriks *kernel polynomial degree 2* dan perhitungan $w.x^{+}$ dan $w.x^{-}$ pada Tabel 4.19. Untuk pemilihan nilai $K(x_i, x^{+})$

yaitu dengan mencari nilai α_i terbesar untuk kelas positif dan nilai $K(x_i, x^-)$ yaitu dengan mencari nilai α_i terbesar untuk kelas negatif. Nilai $K(x_i,x^+)$ yaitu pada data ke-2 dengan α_i terbesar yaitu 0.000308014 dan nilai $K(x_i,x^-)$ yaitu pada data ke-12 α_i terbesar yaitu 0.000304677.

Tabel 4.19 Hasil Perhitungan *Kernel* dan $w.x^+$ dan $w.x^-$.

	$K(x_i,x^+)$	$K(x_i,x^-)$	$\alpha_i y_i K(x_i,x^+)$	$\alpha_i y_i K(x_i,x^-)$
1	4.57976999	2.430136805	0.001408727	0.000747504
2	5.689445568	2.861479983	0.001752427	0.000881375
3	5.563960136	3.638937225	0.001713526	0.001120679
4	0.208019915	0.01911519	6.36477E-05	5.84866E-06
5	5.95845077	3.01917147	-0.001808652	-0.00091645
6	8.21560478	5.362903574	-0.002486925	-0.00162339
7	11.18766384	4.529106764	-0.003373623	-0.00136575
8	4.215685951	3.188683314	-0.001282371	-0.00096997
9	12.65442566	4.918887223	-0.003808787	-0.00148051
10	6.142579333	3.562408577	-0.001864158	-0.00108112
11	12.13999387	3.560508045	-0.003655241	-0.00107204
12	2.861479983	2.734718088	-0.000871828	-0.00083321
13	4.632289336	1.130341778	-0.001407812	-0.00034353
14	19.13135324	10.14842566	-0.005712237	-0.00303012
15	6.660244884	7.956108466	-0.002018801	-0.00241159
16	13.58415201	6.178792197	-0.004084326	-0.0018578
		Jumlah	-0.027436432	-0.01423002
			TOTAL	-0.04166646

Kemudian didapatkan nilai b yaitu sebagai berikut.

$$b = -\frac{1}{2}\left[\sum_{i=1}^{m}\alpha_i y_i K(x_i, x^+) + \sum_{i=1}^{m}\alpha_i y_i K(x_i, x^-)\right] = -\frac{1}{2} * (-0.04167) = 0.020833229$$

Training Level 2

Selanjutnya untuk proses training Level 2, data yang digunakan adalah data yang termasuk kedalam kelas -1 pada Level 1 yaitu 2, 3, 4. Variabel yang digunakan sama dengan variabel pada Level 1 dan proses perhitungan pun tetap sama seperti level 1, yang membedakan hanya data yang digunakan. Data yang dihitung hanya data pada kelas 2, 3, dan 4. Data yang digunakan bisa di lihat pada Tabel 4.20 di bawah ini.

UNIVERSITAS BRAWIJAYA repository.ub.ac

Tabel 4.20 Data yang di Training pada Level 2.

Pasien ke-	MLII					VI					Kelas	Kelas Level 2
5	0.462857	0.477778	0.50793651	0.52820513	0.523584906	0.501767	0.5	0.498294	0.501695	0.494915254	2	1
6	0.497143	0.466667	0.43915344	0.43589744	0.41509434	0.763251	0.742958	0.709898	0.701695	0.701694915	2	1
7	0.8	0.811111	0.7989418	0.80512821	0.79245283	0.579505	0.573944	0.56314	0.555932	0.566101695	2	1
8	0.228571	0.25	0.25925926	0.29230769	0.311320755	0.611307	0.588028	0.569966	0.549153	0.545762712	2	1
9	0.874286	0.872222	0.88359788	0.88205128	0.853773585	0.55477	0.577465	0.583618	0.589831	0.610169492	3	1
10	0.428571	0.433333	0.44973545	0.46666667	0.462264151	0.575972	0.570423	0.56314	0.559322	0.566101695	3	1
11	1	1	1	1	0.952830189	0.431095	0.440141	0.443686	0.430508	0.430508475	3	1
12	0.057143	0.1	0.14814815	0.17948718	0.202830189	0.586572	0.570423	0.549488	0.532203	0.538983051	3	1
13	0.388571	0.516667	0.67195767	0.85128205	1	0.226148	0.207746	0.197952	0.176271	0.149152542	4	-1
14	0.731429	0.811111	0.88359788	0.94871795	0.966981132	1	0.964789	0.918089	0.874576	0.850847458	4	-1
15	0.028571	0.016667	0.06349206	0.09230769	0.108490566	0.985866	1	1	1	1	4	-1
16	0.822857	0.805556	0.7989418	0.82564103	0.830188679	0.69258	0.690141	0.68942	0.684746	0.705084746	4	-1

Proses perhitungan yang dilakukan sama dengan proses pada Level 1, sehingga didapatkan nilai α_i iterasi ke-1 seperti yang ditunjukkan pada Tabel 4.21 di bawah ini.

Tabel 4.21 Hasil Perhitungan α_i.

	5	6	7	8	9	10	11	....	16
α_i	0.000305338	0.000305245	0.000304823	0.00030556	0.000304676	0.000305357	0.000304617	....	0.00030699

Kemudian didapatkan nilai *b* = -0.0101945.

UNIVERSITAS BRAWIJAYA repository.ub.ac

Training Level 3

Selanjutnya untuk proses training Level 3, data yang digunakan adalah data yang termasuk kedalam kelas +1 pada Level 2 yaitu 2, 3. Variabel yang digunakan sama dengan variabel pada Level 1 dan Level 2. Data yang digunakan yaitu pada Tabel 4.22 di bawah ini.

Tabel 4.22 Data yang di Training pada Level 3.

Pasien ke-	MLII					VI					Kelas	Kelas Level 3
5	0.462857	0.477778	0.50793651	0.52820513	0.523584906	0.501767	0.5	0.498294	0.501695	0.494915254	2	1
6	0.497143	0.466667	0.43915344	0.43589744	0.41509434	0.763251	0.742958	0.709898	0.701695	0.701694915	2	1
7	0.8	0.811111	0.7989418	0.80512821	0.79245283	0.579505	0.573944	0.56314	0.555932	0.566101695	2	1
8	0.228571	0.25	0.25925926	0.29230769	0.311320755	0.611307	0.588028	0.569966	0.549153	0.545762712	2	1
9	0.874286	0.872222	0.88359788	0.88205128	0.853773585	0.55477	0.577465	0.583618	0.589831	0.610169492	3	-1
10	0.428571	0.433333	0.44973545	0.46666667	0.462264151	0.575972	0.570423	0.56314	0.559322	0.566101695	3	-1
11	1	1	1	1	0.952830189	0.431095	0.440141	0.443686	0.430508	0.430508475	3	-1
12	0.057143	0.1	0.14814815	0.17948718	0.202830189	0.586572	0.570423	0.549488	0.532203	0.538983051	3	-1

Proses perhitungan yang dilakukan sama dengan proses pada Level 1 dan Level 2, sehingga didapatkan nilai α_i pada iterasi ke-1 seperti yang ditunjukkan pada Tabel 4.23 di bawah ini.

Tabel 4.23 Hasil Perhitungan α_i.

	5	6	7	8	9	10	11	12
α_i	0.000579562	0.000579353	0.000580248	0.000579184	0.000577921	0.000578944	0.000577503	0.000579303

Kemudian didapatkan nilai *b* = 0.003197235.

4.3.6 *Testing* Data

Data yang digunakan untuk proses *testing* adalah data ke- 17,18,19, dan 20. Dibawah ini merupakan data *testing* yang telah dinormalisasi pada Tabel 4.24.

UNIVERSITAS BRAWIJAYA repository.ub.ac

Tabel 4.24 Data *Testing* Hasil Normalisasi.

Pasien ke-	MLII					VI					Kelas
17	0.3486	0.35	0.3439	0.3641	0.3821	0.6042	0.5352	0.4334	0.3153	0.2102	1
18	0	0	0	0	0	0.5654	0.5528	0.5222	0.5017	0.4949	2
19	0.5543	0.5667	0.5767	0.5795	0.5613	0.4629	0.4648	0.4642	0.4678	0.461	3
20	0.5829	0.5722	0.582	0.6051	0.5708	0.7208	0.7113	0.7201	0.7322	0.739	4

Dalam perhitungan data testing dapat menggunakan fungsi :

$$f(x) = w.x + b = \sum_{i=0}^{m} \alpha_i y_i K(x_i, x) + b$$

Masing-masing data *testing* diuji pada level 1 menggunakan fungsi diatas. Berikut ini adalah hasil perhitungan data uji dengan menggunakan *kernel polynomial degree* 2. Berikut ini hasil perhitungan dengan fungsi diatas pada Tabel 4.24 mendapatkan hasil data uji 17, nilai total dibawah dijumlahkan dengan nilai b dari hasil perhitungan SVM untuk setiap level, karena ini level 1 maka dijumlahkan dengan nilai b pada SVM level 1 sehingga mendapatkan nilai *f(x)*= 0.003351 yaitu *sign f(x)*=1 dan data uji 18 nilai *f(x)*= 0.012133 yaitu *sign f(x)*=1. Untuk proses perhitungan pada hasil yang ditunjukkan pada Tabel 4.25 yaitu sebagai berikut.

$$K(x_1, x) = \left((0.38857143 * 0.34857143) + (0.4 * 0.35) + (0.439153 * 0.343915) + ... + (0.450847 * 0.210169)\right)^2 = 2.93454621$$

....,dst.

$$\alpha_1 y_1 K(x_1, x) = 0.000307 * 1 * 4.468735 = 0.001374012$$ ..., dst.

Tabel 4.25 Data *Testing* 17 dan 18.

Data ke-	$K(x_i,x)$	$\alpha_i y_i K(x_i,x)$
1	2.93454621	0.000902292
2	3.61035731	0.001112039
3	3.67592762	0.001132071
4	0.11166384	3.41657E-05
5	3.78136298	-0.00114781
....	....	....
16	8.48026195	-0.00254975
TOTAL		-0.01748254

Data ke-	$K(x_i,x)$	$\alpha_i y_i K(x_i,x)$
1	1.439495	0.000443
2	1.650684	0.000508
3	2.428101	0.000748
4	0	0
5	1.734312	-0.00053
....	....	....
16	3.332745	-0.001
TOTAL		-0.0087

Kemudian kita hitung dengan fungsi data *testing* untuk data uji ke 19 dan 20, dimana data uji 19 mendapatkan hasil *f(x)*= -0.00993 yaitu *sign f(x)*=-1 dan data uji 20 mendapatkan hasil *f(x)*= -0.02816 yaitu *sign f(x)*=-1. Begitu seterusnya hingga mencapai Level 3. Berikut ini hasil f(x) untuk setiap level yang ditunjukkan pada Tabel 4.26, dan 4.27

Pada saat *testing* di Level 1, didapatkan data ke-17-18 mendapatkan nilai *sign f(x)*=+1 sehingga berdasarkan Gambar 4.8 pada pohon biner yang telah dibuat sebelumnya, maka bergerak ke arah kiri. Kemudian pada proses data *testing* Level 2 mendapatan nilai *sign f(x)*=-1 untuk data testing yang tersisa yaitu data testing ke-19 dan 20 yang bergerak ke arah kanan. Sehingga perhitungan data uji hanya sampai Level 2 karena semua data uji telah berhasil menempati kelas nya.

Tabel 4.26 Hasil Data *Testing* Level 1.

Pasien Ke-	f(x) Level 1	sign f(x)	Kelas	Ket
17	0.00335106	+1	1	Normal
18	0.01213326	+1	1	Normal
19	-0.0099257	-1	-	-
20	-0.028158	-1	-	-

Tabel 4.27 Hasil Data *Testing* Level 2.

Pasien Ke-	f(x) Level 2	sign f(x)	Kelas	Ket
19	-0.0032904	-1	4	Ventricular tachycardia
20	-0.0004744	-1	4	Ventricular tachycardia

Dibawah ini merupakan Tabel 4.28 untuk membandingkan hasil klasifikasi dari pakar dan hasil klasifikasi dari sistem. Dari perbandingan hasil klasifikasi tersebut akan dihitung nilai akurasi dengan membandingkan jumlah kelas yang sama antar kelas pakar dan kelas sistem.

Tabel 4.28 Perbandingan Hasil Klasifikasi Sistem.

Pasien Ke-	Kelas Pakar	Kelas Sistem	Keterangan
17	1	1	Benar
18	2	1	Salah
19	3	1	Salah
20	4	4	Benar

Setelah menyelesaikan tahap testing, selanjutnya dihitung nilai akurasi dengan cara membandingkan hasil klasifikasi kelas pakar dengan kelas sistem. Untuk menghitung akurasi dapat dilakukan dengan menggunakan Persamaan 2.22. Sehingga akurasi perhitungan manual ini adalah sebagai berikut.

$$akurasi = \frac{2}{4} x100\% = 50\%$$

4.4 Perancangan Antarmuka Sistem

Perancangan antar muka bertujuan untuk mengambarkan keadaan sebenarnya dari implementasi sistem klasifikasi detak jantung berdasarkan hasil pemeriksaan EKG yang akan dibangun. Implementasi terdiri dari 6 halaman yaitu Halaman *Import* Data, Halaman Data EKG, Halaman *Step*-BDT, Halaman *Step*-Normalisasi, Halaman *Step*-SVM, dan Halaman Evaluasi.

4.4.1 Perancangan Halaman *Import* Data

Halaman *import* data merupakan halaman untuk memasukkan data detak jantung berupa *file* dengan format *file* .csv. Rancangan antarmuka halaman *import* data ditunjukkan pada Gambar 4.29 di bawah ini.

Gambar 4.29 Perancangan Halaman *Import* Data.

Keterangan rancangan halaman antarmuka *Import* Data pada Gambar 4.29 di atas adalah sebagai berikut.

1. Penjelasan untuk poin nomor 1 yaitu *Header* dari aplikasi ini.

2. Penjelasan untuk poin nomor 2 yaitu Menu *Import* Data. Tab menu *Import* Data berwarna lebih gelap yang menandakan bahwa halaman tersebut sedang aktif.
3. Penjelasan untuk poin nomor 3 yaitu *Button Browse file* dengan format *file* .csv.
4. Penjelasan untuk poin nomor 4 yaitu *Textbox* yang menampilkan lokasi atau *path file* yang di *import*.
5. Penjelasan untuk poin nomor 5 yaitu *Button Load* untuk *load* data yang telah di *import*.
6. Penjelasan untuk poin nomor 6 yaitu *Textbox* yang menampilkan bahwa *file* telah berhasil di *load*.
7. Penjelasan untuk poin nomor 7 yaitu gambar jantung manusia.
8. Penjelasan untuk poin nomor 8 yaitu *footer* untuk aplikasi ini.

4.4.2 Perancangan Halaman Data EKG

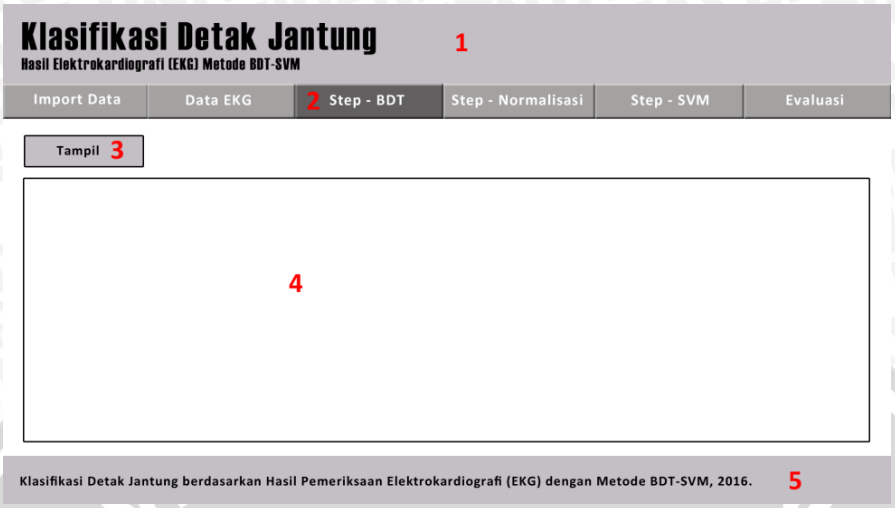
Halaman Data EKG merupakan halaman untuk menampilkan data detak jantung hasil pemeriksaan EKG yang telah berhasil di load ke dalam tabel. Rancangan antarmuka halaman data EKG ditunjukkan pada Gambar 4.30 di bawah ini.

Gambar 4.30 Perancangan Halaman Data EKG.

Keterangan rancangan halaman antarmuka Data EKG pada Gambar 4.30 di atas adalah sebagai berikut.

1. Penjelasan untuk poin nomor 1 yaitu *Header* dari aplikasi ini.
2. Penjelasan untuk poin nomor 2 yaitu Menu Data EKG. Tab menu Data EKG berwarna lebih gelap yang menandakan bahwa halaman tersebut sedang aktif.
3. Penjelasan untuk poin nomor 3 yaitu Tabel yang menampilkan data yang telah di *load*.
4. Penjelasan untuk poin nomor 4 yaitu *footer* untuk aplikasi ini.

4.4.3 Perancangan Halaman *Step*-BDT



Gambar 4.31 Perancangan Halaman *Step*-BDT.

Halaman *Step*-BDT merupakan halaman untuk menampilkan hasil perhitungan data detak jantung hasil pemeriksaan EKG dengan menggunakan algoritma *Binary Decision Tree* dari data yang berhasil di *load*. Rancangan antarmuka halaman *Step*-BDT ditunjukkan pada Gambar 4.31 di atas ini. Keterangan rancangan halaman antarmuka Step-BDT pada Gambar 4.31 di atas adalah sebagai berikut.

1. Penjelasan untuk poin nomor 1 yaitu *Header* dari aplikasi ini. *Header* ini adalah untuk menampilkan judul dari sistem ini yaitu Klasifikasi Detak Jantung Hasil Pemeriksaan EKG atau Elektrokardiografi dengan Metode *BDT-SVM.*
2. Penjelasan untuk poin nomor 2 yaitu Menu *Step*-BDT. Tab menu *Step*-BDT berwarna lebih gelap yang menandakan bahwa halaman tersebut sedang aktif.
3. Penjelasan untuk poin nomor 3 yaitu Button yang digunakan dalam memproses untuk menampilkan hasil perhitungan.
4. Penjelasan untuk poin nomor 4 yaitu menampilkan hasil perhitungan.
5. Penjelasan untuk poin nomor 5 yaitu *footer* untuk aplikasi ini. *Footer* dari aplikasi ini yaitu Klasifikasi Detak Jantung Berdasarkan Hasil Pemeriksaan Elektrokardiografi (EKG) denan Metode *BDT-SVM*, 2016.

4.4.4 Perancangan Halaman *Step*-Normalisasi

Halaman *Step*-Normalisasi merupakan halaman untuk menampilkan hasil perhitungan normalisasi data detak jantung hasil pemeriksaan EKG dengan menggunakan algoritma *Min-Max Normalization* dari data yang berhasil di *load* ke sistem. Dimana nantinya akan menampilkan hasil data *training* dan data *testing.* Rancangan antarmuka halaman *Step*-Normalisasi ditunjukkan pada Gambar 4.32 di bawah ini.

Gambar 4.32 Perancangan Halaman *Step*-Normalisasi.

Keterangan rancangan halaman antarmuka *Step*-Normalisasi pada Gambar 4.32 di atas adalah sebagai berikut.

1. Penjelasan untuk poin nomor 1 yaitu *Header* dari aplikasi ini.
2. Penjelasan untuk poin nomor 2 yaitu Menu *Step*-Normalisasi. Tab menu *Step*-Normalisasi berwarna lebih gelap yang menandakan bahwa halaman tersebut sedang aktif.
3. Penjelasan untuk poin nomor 3 yaitu Button yang digunakan dalam memproses untuk menampilkan hasil perhitungan.
4. Penjelasan untuk poin nomor 4 yaitu menampilkan hasil perhitungan.
5. Penjelasan untuk poin nomor 5 yaitu *footer* untuk aplikasi ini.

4.4.5 Perancangan Halaman *Step*-SVM

Halaman *Step*-SVM merupakan halaman untuk menampilkan hasil perhitungan menggunakan algoritma *Support Vector Machine* dari data detak jantung hasil pemeriksaan EKG yang telah dinormalisasi. Rancangan antarmuka halaman *Step*-SVM ditunjukkan pada Gambar 4.33 di bawah ini.

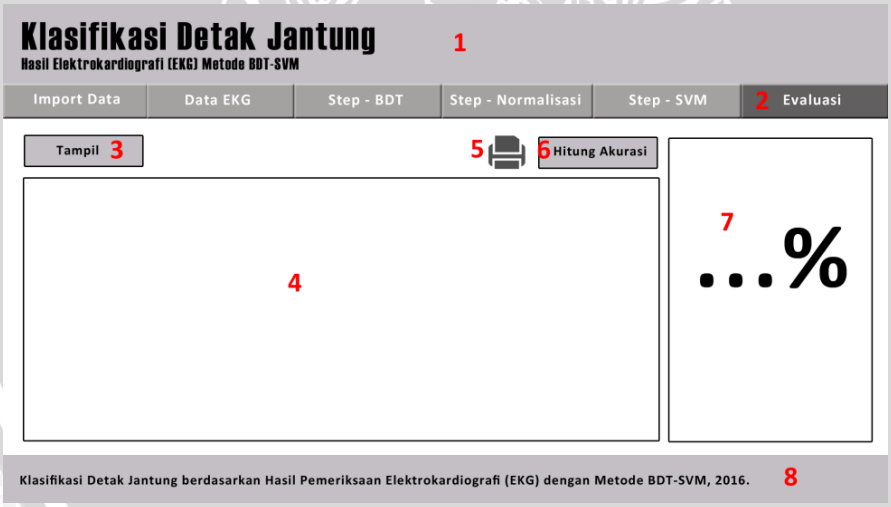
Keterangan rancangan halaman antarmuka *Step*-SVM pada Gambar 4.33 di bawah adalah sebagai berikut.

1. Penjelasan untuk poin nomor 1 yaitu *Header* dari aplikasi ini. *Header* ini adalah untuk menampilkan judul dari sistem ini yaitu Klasifikasi Detak Jantung Hasil Pemeriksaan EKG atau Elektrokardiografi dengan Metode *BDT-SVM.*
2. Penjelasan untuk poin nomor 2 yaitu Menu *Step*-SVM. Tab menu *Step*-SVM berwarna lebih gelap yang menandakan bahwa halaman tersebut sedang aktif.
3. Penjelasan untuk poin nomor 3 yaitu Button yang digunakan dalam memproses untuk menampilkan hasil perhitungan.
4. Penjelasan untuk poin nomor 4 yaitu menampilkan hasil perhitungan.
5. Penjelasan untuk poin nomor 5 yaitu *footer* untuk aplikasi ini.

Gambar 4.33 Perancangan Halaman *Step*-SVM.

4.4.6 Perancangan Halaman Evaluasi

Halaman Evaluasi merupakan halaman untuk menampilkan hasil klasifikasi dari perhitungan menggunakan algoritma *Binary Decision Tree-Support Vector Machine* dari data detak jantung hasil pemeriksaan EKG. Rancangan antarmuka halaman Evaluasi ditunjukkan pada Gambar 4.34 di bawah ini.



Gambar 4.34 Perancangan Halaman Evaluasi.

Keterangan rancangan halaman antarmuka Evaluasi pada Gambar 4.34 di atas adalah sebagai berikut.

1. Penjelasan untuk poin nomor 1 yaitu *Header* dari aplikasi ini.
2. Penjelasan untuk poin nomor 2 yaitu Menu Evaluasi. Tab menu Evaluasi berwarna lebih gelap yang menandakan bahwa halaman tersebut sedang aktif.
3. Penjelasan untuk poin nomor 3 yaitu *Button* yang digunakan dalam memproses untuk menampilkan hasil perhitungan.

4. Penjelasan untuk poin nomor 4 yaitu menampilkan hasil perhitungan.
5. Penjelasan untuk poin nomor 5 yaitu *Button* untuk mencetak hasil klasifikasi ke dalam format .pdf.
6. Penjelasan untuk poin nomor 6 yaitu *Button* untuk menghitung nilai akurasi dari hasil klasifikasi.
7. Penjelasan untuk poin nomor 7 yaitu *Textbox* untuk menampilkan nilai akurasi dari hasil klasifikasi.
8. Penjelasan untuk poin nomor 8 yaitu *footer* untuk aplikasi ini.

4.5 Perancangan Pengujian Sistem

Pada tahap perancangan pengujian dilakukan untuk menguji validasi dari perangkat lunak yang telah dibuat. Sistem yang baik adalah sistem yang memiliki sedikit *error* saat diijalankan. Proses pengujian yaitu sebagai berikut.

1. Pengujian Variasi Fitur atau Parameter
2. Pengujian Rasio Perbandingan Data Latih dan Data Uji.
3. Pengujian Normalisasi
4. Pengujian terhadap Kernel.
5. Pengujian terhadap jumlah iterasi SVM.
6. Pengujian terhadap parameter λ (*Lambda*).
7. Pengujian terhadap parameter *C* (*Complexity*).
8. Pengujian terhadap parameter ε (*Epsilon*).
9. Pengujian terhadap parameter γ (*Gamma*).

4.5.1 Pengujian Variasi Fitur atau Parameter

Skenario pengujian variasi jumlah fitur atau parameter dilakukan untuk mengetahui pengaruh banyaknya fitur yang digunakan terhadap akurasi sistem. Variasi fitur ini terdiri dari 7202 (gabungan MLII dan VI selama 20 detik), 3601 (MLII atau VI selama 10 detik), dan 2161 (MLII atau VI selama 6 detik). Pengujian variasi fitur atau parameter ditunjukkan pada Tabel 4.29.

Tabel 4.29 Perbandingan Variasi Fitur atau Parameter.

No.	Rasio Data (%)	Percobaan ke-*i*					Rata-Rata Nilai Akurasi (%)
		1	2	3	4	5	
1	7202 (MLII dan VI)						
2	3601 (MLII)						
3	3601 (VI)						
4	2161 (MLII)						
5	2161 (VI)						

4.5.2 Pengujian Rasio Perbandingan Jumlah Data Latih dan Data Uji

Skenario pengujian perandingan jumlah data latih dan data uji dilakukan untuk mengetahui pengaruh banyaknya data yang digunakan terhadap akurasi sistem. Pengujian variasi jumlah data ditunjukkan pada Tabel 4.30.

Tabel 4.30 Perbandingan Rasio Data.

No.	Rasio Data (%)	Percobaan ke-*i*					Rata-Rata Nilai Akurasi (%)
		1	2	3	4	5	
1	90 : 10						
2	80 : 20						
3	70 : 30						
4	60 : 40						
5	50 : 50						
6	40 : 60						
7	30 : 70						
8	20 : 80						
9	10 : 90						

4.5.3 Pengujian Normalisasi

Skenario pengujian normalisasi dilakukan untuk mengetahui apakah hasil akurasi terbaik jika data di normalisasi atau jika data tidak dinormalisasi. Berikut ini adalah Tabel 4.31 skenario pengujian normalisasi.

Tabel 4.31 Pengujian Normalisasi.

No.	Jenis-Jenis Kernel	Percobaan ke-*i*					Rata-Rata Nilai Akurasi (%)
		1	2	3	4	5	
1	*Normalisasi*						
2	*Tanpa Normalisasi*						

4.5.4 Pengujian Kernel

Skenario pengujian kernel dilakukan untuk mengetahui kernel terbaik dan pengaruh terhadap nilai akurasi. Proses pengujian yang akan dilakukan yaitu dengan memasukkan 6 fungsi kernel berbeda ke dalam sistem yaitu, *kernel linear, kernel polynomial degree d, kernel invers multi quadratic, kernel laplacian* , *kernel RBF,* dan *kernel cauchy.* Berikut ini adalah Tabel 4.32 skenario pengujian terhadap jumlah iterasi SVM.

Tabel 4.32 Pengujian Kernel.

No.	Jenis-Jenis Kernel	Percobaan ke-*i*					Rata-Rata Nilai Akurasi (%)
		1	2	3	4	5	
1	*Kernel Linear*						
2	*Kernel Polynomial Degree D*						
3	*Kernel Invers Multi Quadratic*						
4	*Kernel Laplacian*						
5	*Kernel RBF*						

6	*Kernel Cauchy*						

4.5.5 Pengujian Terhadap Jumlah Iterasi *SVM*

Skenario pengujian jumlah iterasi *SVM* dilakukan untuk mengetahui pengaruh jumlah iterasi terhadap nilai akurasi. Proses pengujian yang akan dilakukan yaitu dengan memasukkan nilai iterasi dengan jumlah 100, 500, 1000, 2000, 5000, 10000, 50000, dan 100000. Berikut ini adalah Tabel 4.33 skenario pengujian terhadap jumlah iterasi *SVM*.

Tabel 4.33 Perbandingan Jumlah Iterasi *SVM*.

No.	Jumlah Iterasi SVM	Percobaan ke-*i*					Rata-Rata Nilai Akurasi (%)
		1	2	3	4	5	
1	50						
2	100						
3	500						
4	1000						
5	2000						
6	3000						
7	5000						
8	10000						

4.5.6 Pengujian Terhadap Parameter λ (*Lambda*)

Skenario pengujian terhadap nilai λ dilakukan untuk mengetahui pengaruh nilai dari parameter λ terhadap nilai akurasi. Proses pengujian yang akan dilakukan yaitu dengan memasukkan nilai dari parameter λ dengan nilai 0.1, 0.5, 100, 1000, dan 5000000. Berikut ini adalah Tabel 4.34 skenario pengujian terhadap parameter λ.

Tabel 4.34 Perbandingan Parameter λ.

No.	Nilai λ	Percobaan ke-*i*					Rata-Rata Nilai Akurasi (%)
		1	2	3	4	5	
1	0.1						
2	0.5						
3	100						
4	1000						
5	5000000						

4.5.7 Pengujian Terhadap Parameter *C* (*Complexity*)

Skenario pengujian nilai *C* dilakukan untuk mengetahui pengaruh nilai dari parameter *C* terhadap nilai akurasi. Proses pengujian yang akan dilakukan yaitu dengan memasukkan nilai dari parameter C dengan interval [1,10], [10,20], [20,30], [30,40], dan [40,50], [50,60], [60,70], [70,80], [80,100]. Berikut ini adalah Tabel 4.35 skenario pengujian terhadap parameter *C*.

Tabel 4.35 Perbandingan Parameter *C*.

No.	Interval *C*	Percobaan ke-*i*					Rata-Rata Nilai Akurasi (%)
		1	2	3	4	5	
1	[1,10]						
2	[10,20]						
3	[20,30]						
4	[30,40]						
5	[40,50]						
6	[50,60]						
7	[60,70]						
8	[70,80]						
9	[80,100]						

4.5.8 Pengujian Terhadap Parameter *ε* (*Epsilon*)

Skenario pengujian nilai *ε* dilakukan untuk mengetahui pengaruh nilai dari parameter *ε* terhadap nilai akurasi. Proses pengujian yang akan dilakukan yaitu dengan memasukkan nilai dari parameter *ε* dengan interval 0.1, 0.01, 0.001, 0.0001, dan 0.00001. Berikut ini adalah Tabel 4.36 skenario pengujian terhadap parameter *ε*.

Tabel 4.36 Perbandingan Parameter *ε*.

No.	Interval *ε*	Percobaan ke-*i*					Rata-Rata Nilai Akurasi (%)
		1	2	3	4	5	
1	0.1						
2	0.01						
3	0.001						
4	0.0001						
5	0.00001						

4.5.9 Pengujian Terhadap Parameter *Konstanta ɣ*

Skenario pengujian nilai *ɣ* dilakukan untuk mengetahui pengaruh nilai dari parameter *ɣ* terhadap nilai akurasi. Proses pengujian yang akan dilakukan yaitu dengan memasukkan nilai dari parameter *ɣ* dengan interval [1,2], [2,4], [4,6], [6,8], [8,10]. Berikut ini adalah Tabel 4.37 skenario pengujian terhadap parameter *ɣ*.

Tabel 4.37 Perbandingan Parameter *ɣ*.

No.	Interval *ε*	Percobaan ke-*i*					Rata-Rata Nilai Akurasi (%)
		1	2	3	4	5	
1	[1,2]						
2	[2,4]						
3	[4,6]						

4	[6,8]						
5	[8,10]						

4.6 Pengambilan Keputusan

Tahap pengambilan keputusan akan dilakukan setelah penelitian ini selesai dilaksanakan. Tahap ini berisi tentang kesimpulan dan saran yang bertujuan untuk mengambangkan sistem menjadi lebih baik serta mampu memberikan kekurangan-kekurangan dari penelitian ini.