BAB 5 IMPLEMENTASI

5.1 Lingkungan Implementasi

Penerapan *Support Vector Machine* (SVM) dalam klasifikasi pendonor darah menggunakan *dataset* RFMTC menggunakan komputer perangkat keras yang di dalamnya terdapat perangkat lunak dengan spesifikasi seperti pada Tabel 5.1 dan Tabel 5.2.

Tabel 5.1 Lingkungan perangkat keras

Nama	**Spesifikasi**
Manufacture	Lenovo
Model	G410
Processor	Intel(R) Core(TM) i5-4200M CPU @ 2.50GHz (4 CPUs), ~2.5GHz
RAM	4096MB

Tabel 5.2 Lingkungan perangkat lunak

Nama	**Spesifikasi**
OS	Windows 10 Pro 64-Bit
Versi OS	10.0, Build 14393
Bahasa Pemrograman	PHP
Tools	Visual Studio Code
DBMS	MySQL

5.2 Batasan-batasan implementasi

Batasan implementasi sistem klasifikasi pendonor darah adalah sebagai berikut:

- Aplikasi hanya melakukan klasifikasi pendonor darah menggunakan *dataset* RFMTC.
- Pembahasan difokuskan pada implementasi metode *Support Vector Machine* (SVM) dan mengetahui kinerja (akurasi) *Support Vector Machine* (SVM) jika di terapkan pada klasifikasi pendonor darah menggunakan *Dataset* RFMTC.

5.3 Implementasi algoritme

Pada bab implementasi bahasa pemrograman akan membahas tentang klasifikasi pendonor darah menggunakan metode *SVM* pada *dataset* RFMTC yang di implementasikan menggunakan bahasa pemrograman *PHP*. Kode sumber yang

akan dijelaskan adalah proses seleksi fitur, proses perhitungan *kernel linier, matrix hessian*, *Ei,* $\delta\alpha_i$, α_i,wx^+ dan wx^-, nilai bias, pengecekan kondisi iterasi, perhitungan hasil, dan akurasi *SVM*.

5.3.1 Proses perhitungan kernel

Proses perhitungan kernel linier merupakan proses awal dalam metode *Support Vector Machine* (SVM). Berikut adalah kode sumber dari proses perhitunga *kernel linier* data latih dan uji.

No	Kernel Linier data latih dan uji
1	`$kernel_latih = array();`
2	`for($i = 0; $i < count($dataset); $i++) {`
3	`for($j = 0; $j < count($transpose_latih[0]); $j++) {`
4	`$kernel_latih[$i][$j] = 0;`
5	`$kernel_uji[$i][$j] = 0;`
6	`for($k = 0; $k < count($transpose_latih); $k++){`
7	`$kernel_latih[$i][$j]+=($dataset[$i][$k]*$transpose_latih[$k][$j]);`
8	`}`
9	`}`
10	`}`
11	`$kernel_uji = array();`
12	`for($i = 0; $i < count($dataset); $i++) {`
13	`for($j = 0; $j < count($transpose_uji[0]); $j++) {`
14	`$kernel_uji[$i][$j] = 0;`
15	`for($k = 0; $k < count($dataset[$i]); $k++){`
16	`$kernel_uji[$i][$j] += ($dataset[$i][$k] * $transpose_uji[$k][$j]);`
17	`}`
18	`}`
19	`}`

Kode Sumber 5.1 Perhitungan kernel latih dan kernel uji

Berikut adalah penjelasan dari kode sumber perhitungan kernel latih dan kernel uji:

Baris 1 : inisialisasi *array* kernel latih

Baris 2-10: Fungsi perulangan sebanyak *dataset* untuk mencari nilai kernel latih

Baris 11: inisialisasi *array* kernel uji

Baris 12-19: Fungsi perulangan sebanyak *dataset* untuk mencari nilai kernel uji

5.3.2 Proses Perhitungan Matrix Hessian

Proses perhitungan *matrix hessian* merupakan tahap kedua untuk mencari *matrix hessian.* Berikut kode sumber dari proses perhitungan *matrix Hessian*:

No	*Matrix Hessian*
1	`$matrix_latih = array();`
2	`for($i = 0; $i < count($kelas_latih); $i++){`
3	`for($j = 0; $j < count($kernel_latih[$i]); $j++){`
4	`$matrix_latih[$i][$j]=($kelas_latih[$i]*$kelas_latih[$j])* ($kernel_latih[$i][$j] + pow($t_lamda, 2));`
5	`}`
6	`}`

Kode Sumber 5.2 Proses perhitungan *matrix hessian*

Berikut adalah penjelasan dari kode sumber Matriks Hessian:

Baris 1: Inisialisasi *array* matrik latih

Baris 2-6: Fungsi Perulangan sejumlah kelas latih untuk mencari nilai matrix latih

5.3.3 Proses Perhitungan *Error*

Proses perhitungan nilai *Error* merupakan tahap ketiga dalam metode *Support Vector Machine* untuk mencari nilai *Error*. Berikut kode sumber perhitungan nilai *Error*.

No	Perhitungan *Error*
1	`for($i = 0; $i < count($matrix_latih); $i++){`
2	`$error[$i] = 0;`
3	`for ($j=0; $j < count($matrix_latih[0]); $j++) {`
4	`$error[$i] = $error[$i] + ($alpa[$i] * $matrix_latih[$i][$j]);`
5	`}`
6	`}`

Kode Sumber 5.3 Proses perhitungan nilai *Error*

Berikut adalah penjelasan dari kode sumber perhitungan nilai *Error*:

Baris 1-6: Fungsi perulangan sejumlah matrix latih untuk mencari nilai *Error.*

5.3.4 Proses Perhitungan Delta Alpha

Proses perhitungan delta alpha ($\delta\alpha_i$) merupakan tahap ke-empat dalam metode *Support Vector Machine* untuk mencari nilai δα. Berikut kode sumber perhitungan δα.

No	Perhitungan δα
1	`for($i = 0; $i < count($matrix_latih); $i++){`
2	`$delta_alpa[$i] = min(max($t_gamma * (1 - $error[$i]), $alpa[$i] * -1), $t_c - $alpa[$i]);`
3	`}`

Kode Sumber 5.4 Proses perhitungan nilai $\delta\alpha_i$

Berikut adalah penjelasan kode sumber perhitungan nilai δα:

Baris 1-3: Fungsi perulangan sejumlah matrix latih untuk mencari nilai delta alpha

5.3.5 Proses Perhitungan Alpha

Proses perhitungan alpha (α_i) merupakan tahap ke-lima dalam metode *Support Vector Machine* untuk mencari nilai α_i. Berikut kode sumber perhitungan α_i.

No	Perhitungan α_i
1	`for($i = 0; $i < count($matrix_latih); $i++){`
2	`$alpa[$i] = $alpa[$i] + $delta_alpa[$i];`
3	`}`

Kode Sumber 5.5 Proses perhitungan nilai α_i

Berikut adalah penjelasan kode sumber perhitungan nilai α:

Baris 1-3: Fungsi perulangan sejumlah matrik latih untuk mencari nilai alpha

5.3.6 Proses Pengecekan Kondisi Iterasi

Proses ini berfungsi untuk melakukan pengecekan di mana nilai δα harus lebih kecil daripada nilai *Epsilon* (ε). Berikut kode sumber pengecekan kondisi iterasi:

No	Pengecekan kondisi iterasi
1	`if($delta_alpa[$i] < $t_epsilon) {`
2	`$jumlah_delta++;`
3	`}`

Kode Sumber 5.6 Proses pengecekan iterasi

Berikut adalah penjelasan kode sumber pengecekan kondisi iterasi:

Baris 1-3: Seleksi kondisi jika delta alpha kurang dari epsilon maka jumlah delta ditambah

5.3.7 Proses Perhitungan Nilai W^+ dan W^-

Proses perhitungan nilai W^+ dan W^- digunakan untuk mencari nilai bias dalam metode *Support Vector Machine*. Berikut kode sumber perhitungan nilai W^+dan W^-:

No	Perhitungan W^+
1	`$wx_positif = array();`
2	`$wx_negatif = array();`
3	`for($i = 0; $i < count($kernel_latih); $i++){`
4	`$wx_positif[$i] = 0;`
5	`$wx_negatif[$i] = 0;`
6	`$positif_available = false;`
7	`$negatif_available = false;`
8	`for($j = 0; $j < count($alpa); $j++){`
9	`if($kelas_latih[$j] == 1){`
10	`if($alpa[$j] == $max_plus && !$positif_available) {`
11	`$wx_positif[$i] = $kernel_latih[$i][$j];`
12	`$positif_available = true;`
13	`}`
14	`}`
15	`else if($kelas_latih[$j] == -1){`
16	`if($alpa[$j] == $max_negative && !$negatif_available) {`
17	`$wx_negatif[$i] = $kernel_latih[$i][$j];`
18	`$negatif_available = true;`
19	`}`
20	`}`
21	`}`
22	`}`

Kode Sumber 5.7 Proses perhitungan nilai W^+ dan W^-

Berikut adalah penjelasan kode sumber dari perhitungan nilai W^+dan W^-:

Baris 1: Deklarasi *array wx* positif

Baris 2: Deklarasi *array wx* negatif

Baris 3: Pendefinisian kondisi fungsi perulangan kernel latih

Baris 4: Inisialisasi *wx* positif

Baris 5: Inisialisasi *wx* negatif

Baris 6: Inisialisasi positif *available*

Baris 7: Inisialisasi negatif *available*

Baris 8: Fungsi perulangan sejumlah *alpha*

Baris 9-14: Seleksi kondisi untuk mencari nilai *wx* positif

Baris 15: Seleksi kondisi jika kelas latih bernilai -1

Baris 16-20: Seleksi kondisi untuk mencari nilai *wx* negatif

5.3.8 Proses Perhitungan Nilai Bias

Proses perhitungan berikutnya adalah untuk mencari nilai bias. Berikut kode sumber untuk mencari nilai bias.

No	Perhitungan nilai bias
1	`$bias = 0;`
2	`for($i = 0; $i < count($alpa); $i++){`
3	`$bias += $alpa[$i] * $kelas_latih[$i] * ($wx_positif[$i] + $wx_negatif[$i]);`
4	`}`
5	`$bias = $bias * -1 * 0.5;`

Kode Sumber 5.8 Proses perhitungan nilai bias

Berikut adalah penjelasan kode sumber perhitungan nilai bias:

Baris 1: Inisialisasi bias

Baris 2-4: Fungsi perulangan sejumlah alpha untuk mencari bias

Baris 5: Perhitungan untuk mencari hasil bias

5.3.9 Proses Perhitungan Hasil

Proses selanjutnya adalah perhitungan data uji. Berikut adalah kode sumber dari perhitungan data uji:

No	Perhitungan Hasil
1	`$fx = array();`
2	`for($i = 0; $i < count($kernel_uji); $i++){`
3	`$fx[$i] = 0;`
4	`for($j = 0; $j < count($kernel_uji[0]) && $j < count($alpa); $j++){`
5	`$fx[$i] = $fx[$i] + ($alpa[$j] * $kelas_latih[$j] * $kernel_uji[$j][$i]);`
6	`}`
7	`$fx[$i] = $fx[$i] + $bias;`
8	`}`
9	`$hasil = array();`
10	`for($i = 0; $i < count($fx); $i++){`
11	`if($fx[$i] < 0) {`
12	`$hasil[$i] = -1;`

```
13              } else {
14                  $hasil[$i] = 1;
15              }
16          }
```

Kode Sumber 5.9 Proses perhitungan hasil data uji

Berikut adalah penjelasan kode sumber perhitungan hasil data uji:

Baris 1: Deklarasi *array fx*

Baris 2-8: Fungsi perulangan sejumlah kernel uji untuk mencari nilai *fx*

Baris 9: Deklarasi *array* hasil

Baris 10-16: Fungsi perulangan sejumlah fx untuk mencari nilai hasil

5.3.10 Proses Perhitungan Akurasi

Proses selanjutnya adalah menentukan nilai akurasi. Berikut kode sumber perhitungan nilai akurasi:

No	Perhitungan nilai akurasi
1	`$benar = 0;`
2	`for($i = 0; $i < count($hasil); $i++){`
3	`if($hasil[$i] == $kelas_uji[$i]){`
4	`$benar++;`
5	`}`
6	`}`

Kode Sumber 5.10 Proses perhitungan akurasi

Berikut adalah penjelasan kode sumber perhitungan nilai akurasi:

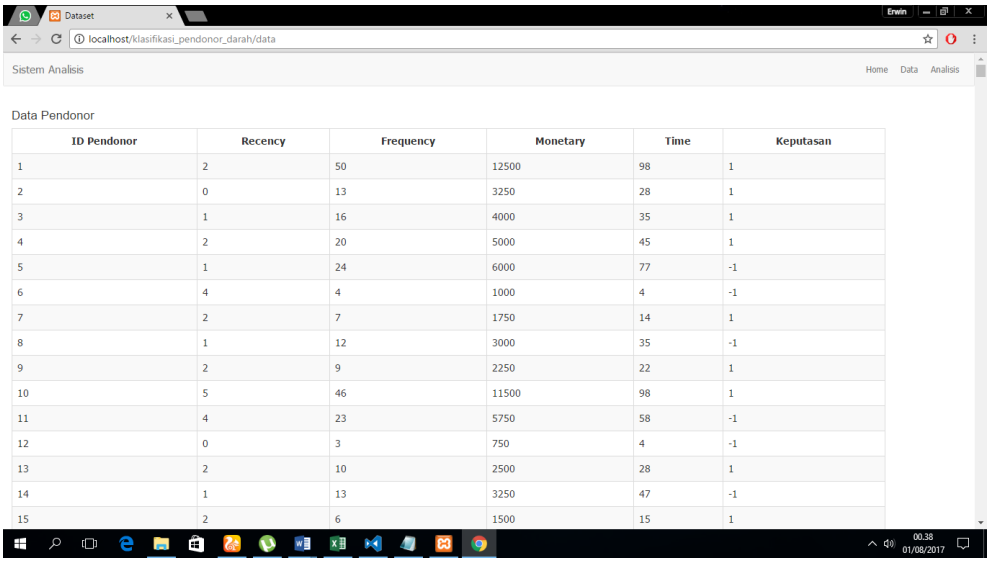
Baris 1: Inisialisasi variabel benar

Baris 2-6: Fungsi perulangan sejumlah hasil untuk mencari nilai akurasi

5.4 Implementasi Interface

Pada bab ini merupakan hasil tampilan program. Tampilan program *interface* meliputi Home, Data, dan Analisis.

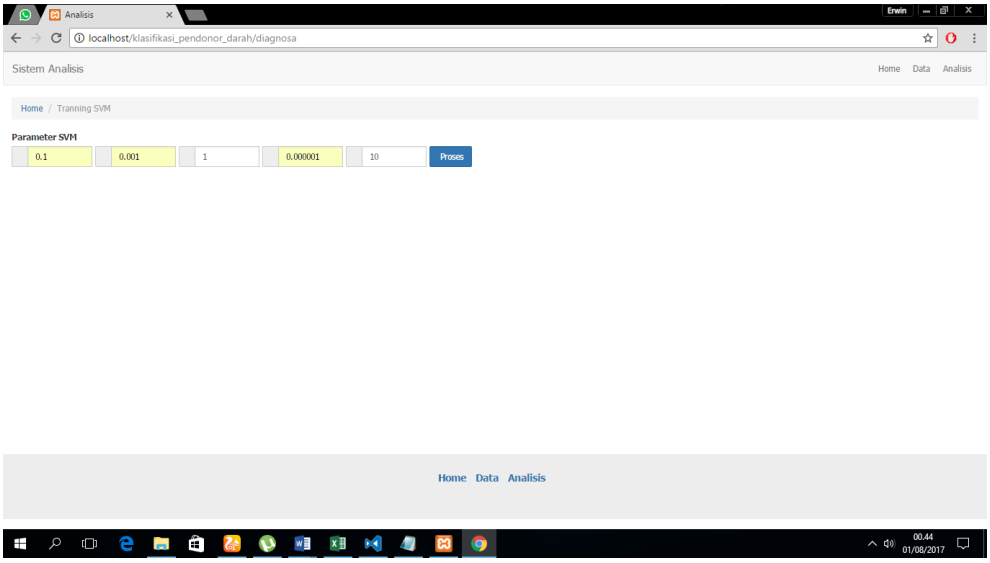
5.4.1 Tampilan Interface Dataset



Gambar 5.1 Tampilan *interface dataset*

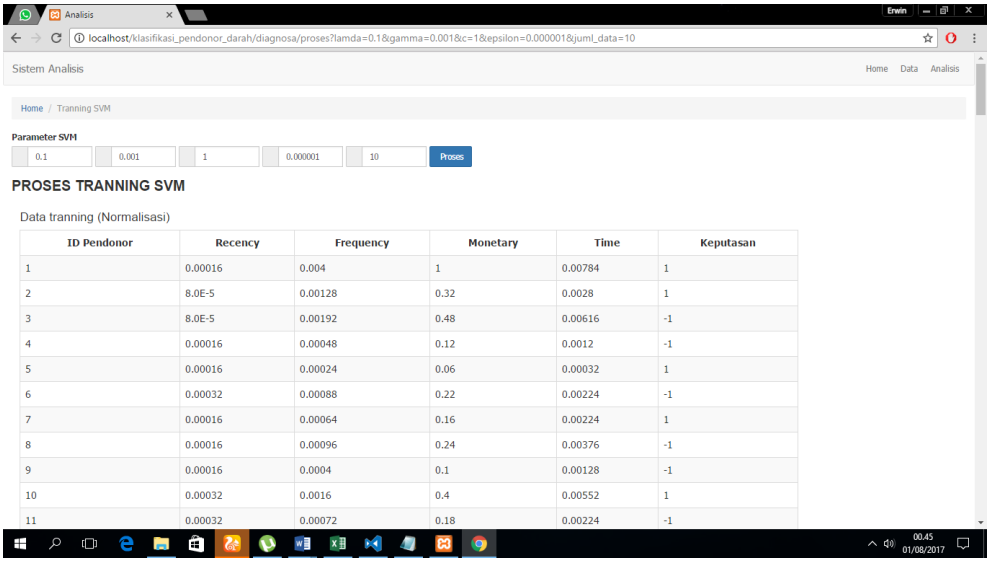
Pada Gambar 5.1 merupakan tampilan keseluruhan data yang akan digunakan pada pengujian. Untuk menampilkan keseluruhan *dataset* dapat dengan memilih menu Data.

5.4.2 Tampilan Interface Proses Training



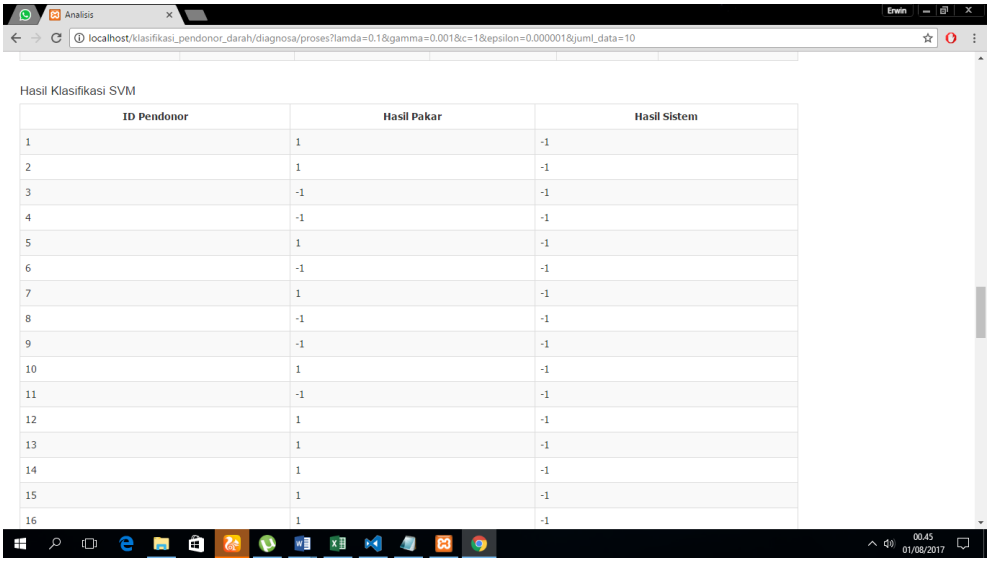
Gambar 5.2 Tampilan proses analisis

Pada Gambar 5.2 merupakan tahap awal masuk dalam proses analisis. Terdapat *form Lambda, Gamma, C (complexity),* dan *Epsilon.* Ketika semua parameter sudah terisi tekan tombol Proses maka akan menampilkan data klasifikasi.



Gambar 5.3 Tampilan setelah diproses

Pada Gambar 5.3 merupakan tampilan setelah klik tombol proses. Tampilan diatas menampilkan *dataset Recency, Frequency, Monetary, Time,* dan Keputusan.



Gambar 5.4 Tampilan hasil klasifikasi SVM

Pada Gambar 5.4 merupakan tampilan hasil dari *dataset* klasifikasi dari *dataset* pada Gambar 5.3 yang menampilkan Hasil pakar dan Hasil Sistem.

Gambar 5.5 Tampilan akurasi dan jumlah iterasi

Pada Gambar 5.5 merupakan tampilan hasil akurasi dan jumlah iterasi. Di mana jumlah iterasi merupakan kondisi dari perulangan $\delta\alpha$ harus kurang dari nilai *Epsilon*(ε).