# BAB 4 PERANCANGAN

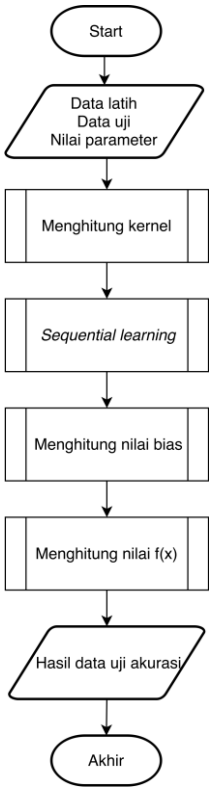
Pada bab ini akan dibahas perancangan basis data, perancangan antarmuka, dan yang terakhir adalah perancangan uji coba dan evaluasi.

## 4.1 Formulasi SVM

Proses klasifikasi pendonor darah pada *dataset* RFMTC dengan menggunakan metode SVM adalah sebagai berikut:

1. Menentukan nilai parameter
2. Menghitung kernel
3. *Sequential learning*
   a. Menghitung matriks $[D]_{ij}$
   b. Iterasi:
      - Menghitung nilai $E_i$
      - Menghitung nilai $\delta\alpha_i$
      - Menghitung $\alpha_i$ baru
5. Menghitung nilai bias
6. Menghitung nilai fungsi dari $f(x)$

Diagram alir (*Flowchart*) untuk perhitungan metode SVM berupa perhitungan kernel, *sequential learning*, menghitung bias, dan menghitung nilai $f(x)$ ditunjukkan pada Gambar 4.1




**Gambar 4.1 Alur SVM**

Pada Gambar 4.1 menjelaskan proses *Support Vector Machine* (SVM)

1. Melakukan *input dataset* ternormalisasi
2. Melakukan perhitungan kernel linier SVM
3. Melakukan perhitungan data *training* SVM, dengan metode *sequential learning*.
4. Menghitung nilai bias
5. Menghitung nilai f(x) dari data uji dan data *training*
6. Menghitung nilai Akurasi

*Output* yang dihasilkan berupa Hasil *Class* dan nilai Akurasi.

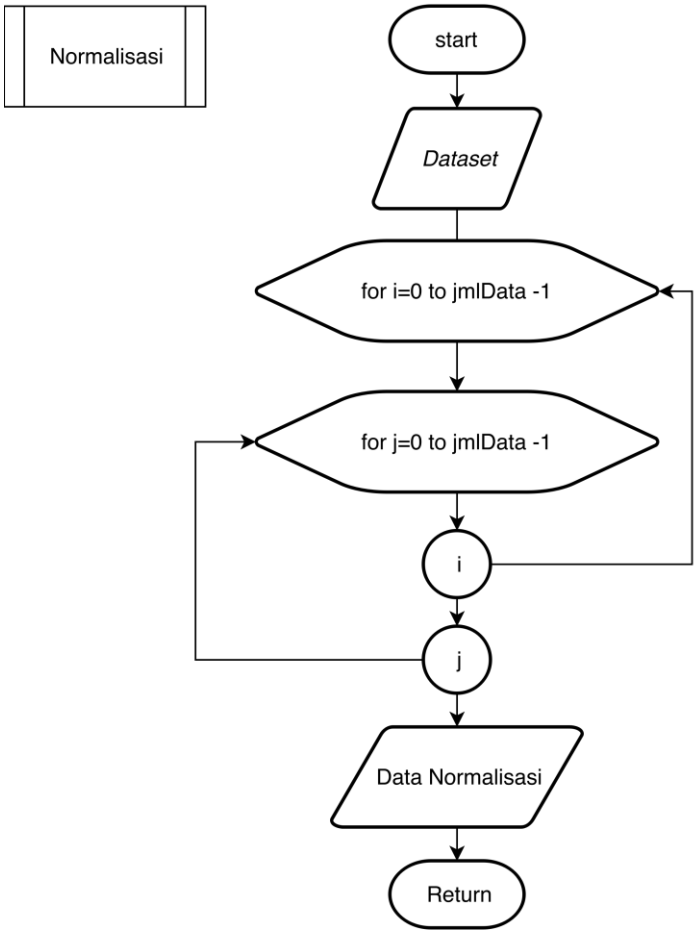
Pada penelitian ini digunakan 10 data *training*, dimana 5 data potensial menyumbangkan darahnya dan 5 lainnya tidak. Ditunjukkan pada Tabel 4.1.

**Tabel 4.1 Data *Training***

| No | *Recency* | *Frequency* | *Monetary* | *Time* | *Churn probability* |
|---|---|---|---|---|---|
| **1** | 2 | 50 | 12500 | 98 | 1 |
| **2** | 0 | 13 | 3250 | 28 | 1 |
| **3** | 1 | 16 | 4000 | 35 | 1 |
| **4** | 2 | 20 | 5000 | 45 | 1 |
| **5** | 2 | 9 | 2250 | 22 | 1 |
| **6** | 1 | 24 | 6000 | 77 | -1 |
| **7** | 4 | 4 | 1000 | 4 | -1 |
| **8** | 1 | 12 | 3000 | 35 | -1 |
| **9** | 4 | 23 | 5750 | 58 | -1 |
| **10** | 1 | 13 | 3250 | 47 | -1 |

Dengan data training diatas yang sudah di normalisasi berikut penjelasan dari tiap langkah perhitungan manual.

### 4.1.2 Normalisasi Dataset




**Gambar 4.2 Alur Normalisasi Dataset**

Pada Gambar 4.2 dijelaskan tahapan dari normalisasi *dataset*. Berikut adalah penjelasan dari tahapan-tahapan dari proses normalisasi.

1. Meng-*input*-kan *dataset* awal pendonor dayah yang akan digunakan.
2. Perulangan terhadap jumlah data untuk proses perhitungan normalisasi.
3. Menghitung nilai normalisasi.
4. *Output* yang dihasilkan adalah data yang telah ternormalisasi.

Pada perhitungan normalisasi rumus yang digunakan yaitu:

Xbaru = (((Xlama – Minlama)/(Maxlama-Minlama))*(Maxbaru-Minbaru))+Min baru

Sebelum masuk perhitungan normalisasi berikan nilai max baru dan nilai min baru. Pada penelitian ini menggunakan nilai max baru 1 dan min baru 0.

Berikut contoh perhitungan normalisasi.

X = (((2 – 0)/(4 – 0))*(1-0))+0

= 0.5

Untuk semua data dihitung berdasarkan atribut yang sama. Hasil ditunjukkan pada Tabel 4.2

**Tabel 4.2 Data Normalisasi**

| No | *Recency* | *Frequency* | *Monetary* | *Time* | *Churn probability* |
|---|---|---|---|---|---|
| **1** | 0.5 | 1 | 1 | 1 | 1 |
| **2** | 0 | 0.195652174 | 0.195652174 | 0.255319149 | 1 |
| **3** | 0.25 | 0.260869565 | 0.260869565 | 0.329787234 | 1 |
| **4** | 0.5 | 0.347826087 | 0.347826087 | 0.436170213 | 1 |
| **5** | 0.5 | 0.108695652 | 0.108695652 | 0.191489362 | 1 |
| **6** | 0.25 | 0.434782609 | 0.434782609 | 0.776595745 | -1 |
| **7** | 1 | 0 | 0 | 0 | -1 |
| **8** | 0.25 | 0.173913043 | 0.173913043 | 0.329787234 | -1 |
| **9** | 1 | 0.413043478 | 0.413043478 | 0.574468085 | -1 |
| **10** | 0.25 | 0.195652174 | 0.195652174 | 0.457446809 | -1 |

### 4.1.3 Menentukan Nilai Parameter

Menentukan parameter dilakukan secara acak, tapi harus memiliki nilai lebih dari nol. Pada penilitian ini parameternya adalah *lambda* (*λ*), *gamma* (*γ*), *Complexity* (*C*), Epsilon (ε), Alpha (α) ditunjukkan pada Tabel 4.3.

**Tabel 4.3 Dataset parameter terpilih**

| ***λ*** | ***γ*** | ***C*** | **ε** | **α** |
|---|---|---|---|---|
| 0.5 | 0.5 | 1 | 0.001 | 0.5 |

### 4.1.4 Menghitung Nilai Kernel

Proses perhitungan kernel dilakukan setelah data dimasukkan ke dalam sistem. Data merupakan linier maka fungsi kernel yang digunakan adalah kernel linier dengan Persamaan 2.19

$$K(x, y) = x \cdot y$$

Sebelumnya data di-*trasnpose* karena menggunakan matriks A x $A^T$

**Tabel 4.4 Transpose Data**

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| **R** | 0.5 | 0 | 0.25 | 0.5 | 0.5 | 0.25 | 1 | 0.25 | 1 | 0.25 |
| **F** | 1 | 0.195652174 | 0.260869565 | 0.347826087 | 0.108695652 | 0.434782609 | 0 | 0.173913043 | 0.413043478 | 0.195652174 |
| **M** | 1 | 0.195652174 | 0.260869565 | 0.347826087 | 0.108695652 | 0.434782609 | 0 | 0.173913043 | 0.413043478 | 0.195652174 |
| **T** | 1 | 0.255319149 | 0.329787234 | 0.436170213 | 0.191489362 | 0.776595745 | 0 | 0.329787234 | 0.574468085 | 0.457446809 |

*1-10 = Jumlah data

*RFMTC = *Regency, Frequency, Monetary, Time, Churn Probability*

Pada metode ini data direpresentasikan lewat perbandingan antara sepasang data. Setiap data akan dibandingan dengan dirinya dan data lainnya. Sebagai contoh data uji berjumlah 5 data maka perbandingannya seperti pada Tabel 4.5:

**Tabel 4.5 Perbandingan Data**

| | A1 | A2 | A3 | A4 | A5 |
|---|---|---|---|---|---|
| A1 | K(A1,A1) | K(A1,A2) | K(A1,A3) | K(A1,A4) | K(A1,A5) |
| A2 | K(A1,A2) | K(A2,A2) | K(A2,A3) | K(A2,A4) | K(A2,A5) |
| A3 | K(A1,A3) | K(A3,A2) | K(A3,A3) | K(A3,A4) | K(A3,A5) |
| A4 | K(A1,A4) | K(A4,A2) | K(A4,A3) | K(A4,A4) | K(A4,A5) |
| A5 | K(A1,A5) | K(A5,A2) | K(A5,A3) | K(A5,A4) | K(A5,A5) |

Berikut contoh perhitungan dengan data A1 x A1

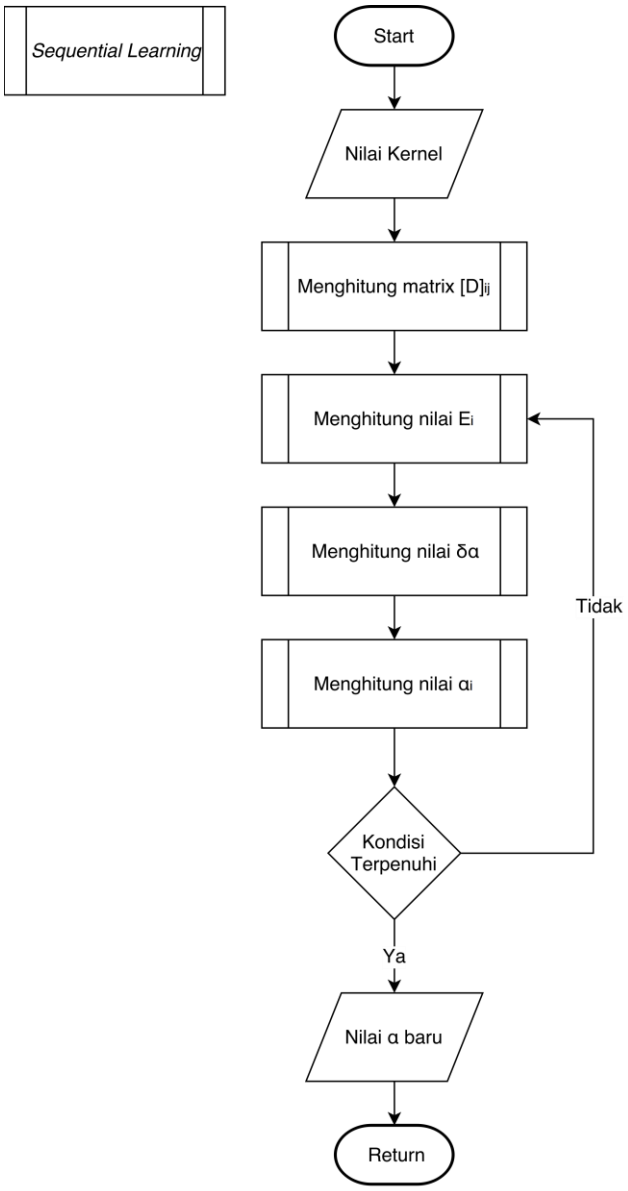
K(A1, A1) = ((0.5*0.5)+(1*1)+(1*1)+(1*1)) = 3.25

Semua data uji dihitung dengan cara yang sama sehingga menghasilkan *dot product* ditunjukkan pada Tabel 4.6:

**Tabel 4.6 Hasil Perhitungan Kernel**

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| **1** | 3.25 | 0.646623497 | 0.976526364 | 1.381822387 | 0.658880666 | 1.771160962 | 0.5 | 0.802613321 | 1.900555042 | 0.973751156 |
| **2** | 0.646623497 | 0.141747414 | 0.186280391 | 0.247468468 | 0.091423982 | 0.36841209 | 0 | 0.152253926 | 0.308298411 | 0.193354476 |
| **3** | 0.976526364 | 0.186280391 | 0.30736548 | 0.450317848 | 0.244861522 | 0.545454463 | 0.25 | 0.2619686 | 0.654953186 | 0.315439513 |
| **4** | 1.381822387 | 0.247468468 | 0.450317848 | 0.682210428 | 0.409136322 | 0.766185398 | 0.5 | 0.389826355 | 1.03790046 | 0.460630532 |
| **5** | 0.658880666 | 0.091423982 | 0.244861522 | 0.409136322 | 0.310297665 | 0.368227782 | 0.5 | 0.22595793 | 0.699796587 | 0.255129279 |
| **6** | 1.771160962 | 0.36841209 | 0.545454463 | 0.766185398 | 0.368227782 | 1.043672784 | 0.25 | 0.469840096 | 1.055297712 | 0.58788357 |
| **7** | 0.5 | 0 | 0.25 | 0.5 | 0.5 | 0.25 | 1 | 0.25 | 1 | 0.25 |
| **8** | 0.802613321 | 0.152253926 | 0.2619986 | 0.389826355 | 0.22595793 | 0.469840096 | 0.25 | 0.231751113 | 0.583119538 | 0.281413048 |
| **9** | 1.900555042 | 0.308298411 | 0.654953186 | 1.03790046 | 0.699796587 | 1.055297712 | 1 | 0.583119538 | 1.671223411 | 0.674414301 |
| **10** | 0.973751156 | 0.193354476 | 0.315439513 | 0.460630532 | 0.255129279 | 0.58788357 | 0.25 | 0.281413048 | 0.674414301 | 0.348317129 |

### *4.1.5 Sequential Learning*

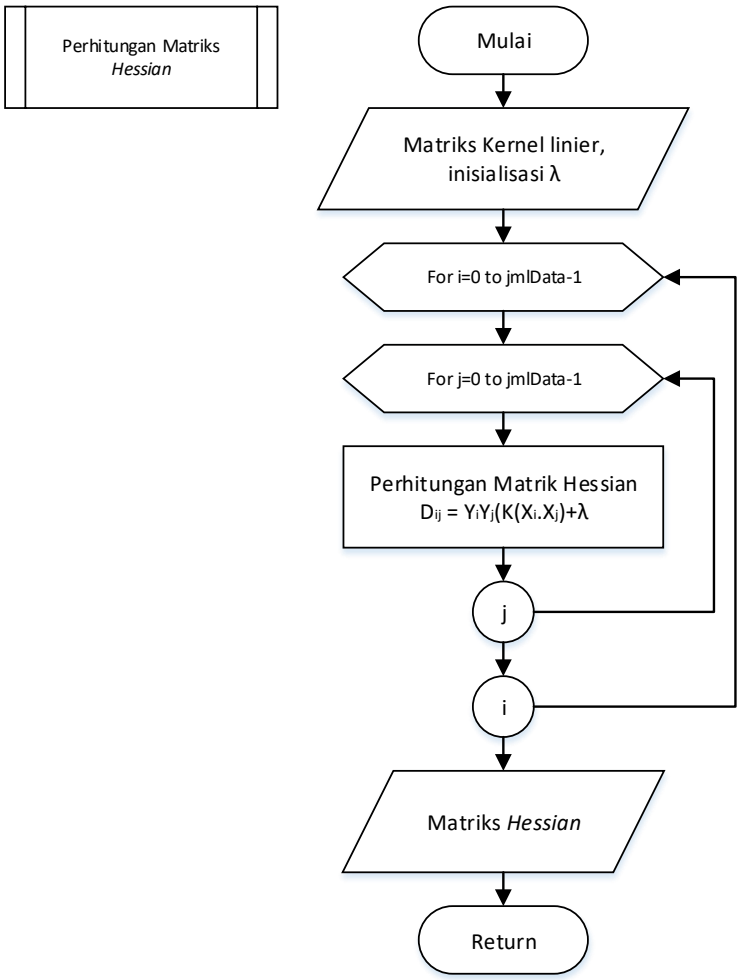



**Gambar 4.3 *Flowchart Sequential Learning***

Pada Gambar 4.3 menjelaskan proses *Sequential Learning*

1. Melakukan *input data kernel*
2. Melakukan perhitungan Matrix $D_{ij}$
3. Melakukan perhitungan nilai $E_i$
4. Menghitung nilai $\alpha_i$
5. Jika iterasi terpenuhi berlanjut jika tidak kembali ke perhitungan nilai $E_i$
6. Output nilai $\alpha$ baru

#### 4.1.5.1 Menghitung Matrix Hessian




**Gambar 4.4 *Flowchart* Matrix Hessian**

Pada Gambar 4.4 menjelaskan proses *Matrix Hessian*

1. Melakukan *input data kernel linier*
2. Melakukan proses perhitungan *Matrix Hessian* menggunakan persamaan 2.15
3. Perulangan terhadap jumlah data
4. Output yg dihasilkan adalah nilai *Matrix Hessian*

Langkah pertama adalah menghitung matrik menggunakan Persamaan 2.15:

$$D_{ij} = y_i y_j(\mathrm{K}(\vec{x}_i.\vec{x}_j)+\lambda^2$$

Keterangan: $D_{ij}$ = elemen matriks hessian ke-ij
$y_i$ = kelas data ke-i
$y_j$ = kelas data ke-j
λ = batas teoritis yang akan diturunkan

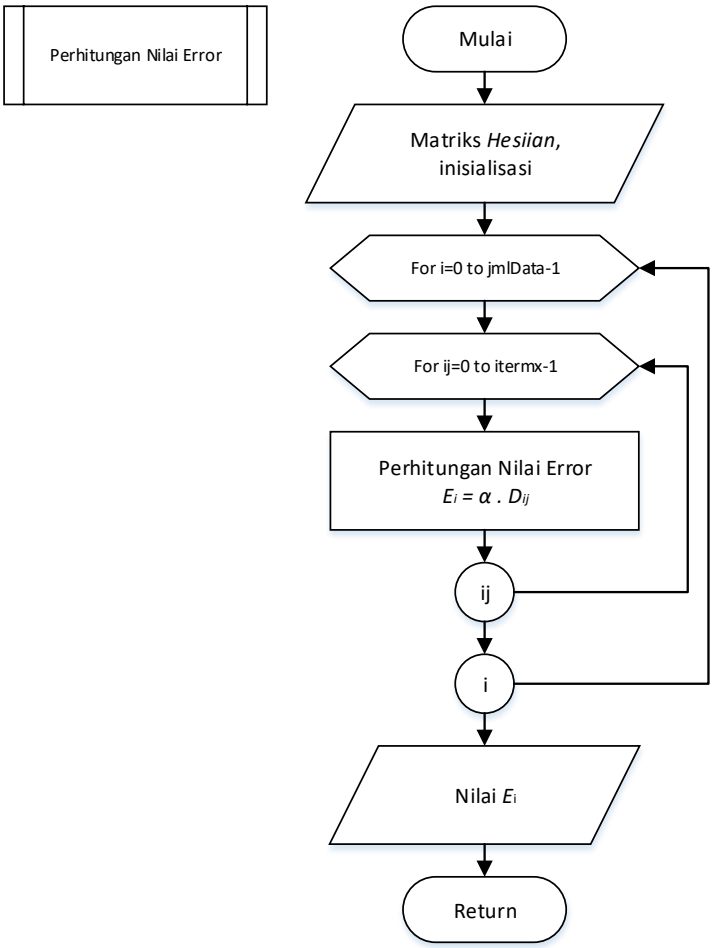
Contoh perhitungan untuk data A1 x A1:
$D_{1.1}$ = 1*1(3.25) + $0.5^2$ = 3.5

Semua data dihitung dengan cara yang sama. Hasil ditunjukan seperti Tabel 4.7:

**Tabel 4.7 Matrix**

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| **1** | 3.5 | 0.896623497 | 1.226526364 | 1.631822387 | 0.908880666 | -2.021160962 | -0.75 | -1.052613321 | -2.15055504 | -1.22375116 |
| **2** | 0.896623497 | 0.391747414 | 0.436280391 | 0.497468468 | 0.341423982 | -0.61841209 | -0.25 | -0.402253926 | -0.55829841 | -0.44335448 |
| **3** | 1.226526364 | 0.436280391 | 0.55736548 | 0.700317848 | 0.494861522 | -0.795454463 | -0.5 | -0.51199686 | -0.90495319 | -0.56543951 |
| **4** | 1.631822387 | 0.497468468 | 0.700317848 | 0.932210428 | 0.659136322 | -1.016185398 | -0.75 | -0.639826355 | -1.28790046 | -0.71063053 |
| **5** | 0.908880666 | 0.341423982 | 0.494861522 | 0.659136322 | 0.560297665 | -0.618227782 | -0.75 | -0.47595793 | -0.94979659 | -0.50512928 |
| **6** | -2.02116096 | -0.61841209 | -0.795454463 | -1.016185398 | -0.618227782 | 1.293672784 | 0.5 | 0.719840096 | 1.305297712 | 0.83788357 |
| **7** | -0.75 | -0.25 | -0.5 | -0.75 | -0.75 | 0.5 | 1.25 | 0.5 | 1.25 | 0.5 |
| **8** | -1.05261332 | -0.402253926 | -0.51199686 | -0.639826355 | -0.47595793 | 0.719840096 | 0.5 | 0.481751113 | 0.833119538 | 0.531413048 |
| **9** | -2.15055504 | -0.558298411 | -0.904953186 | -1.28790046 | -0.949796587 | 1.305297712 | 1.25 | 0.833119538 | 1.921223411 | 0.924414301 |
| **10** | -1.22375116 | -0.443354476 | -0.565439513 | -0.710630532 | -0.505129279 | 0.83788357 | 0.5 | 0.531413048 | 0.924414301 | 0.598317129 |

### 4.1.5.2 Mencari Nilai *Error*




**Gambar 4.5 *Flowchart* Nilai Error**

Pada Gambar 4.5 menjelaskan proses perhitungan nilai *error*

1. Melakukan *input data Matrix Hessian*
2. Melakukan proses perhitungan nilai *error* menggunakan persamaan 2.16
3. Perulangan terhadap jumlah data
4. Output yg dihasilkan adalah nilai *error*

Langkah kedua, inisialisasi α awal yaitu 0.5. Kemudian menghitung nilai *Error* ($E_i$), δα, dan nilai α baru sampai nilai δα kurang dari nilai ε (0.001). Pada penelitian ini menggunakan 2 kali proses pembaruan α. Perhitungan nilai *error* dinyatakan menggunakan Persamaan 2.16.

$$E_i = \sum_{j=1}^{i} \alpha_j \, . \, D_{ij}$$

Keterangan $E_i$ = nilai *error* data ke-i

Sebagai contoh perhitungan data ke 1

$E_1$ = (0.5 * 3.5) + (0.5 * 0.896623497 ) + ..... + (0.5 *-1.22375116) = 0.482886216

Maka didapat keseluruhan nilai $E_1$ pada iterasi 1 seperti Tabel 4.8:

**Tabel 4.8 Iterasi Error**

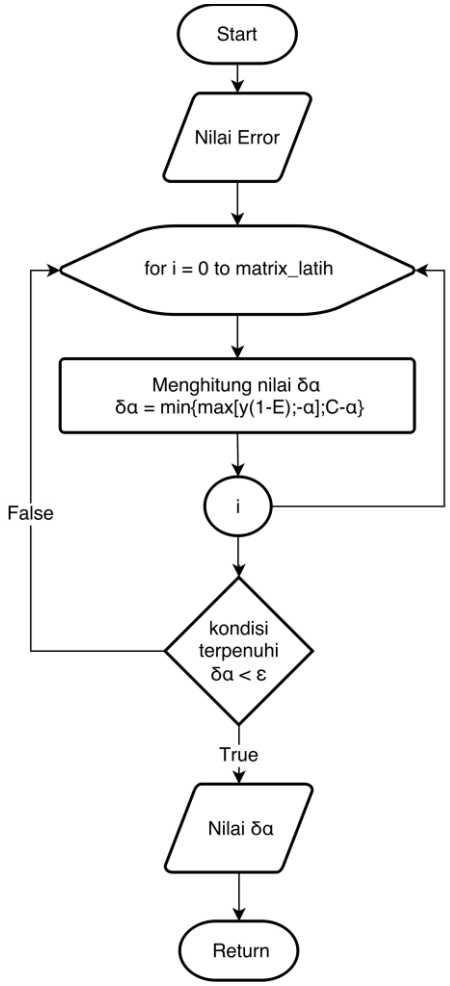
| | Iterasi 1 |
|---|---|
| **1** | 0.482886216 |
| **2** | 0.145612424 |
| **3** | 0.068753792 |
| **4** | 0.008206354 |
| **5** | -0.16725571 |
| **6** | -0.206373266 |
| **7** | 0.5 |
| **8** | -0.008262299 |
| **9** | 0.191275637 |
| **10** | -0.02813845 |

Pada penelitian ini untuk iterasi $E_i$ sebanyak 3 iterasi seperti Tabel 4.9:

**Tabel 4.9 Iterasi Error**

| | Iterasi 1 | Iterasi 2 | Iterasi 3 |
|---|---|---|---|
| **1** | 0.482886216 | 0.399756361 | 1.073585849 |
| **2** | 0.145612424 | 0.145074915 | 0.327162654 |
| **3** | 0.068753792 | -0.000878951 | 0.209383195 |
| **4** | 0.008206354 | -0.131026144 | 0.124226123 |
| **5** | -0.16725571 | -0.320192063 | -0.226698005 |
| **6** | -0.206373266 | -0.098047887 | -0.484622142 |
| **7** | 0.5 | 0.787502442 | 0.820310974 |
| **8** | -0.008262299 | 0.082456668 | -0.088400208 |
| **9** | 0.191275637 | 0.48258791 | 0.202862249 |
| **10** | -0.02813845 | 0.080413266 | -0.12815252 |

#### 4.1.5.3 Mencari Nilai Delta Alpha




**Gambar 4.6 *Flowchart* Delta Alpha**

Pada Gambar 4.6 menjelaskan proses perhitungan delta alpha

1. Melakukan *input data* error
2. Melakukan proses perhitungan nilai delta alpha menggunakan Persamaan 2.17
3. Perulangan terhadap jumlah data
4. Pengecekan kondisi
5. Output yg dihasilkan adalah nilai delta alpha

Selanjutnya mencari nilai δα ditentukan menggunakan Persamaan 2.17.

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

Contoh perhitungan untuk data pertama:
= Min {Max [0.5*(1 – 0.482886216); - 0.5]; 1 – 0.5}
= Min {Max [0.258556892 ; – 0.5]; 0.5}
= Min {0.258556892; 0.5}

= 0.258556892

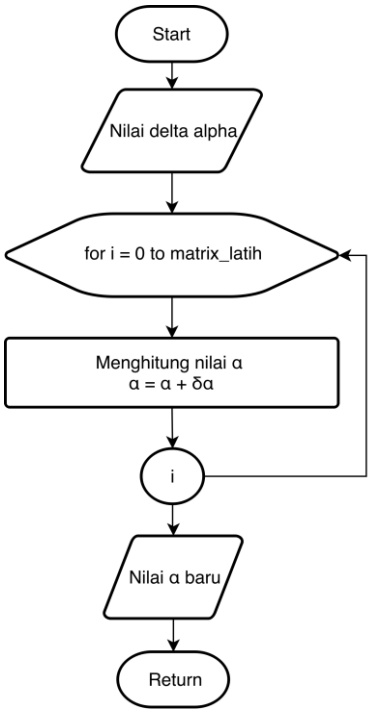
Maka didapat keseluruhan δα pada iterasi 1 ditunjukkan pada Tabel 4.10:

**Tabel 4.10 Iterasi Delta Alpha**

| | Iterasi 1 |
|---|---|
| **1** | 0.25855689 |
| **2** | 0.427193788 |
| **3** | 0.465623104 |
| **4** | 0.495896823 |
| **5** | 0.5 |
| **6** | 0.5 |
| **7** | 0.25 |
| **8** | 0.5 |
| **9** | 0.404362181 |
| **10** | 0.5 |

Nilai maksimum δα adalah 0.5 dan lebih dari epsilon (ε) yaitu 0.001 maka iterasi masih berlanjut.

### 4.1.5.4 Mencari Nilai Alpha Baru




**Gambar 4.7 Flowchard Alpha Baru**

Pada Gambar 4.7 menjelaskan proses perhitungan alpha baru

1. Melakukan *input data* delta alpha
2. Melakukan proses perhitungan nilai alpha baru menggunakan persamaan 2.18
3. Perulangan terhadap jumlah data
4. Output yg dihasilkan adalah nilai alpha baru

Selanjutnya menghitung nilai α baru menggunakan Persamaan 2.18.

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

Contoh perhitungan untuk data pertama

$\alpha_1$ = 0.5 + 0.25855689

$\alpha_1$ = 0.758557

Pada persamaan ini dilakukan 3 kali iterasi. Berikut nilai keseluruhan α (alpha) baru pada Tabel 4.11:

**Tabel 4.11 Iterasi alpha baru**

| | **Iterasi 1** |
|---|---|
| **1** | 0.758557 |
| **2** | 0.927194 |
| **3** | 0.965623 |
| **4** | 0.995897 |
| **5** | 1 |
| **6** | 1 |
| **7** | 0.75 |
| **8** | 1 |
| **9** | 0.904362 |
| **10** | 1 |

Berikut Tabel 4.12 adalah hasil dari pembaruan α dengan iterasi sebanyak 3 kali.

**Tabel 4.12 Iterasi alpa baru**

| | Iterasi 1 | Iterasi 2 | Iterasi 3 |
|---|---|---|---|
| **1** | 0.758557 | 1 | 0.963207 |
| **2** | 0.927194 | 1 | 1 |
| **3** | 0.965623 | 1 | 1 |
| **4** | 0.995897 | 1 | 1 |
| **5** | 1 | 1 | 1 |
| **6** | 1 | 1 | 1 |
| **7** | 0.75 | 0.856249 | 0.946093 |
| **8** | 1 | 1 | 1 |
| **9** | 0.904362 | 1 | 1 |
| **10** | 1 | 1 | 1 |

### 4.1.6 Menghitung Nilai Bias

Nilai bias dibutuhkan untuk menghitung fungsi SVM. Untuk mencari nilai bias ditentukan dengan Persamaan 2.13.

$b = -0.5(< \vec{w}.\vec{x}_{-1} > + < \vec{w}.x_{+1} >)$

Sebelum masuk persamaan bias terlebih dahulu mencari nilai *W* dengan Persamaan 2.14.

$$w = \sum_{i=1}^{N} \alpha_i y_i x_i \qquad w.x^+ \sum_{i=1}^{n} \alpha_i y_i x_i x^+ = \sum_{i=1}^{n} \alpha_i y_i k(x_i, x^+)$$

$w\ x^+ = 1\ x\ 1\ x$ 3.25+ ..... + 1 x 1 x 0.973751156
$w\ x^+ = 0.966108737$

$w\ x^- = 1\ x\ 1\ x$ 1.771160962+ ..... + 1 x 1 x 0.58788357
$w\ x^- = 0.412914684$

Maka diperoleh hasil seperti Tabel 4.13

**Tabel 4.13 Hasil nilai $W^+$ dan $W^-$**

| W $X^+$ | W $X^-$ |
|---|---|
| 3.25 | 1.771160962 |
| 0.646623497 | 0.36841209 |

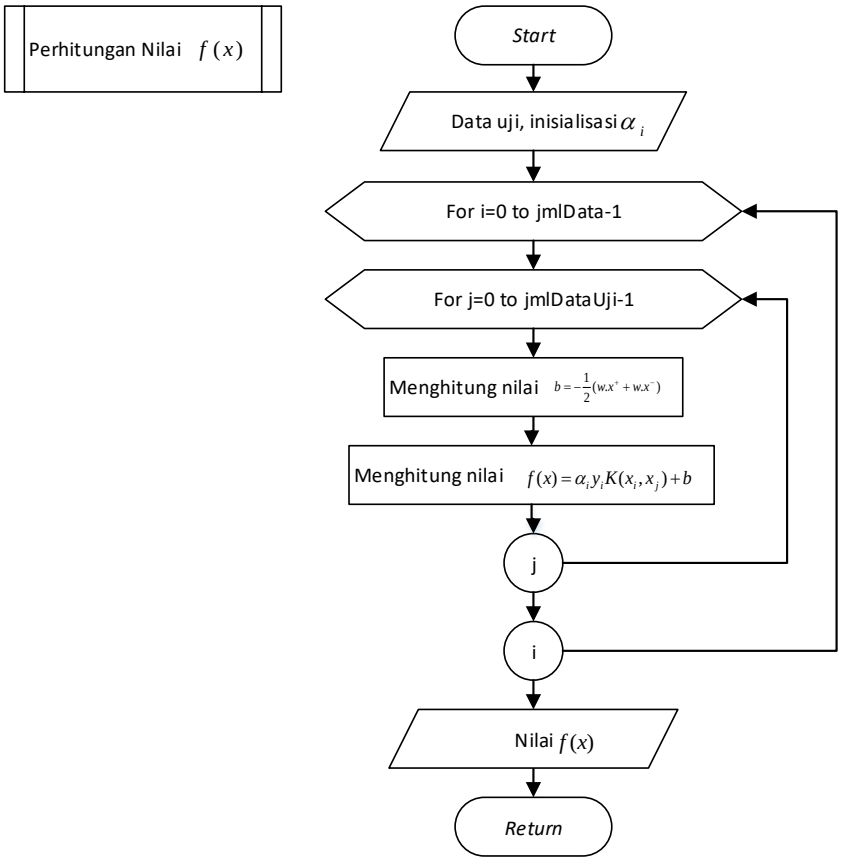
| | |
|---|---|
| 0.976526364 | 0.545454463 |
| 1.381822387 | 0.766185398 |
| 0.658880666 | 0.368227782 |
| -1.771160962 | -1.043672784 |
| -0.499663696 | -0.249831848 |
| -0.802613321 | -0.469840096 |
| -1.900555042 | -1.055297712 |
| -0.973751156 | -0.58788357 |

Maka untuk memperoleh nilai bias digunakan Persamaan 2.13.

$b = - ½ (w x^{+} + w x^{-})$

b = - ½ (0.966108737**+** 0.412914684)
b = - 0.68951171

### 4.1.7 Menghitung Nilai f(x)




**Gambar 4.8 *Flowchart* nilai bias dan fungsi f(x)**

Pada Gambar 4.8 dijelaskan tahapan-tahapan perhitungan Nilai $f(x)$. Berikut adalah penjelasannya.

1. Meng-*input*-kan *dataset testing* terpilih.
2. Inisialisasi terhadap nilai $\alpha_i$, yang diperoleh dari proses iterasi terakhir perhitungan *sequential training.*
3. Perhitungan nilai $K(x_i, x_j)$ terhadap nilai data *testing*.
4. Perhitungan $f(x) = \alpha_i . y_i . K(x_i, x_j)$. Nilai $y_i$ adalah nilai *class* dari data *testing.*
5. *Output* yang dihasilkan adalah nilai *f(x)* data *testing.*

Setelah mendapatkan nilai α, w, dan b dapat dilakukan pengujian dengan contoh data uji seperti pada Tabel 4.14

**Tabel 4.14 Data Uji**

| | Regency | Frequency | Monetery | Time | Churn Probability |
|---|---|---|---|---|---|
| **1** | 5 | 46 | 11500 | 98 | 1 |
| **2** | 2 | 10 | 2500 | 28 | 1 |
| **3** | 2 | 7 | 1750 | 14 | 1 |
| **4** | 2 | 6 | 1500 | 15 | 1 |
| **5** | 2 | 5 | 1250 | 11 | 1 |
| **6** | 2 | 14 | 3500 | 48 | 1 |
| **7** | 2 | 15 | 3750 | 49 | 1 |
| **8** | 2 | 6 | 1500 | 15 | 1 |
| **9** | 2 | 3 | 750 | 4 | 1 |
| **10** | 0 | 3 | 750 | 4 | -1 |

Langkah pertama untuk adalah menghitung *dot product* antara data latih dengan semua data uji dengan fungsi kernel linier. Sebelumnya Data uji dan data latih harus dinormalisasi terlebih dahulu. Persamaannya sebagai berikut:

$$K(x, y) = x \cdot y$$

Hasil dari Persamaan 2.19 dapat dilihat pada Tabel 4.15.

**Tabel 4.15 Hasil perhitungan kernel linier**

| | DATA 1 | DATA 2 | DATA 3 | DATA 4 | DATA 5 | DATA 6 | DATA 7 | DATA 8 | DATA 9 | DATA 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| **DATA 1** | 3.5 | 0.78090054 | 0.49242949 | 0.45655616 | 0.36749134 | 1.179713013 | 1.236862939 | 0.45655616 | 0.2 | 0 |
| **DATA 2** | 0.6466235 | 0.12888858 | 0.06356202 | 0.05717808 | 0.03721333 | 0.219612203 | 0.231428465 | 0.057178076 | 0 | 0 |
| **DATA 3** | 1.10152636 | 0.26913527 | 0.18361762 | 0.17499253 | 0.14882556 | 0.387836642 | 0.403478485 | 0.174992528 | 0.1 | 0 |
| **DATA 4** | 1.63182239 | 0.42460831 | 0.31111292 | 0.29957507 | 0.26483668 | 0.582122313 | 0.602940379 | 0.299575068 | 0.2 | 0 |
| **DATA 5** | 0.90888067 | 0.28428018 | 0.24059366 | 0.23757516 | 0.22437107 | 0.345245047 | 0.35233778 | 0.237575165 | 0.2 | 0 |
| **DATA 6** | 1.89616096 | 0.43983689 | 0.26350636 | 0.25154557 | 0.19827649 | 0.685959818 | 0.714443922 | 0.251545566 | 0.1 | 0 |
| **DATA 7** | 1 | 0.4 | 0.4 | 0.4 | 0.4 | 0.4 | 0.4 | 0.4 | 0.4 | 0 |
| **DATA 8** | 0.92761332 | 0.24082385 | 0.16743966 | 0.16285906 | 0.14073658 | 0.343347259 | 0.354944613 | 0.162859059 | 0.1 | 0 |
| **DATA 9** | 2.40055504 | 0.68115197 | 0.53795892 | 0.52485896 | 0.48120219 | 0.880224525 | 0.905547212 | 0.524858962 | 0.4 | 0 |
| **DATA 10** | 1.09875116 | 0.28049564 | 0.18506496 | 0.18083131 | 0.15226539 | 0.41422515 | 0.428191707 | 0.180831313 | 0.1 | 0 |

dilanjutkan menghitung nilai *f*(x) untuk pengambilan keputusan. Menghitung nilai *f*(x) ditentukan dengan Persamaan 2.12.

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

Data ke 1: F(x) = sign(((1 x 1 x 3.5)*...*(1 x (-1) x 1.09875116))) + (-0.68951171)
= sign (-0.22306667)
= -1

Begitu seterusnya sampai semua data uji memiliki nilai keputusan. Berikut keseluruhan nilai keputusan ditunjukkan pada Tabel 4.16.

**Tabel 4.16 Hasil klasifikasi**

| No | F(x) |
|---|---|
| **Data 1** | -1 |
| **Data 2** | -1 |
| **Data 3** | -1 |
| **Data 4** | -1 |
| **Data 5** | -1 |
| **Data 6** | -1 |
| **Data 7** | -1 |
| **Data 8** | -1 |
| **Data 9** | -1 |
| **Data 10** | -1 |

Dari hasil klasifikasi diatas dapat dicari nilai akurasi dengan cara hasil dari klasifikasi yang benar dibagi semua data uji dan mendapatkan nilai akurasi 10% dari perbandingan *actual class* dengan *prediction class*.

## 4.2 Perancangan Basis Data

Pada tahap ini tabel data yang digunakan ada tiga yaitu, data, data latih, dan data uji. Masing-masing tabel memiliki field yang sama yaitu No, A1, A2, A3, A4, dan Status. Desain *field* tiap tabel dapat dilihat pada Tabel 4.17.

**Tabel 4.17 Design field data**

| No | Field | Tipe | Keterangan |
|---|---|---|---|
| 1 | No | *Int* | Nomer urut data |
| 2 | A1 | *Int* | *Recency* |

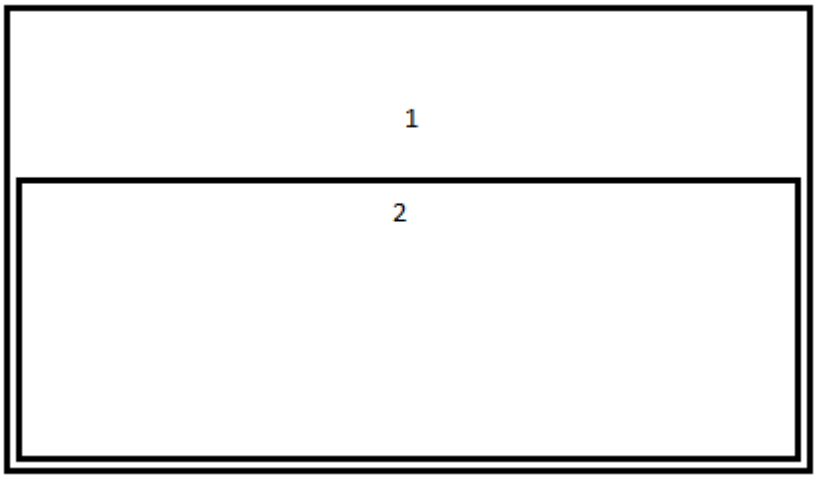
| | | | |
|---|---|---|---|
| 3 | A2 | *Int* | *Frequency* |
| 4 | A3 | *Int* | *Monetary* |
| 5 | A4 | *Int* | *Time* |
| 6 | Status | *varchar* | Bisa donor atau tidak |

## 4.3 Perancangan Antarmuka

Antarmuka sistem ini dibagi menjadi 3 halaman tab. Halaman pertama berisi halaman daftar data, data uji, dan data latih, Halaman kedua berisi parameter algoritme *Support Vector Machine* (SVM), dan Halaman ketiga adalah hasil dari klasifikasi data uji dan *form* untuk status pendonor bisa mendonorkan darahnya kembali atau tidak menurut sistem dari parameter halaman kedua.

### 4.3.1 Perancangan Halaman Data

Pada halaman ini berisi keseluruhan data meliputi data keseluruhan, data latih dan data uji yang digunakan pada sistem ini. Rancangan halaman data ditunjukkan pada Gambar 4.9.

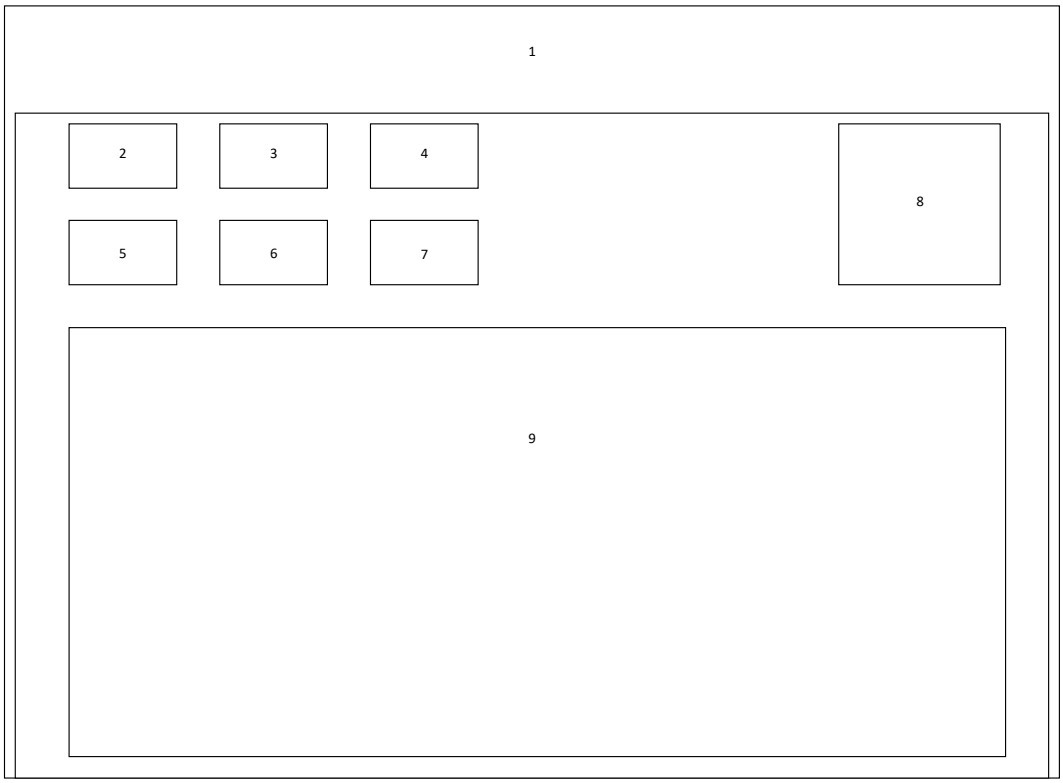



**Gambar 4.9 Halaman Data**

Keterangan:

1. Logo
2. 2. Tabel data keseluruhan pasien sekaligus kelola pembagian data latih dan uji

### 4.3.2 Perancangan Halaman Parameter Algoritme SVM



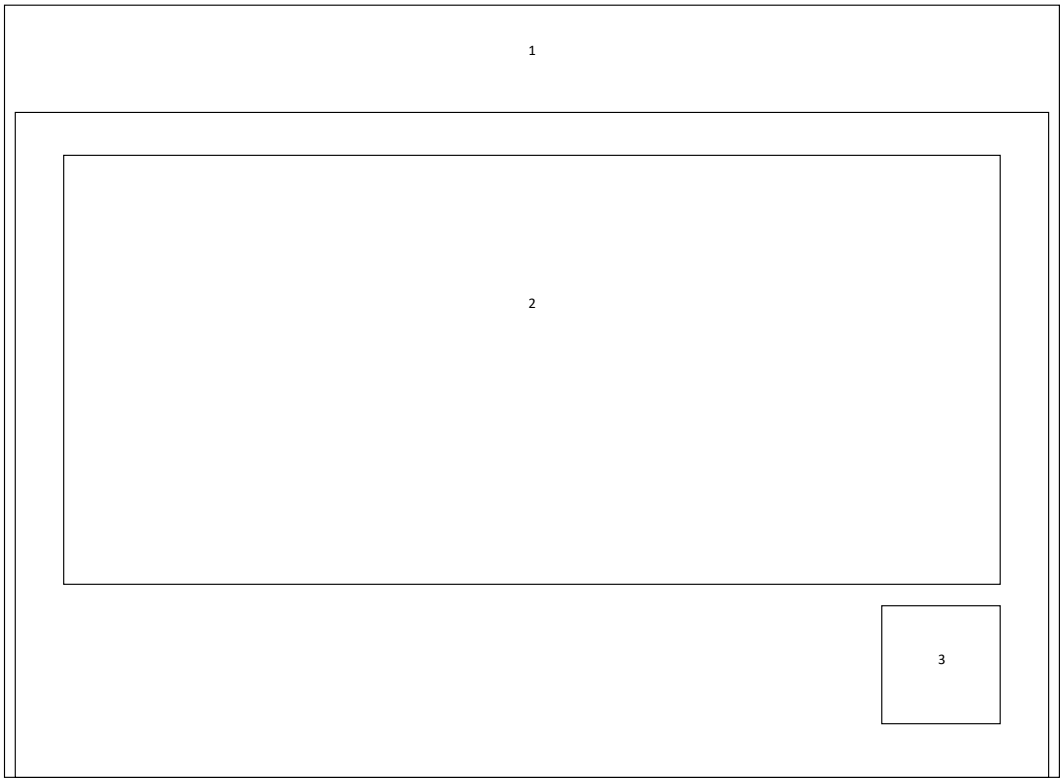

**Gambar 4.10 Halaman parameter algoritme SVM**

Rancangan halaman parameter algoritme *Support Vector Machine* (SVM) adalah halaman yang berisi form untuk memasukkan nilai-nilai parameter algoritme, jumlah data yang akan diuji, data latih, dan data uji. Rancangan halaman parameter dan hasil akurasi ditampilkan pada Gambar 4.10. Keterangan:

1. Logo
2. *Textbox* untuk nilai partikel α
3. *Textbox* untuk nilai partikel λ
4. *Textbox* untuk nilai partikel γ
5. *Textbox* untuk nilai partikel C
6. *Textbox* untuk nilai partikel ε
7. *Textbox* untuk jumlah data uji
8. *Button* untuk melakukan proses perhitungan dari masukan yang telah ada
9. Tabel Data uji dan Data *training*

### 4.3.3 Rancangan Halaman Klasifikasi dan Nilai Akurasi

Rancangan halaman hasil klasifikasi dan Nilai Akurasi adalah halaman yang berisi tabel yang menampilkan hasil klasifikasi dari data uji menurut sistem dan nilai dari masing-masing proses komputasi. Sedangkan sisi kanan bawah adalah *form* untuk nilai akurasi. Rancangan halaman hasil klasifikasi dan *form* klasifikasi ditampilkan pada Gambar 4.11.




**Gambar 4.11 Halaman klasifikasi dan akurasi**

Keterangan:

1. Logo.
2. Tabel hasil klasifikasi dan proses setiap komputasi.
3. *Textbox* hasil akurasi sistem.

## 4.4 Perancangan Pengujian Sistem

Proses pengujian sistem meliputi:

1. Pengujian terhadap data *training* dan data *testing* berdasarkan rasio perbandingan
2. Pengujian parameter *Sequential Training Support Vector Machine* (SVM)

### 4.4.2 Perancangan Pengujian Data uji dan Data latih

Perancangan pengujian rasio data dilakukan untuk mengetahui pengaruh perbandingan rasio terhadap nilai akurasi. *Dataset* RFMTC diambil dari UCI *dataset*. Pada pengujian ini digunakan 20 *dataset* RFMTC yang dibagi menjadi 10 data latih dan 10 data uji. Rasio yang digunakan adalah 90%:10, 80%:20%, 70%:30%, 60%:40%, dan 50%:50%. Berikut Tabel skenario penujian rasio data:

**Tabel 4.18 Pengujian untuk rasio data**

| No | Perbandingan Rasio Data Latih dan Data Uji | 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% | | | | | | |

### 4.4.3 Pengujian Parameter *Sequential Training* SVM

Pengujian parameter *Sequential Training* dilakukan untuk mendapatkan nilai λ (*lambda*) terbaik yaitu 0.1, 0.3, 0.5, dan 1. Perancangan ditunjukkan pada Tabel 4.18. nilai *default* yang digunakan yaitu 0.5.

**Tabel 4.19 Perancangan pengujian nilai λ (*lambda*)**

| **Percobaan ke-i** | **Nilai λ (*lambda*)** | | | | | | |
|---|---|---|---|---|---|---|---|
| | 0.1 | 0.3 | 0.5 | 1 | 1.5 | 2 | 2.5 |
| 1 | | | | | | | |
| 2 | | | | | | | |
| 3 | | | | | | | |
| 4 | | | | | | | |
| 5 | | | | | | | |
| **Rerata** | | | | | | | |

Pengujian berikutnya untuk mencari nilai γ (*gamma*) terbaik yaitu $1.10^{-3}$, $5.10^{-3}$, $1.10^{-2}$, $5.10^{-2}$, $1.10^{-1}$, $5.10^{-1}$, dan 1. Perancangan ditunjukkan pada Tabel 4.19. nilai *default* yang digunakan yaitu 0.5.

**Tabel 4.20 Perancangan pengujian nilai γ (*gamma*)**

| **Percobaan ke-i** | **Nilai γ (*gamma*)** | | | | | | |
|---|---|---|---|---|---|---|---|
| | $1.10^{-3}$ | $5.10^{-3}$ | $1.10^{-2}$ | $5.10^{-2}$ | $1.10^{-1}$ | $5.10^{-1}$ | 1 |
| 1 | | | | | | | |
| 2 | | | | | | | |
| 3 | | | | | | | |
| 4 | | | | | | | |
| 5 | | | | | | | |
| **Rerata** | | | | | | | |

Pengujian selanjutnya untuk mendapatkan nilai ε (*epsilon*) terbaik yaitu $1.10^{-6}$, $1.10^{-5}$, $1.10^{-4}$, $5.10^{-4}$, $1.10^{-3}$, $5.10^{-3}$, dan $1.10^{-2}$. Perancangan ditunjukkan pada Tabel 4.20. Nilai *default* yang digunakan $1.10^{-3}$.

**Tabel 4.21 Perancangan pengujian nilai ε (*epsilon*)**

| **Percobaan ke-i** | **Nilai ε (*epsilon*)** | | | | | | |
|---|---|---|---|---|---|---|---|
| | $1.10^{-6}$ | $1.10^{-5}$ | $1.10^{-4}$ | $5.10^{-4}$ | $1.10^{-3}$ | $5.10^{-3}$ | $1.10^{-2}$ |
| 1 | | | | | | | |
| 2 | | | | | | | |
| 3 | | | | | | | |
| 4 | | | | | | | |
| 5 | | | | | | | |
| **Rerata** | | | | | | | |

Pengujian selanjutnya untuk mendapatkan nilai *C* (*complexity*) yaitu 1, 10, 20, 30, 40, 50, dan 60. Perancangan ditunjukkan pada Tabel 4.21. Nilai *default* untuk parameter *C* (*complexity*) yaitu 1.

**Tabel 4.22 Perancangan pengujian nilai C (*complexity*)**

| **Percobaan ke-i** | **Nilai *C* (complexity)** | | | | | | |
|---|---|---|---|---|---|---|---|
| | 1 | 10 | 20 | 30 | 40 | 50 | 60 |
| 1 | | | | | | | |
| 2 | | | | | | | |
| 3 | | | | | | | |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 4 | | | | | | | |
| 5 | | | | | | | |
| **Rerata** | | | | | | | |

## 4.5 Pengambilan Keputusan

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