BAB 5 IMPLEMENTASI

Dalam bab ini akan membahas mengenai implementasi dari sistem yang akan dibangun berdasarkan dari perancangan yang telah dibuat sebelumnya. Pembahasan yang ada didalamnya yaitu penjelasan dari implementasi algoritme SVM dengan menggunakan konsep OAA untuk menyelesaikan pengklasifikasian penyakit pada kucing dan implementasi dari tampilan sistem.

5.1 Spesifikasi Sistem

Dalam penelitian ini ada 2 spesifikasi sistem yang diperlukan untuk membuat sebuah program yaitu perangkat keras juga pengkat lunak. Spesifikasi perangkat keras adalah laptop yang akan dipakai untuk membuat program, kemudian spesifikasi selanjutnya sistem operasi yang akan digunakan.

5.1.1 Spesifikasi Perangkat Keras

Untuk menunjang pembuatan sistem pengklasifikasian penyakit pada kucing maka spesifikasi dari perangkat ini akan ditampilkan dalam Tabel 5.1.

Tabel 5. 1 Spesifikasi Perangkat Keras

Nama Komponen	Spesifikasi
Prosesor	Intel® Core™ i7-7500U CPU @ 2.70GHz 2.90 GHz
Memori	8,00 GB
Kartu Grafis	NVIDIA GeForce GT 940MX
Harddisk	1 TB

5.1.2 Spesifikasi Perangkat Lunak

Pembuatan sistem pengklasifikasian ini didukung oleh sistem operasi dan bahasa pemprograman yang akan ditunjukkan dalam bentuk Tabel 5.2.

Tabel 5. 2 Spesifikasi Perangkat Lunak

Nama Komponen	Spesifikasi
Sistem operasi	Microsoft Windows 10 64-bit
Bahasa pemrograman	Java
DBMS	MySQL
Tools DBMS	XAMPP Control Panel v3.2.2

5.2 Batasan Implementasi

Beberapa batasan yang digunakan dalam mengimplementasikan sistem klasifikasi gejala penyakit kucing adalah sebagai berikut :

1. Program pengklasifikasian penyakit pada kucing ini akan dibangun menggunakan bahasa pemograman *Java*.
2. Data penyakit pada kucing disimpan ke dalam Database Management Sistem (DBMS) MySQL.

3. Metode yang digunakan untuk membangun sistem klasifikasi ini adalah metode *Support Vector Machine.*
4. Input yang akan diterima oleh sistem berupa data data latih dan data uji untuk penyakit kucing
5. Output yang dihasilkan oleh sistem berupa hasil klasifikasi penyakit pada kucing berdasarkan data gelaja yang telah diproses.

5.3 Implementasi Algoritme

Implementasi algoritme *Support Vector Machine* untuk melakukan klasifikasi penyakit pada kucing ada 10 algoritme. Berikut akan dijelaskan masing-masing dari algoritme yang ada dalam sistem yang akan dibangun.

5.3.1 Implementasi Algoritme Perhitungan *Kernel* SVM

Algoritme perhitungan *kernel* SVM merupakan langkah pertama untuk melakukan proses perhitung dengan menggunakan metode SVM. Dalam perhitungan ini *kernel* yang digunakan adalah *kernel* RBF dan hasil dari perhitungan tersebut didapatkan nilai *matriks kernel* indeks n x n, dimana n adalah banyaknya data latih. Kode program untuk algoritme *kernel* ditunjukkan pada Kode Program 5.1.

```
1   System.out.println("-----------------------------------");
2   System.out.println("        Kernel RBF");
3   System.out.println("-----------------------------------");
4
5     for (int i = 0; i < kernelSVM.length; i++) {
6        for (int j = 0; j < kernelSVM[0].length; j++) {
7           double jumlahPangkat = 0;
8
9           for (int k = 0; k < dataLatih[0].length - 1; k++) {
10             jumlahPangkat   +=   Math.pow((dataLatih[i][k]   -
11                                  dataLatih[j][k]), 2);
12          }
13          kernelSVM[i][j]  =  Math.exp(-(jumlahPangkat)  /  (2  *
14                              Math.pow(DataConnection.sigma,
15                              2)));
16          System.out.print(kernelSVM[i][j] + "|");
17       }
18       System.out.println("");
19    }
```

Kode Program 5. 1 Implementasi Algoritme Perhitungan *Kernel* SVM

Berikut penjelasan Kode Program 5.1.

1. Baris 1-3 merupakan kode program untuk menampilkan tulisan “Kernel RBF”.
2. Baris 5-7 merupakan kode program untuk melakukan perulangan pada data ke-i, data ke-*j,* dan data ke-*k* sepanjang banyaknya data.
3. Baris 9-12 merupakan kode program untuk menghitung nilai pangkat yang ada pada *kernel* RBF.
4. Baris 13-15 merupakan program untuk menghitung nilai dari *kernel* RBF.

5.3.2 Implementasi Algoritme Perhitungan *Matriks Hessian*

Algoritme perhitungan matriks *hessian* SVM merupakan langkah kedua untuk melakukan proses perhitung dengan menggunakan metode SVM. Hasil dari matriks *hessian* akan dipakai untuk melakukan tahap proses perhitungan selanjutnya yaitu melakukan perhitungan *Sequential Training Support Vector Machine*. Kode program untuk algoritme matriks *hessian* ditunjukkan pada Kode Program 5.2.

```
1   System.out.println("-----------------------------------");
2   System.out.println("        Matrix Hessian");
3   System.out.println("-----------------------------------");
4
5      double maxHessian = matrixHessian[0][0];
6
7      for (int i = 0; i < matrixHessian.length; i++) {
8        for (int j = 0; j < matrixHessian[0].length; j++) {
9            matrixHessian[i][j]     =     dataLatih[i][32]     *
10           dataLatih[j][32]      *      (kernelSVM[i][j]      +
11           Math.pow(DataConnection.lambda, 2));
12
13           System.out.print(matrixHessian[i][j] + "|");
14           if (maxHessian < matrixHessian[i][j]) {
15               maxHessian = matrixHessian[i][j];
16           }
17       }
18       System.out.println("");
19    }
```

Kode Program 5. 2 Implementasi Algoritme Perhitungan Matriks *Hessian*

Berikut penjelasan Kode Program 5.2.

1. Baris 1-3 merupakan kode program untuk menampilkan tulisan “Matrix Hessian”.
2. Baris 7-8 merupakan kode program untuk melakukan perulangan pada data ke-*i*, dan data ke-*j* sepanjang banyaknya data.
3. Baris 9-11 merupakan kode program untuk menghitung nilai dari matriks *hessian*.
4. Baris 13 merupakan kode program untuk menampilkan nilai hasil dari perhitungan matriks *hessian.*
5. Baris 14-15 merupakan kode program untuk mendapatkan nilai terbesar dari matriks *hessian*.

5.3.3 Implementasi Algoritme *Sequential Training SVM*

Algoritme perhitungan *Sequential Training* SVM terdapat beberapa proses pertama menghitung nilai E_i untuk setiap data latih yang digunakan, kemudian menghitung nilai $\delta\alpha_i$ dan tahapan yang terakhir yaitu menghitung nilai α_i baru. Tahapan yang ada pada *Sequential Training* SVM ditentukan dari jumlah iterasi yang diinputkan sebelumnya.

5.3.3.1 Implementasi Algoritme Perhitungan Nilai E_i

Algoritme perhitungan nilai E_i didapatkan dari perkalian antara nilai alpha ke-i dengan nilai dari matriks *hessian*. Proses ini merupakan proses awal iterasi dari Sequential *Training* SVM. Kode program untuk algoritme nilai E_i ditunjukkan pada Kode Program 5.3.

```
1   for (int a = 0; a < jumlahIterasi; a++) {
2   System.out.println("-----------------------------------");
3   System.out.println("  Tabel Ei (iterasi " + (a + 1) + ")");
4   System.out.println("-----------------------------------");
5
6    for (int i = 0; i < tabelE.length; i++) {
7       double jumlahBaris = 0;
8       for (int j = 0; j < tabelE.length; j++) {
9           tabelE[i][j] = 1 * alfa[j] * matrixHessian[i][j];
10          jumlahBaris += tabelE[i][j];
11          System.out.print(tabelE[i][j]);
12      }
13      tabelE[i][tabelE[0].length - 1] = jumlahBaris;
14      System.out.println("|"+tabelE[i][tabelE[0].length-1]);
15   }
16   System.out.println("maxH = " + maxHessian);
```

Kode Program 5. 3 Implementasi Algoritme Perhitungan Nilai E_i

Berikut penjelasan Kode Program 5.3.

1. Baris 1 merupakan kode program untuk melakukan perulangan sebanyak jumlah iterasinya.
2. Baris 2-4 merupakan kode program untuk menampilkan tulisan "Tabel E_i"
3. Baris 6-8 merupakan kode program untuk melakukan perulangan pada data ke-*i*, dan data ke-*j* sepanjang banyaknya data.
4. Baris 9 merupakan kode program untuk menghitung nilai E_i
5. Baris 10-12 merupakan kode program untuk menjumlahkan nilai pada baris yang didapatkan pada nilai E_i dan menampilkannya.
6. Baris 13-15 merupakan kode program untuk memasukkan dan menampilkan nilai yang didapatkan pada baris sebelumnya.
7. Baris 16 merupakan kode program untuk menampilkan nilai *maximum* matriks *hessian*.

5.3.3.2 Implementasi Algoritme Perhitungan Nilai $\delta\alpha_i$

Algoritme perhitungan nilai $\delta\alpha_i$ didapatkan dari nilai minimum dari nilai maksimum yang ada pada nilai *gamma*, jumlah nilai E_i, nilai alpha ke-i dan nilai *complexity*. Kode program untuk algoritme nilai $\delta\alpha_i$ ditunjukkan pada Kode Program 5.4.

```
1   System.out.println("-------------------------------------");
2   System.out.println("                          Delta   Alfa");
3   System.out.println("-------------------------------------");
4
```

```
5     for (int i = 0; i < deltaAlfa.length; i++) {
6        deltaAlfa[i]    =    Math.min(Math.max(gamma    *    (1
7        tabelE[i][tabelE[0].length    -    1]),    -alfa[i]),
8        DataConnection.c - alfa[i]);
9
10       System.out.print(deltaAlfa[i] + " | ");
11
12       if(maxDeltaAlfa<deltaAlfa[i])      maxDeltaAlfa      =
13          deltaAlfa[i];
14    }
15    System.out.println("");
```

Kode Program 5. 4 Implementasi Algoritme Perhitungan Nilai $\delta\alpha_i$

Berikut penjelasan Kode Program 5.4.

1. Baris 1-3 merupakan kode program untuk menampilkan tulisan “Delta Alfa”.
2. Baris 5 merupakan kode program untuk melakukan perulangan pada data ke-*i*, sepanjang banyaknya data.
3. Baris 6-8 merupakan kode program untuk menghitung nilai delta alpha ($\delta\alpha_i$).
4. Baris 10-15 merupakan kode program untuk menampilkan nilai delta alpha, max delta alpha dan mendapatkan nilai.

5.3.3.3 Implementasi Algoritme Perhitungan Nilai α_i

Algoritme perhitungan nilai α_i yang baru didapatkan dari penjumlahan antara nilai $\delta\alpha i$ yang baru dengan nilai α_i sebelumnya. Kode program untuk algoritme nilai α_i ditunjukkan pada Kode Program 5.5.

```
1     System.out.println("-----------------------------------");
2     System.out.println("        Alfa Baru");
3     System.out.println("-----------------------------------");
4
5       for (int i = 0; i < alfa.length; i++) {
6           alfa[i] = alfa[i] + deltaAlfa[i];
7           System.out.print(alfa[i] + " | ");
8           if(a    ==    DataConnection.jumlahIterasi-1        ||
9              maxDeltaAlfa<DataConnection.epsilon ){
10
11                String sql4 = " ('" + alfa[i] + "')";
12                sql_insert  +=  (sql_insert.equals("")?"  insert
13                into alfa_lv1 values  ":", ")+sql4;
14          }
15    }
16
```

Kode Program 5. 5 Implementasi Algoritme Perhitungan Nilai α_i

Berikut penjelasan Kode Program 5.5.

1. Baris 1-3 merupakan kode program untuk menampilkan tulisan “Alfa Baru”.
2. Baris 5 merupakan kode program untuk melakukan perulangan pada data ke-*i*, sepanjang banyaknya data.
3. Baris 6-7 merupakan kode program untuk menghitung nilai alpha baru(α_i) dan menampilkan nilai yang didapatkan.

4. Baris 11-13 merupakan kode program untuk menyimpan nilai alpha baru kedalam database dengan nama tabelnya adalah alfa_lv1.

5.3.4 Implementasi Algoritme Perhitungan Nilai *b*

Algoritme perhitungan nilai b didapatkan dari penjumlahan total nilai wx^+ dan total nilai wx^-. Proses pencarian nilai wx^+ dan nilai wx^- menggunakan perhitungan *kernel* masing-masing dari kelas positif dan negatif dengan nilai alpha baru pada masing-masing kelas. Kode program untuk nilai b ditunjukkan pada Kode Program 5.6.

```
1  System.out.println("----------------------------------------");
2  System.out.println("        Tabel K+ dan K-");
3  System.out.println("----------------------------------------");
4
5    KplusKmin = new double[tabelLatih.getRowCount()][2];
6
7    int maxPlus = -1;
8    int maxMin = -1;
9
10   double maxAlfaPlus = -1;
11   double maxAlfaMinus = -1;
12
13   for (int i = 0; i < tabelLatih.getRowCount(); i++) {
14      if (dataLatih[i][32] > 0) {
15         if (maxAlfaPlus < alfa[i]) {
16             maxAlfaPlus = alfa[i];
17             maxPlus = i;
18         }
19
21      }else if (dataLatih[i][32] < 0) {
22         if (maxAlfaMinus < alfa[i]) {
23             maxAlfaMinus = alfa[i];
24             maxMin = i;
25         }
26       }
27   }
28
29   System.out.println("Max Plus = " + maxPlus);
30   System.out.println("nilai : " + alfa[maxPlus]);
31   System.out.println("Max Min  = " + maxMin);
32   System.out.println("nilai : " + alfa[maxMin]);
33
34   for (int i = 0; i < KplusKmin.length; i++) {
35      KplusKmin[i][0] = kernelSVM[i][maxPlus];
36      KplusKmin[i][1] = kernelSVM[i][maxMin];
37      System.out.print(KplusKmin[i][0] + " | "+KplusKmin[i][1]);
38      System.out.println("");
39   }
40
41 System.out.println("----------------------------------------");
42 System.out.println("        Tabel W");
43 System.out.println("----------------------------------------");
44
45   w = new double[tabelLatih.getRowCount() + 1][2];
46
47   double jumlahPlus = 0;
```

```
48    double jumlahMin = 0;
49
50      for (int i = 0; i < w.length - 1; i++) {
51         for (int j = 0; j < w[0].length; j++) {
52             w[i][j] = KplusKmin[i][j]* dataLatih[i][32]*alfa[i];
53             System.out.printf("%.9f | ", w[i][j]);
54
55             if (j == 0) {
56                jumlahPlus += w[i][j];
57             }else if (j == 1) {
58                jumlahMin += w[i][j];
59              }
60         }
61         System.out.println("");
62      }
63
64      w[w.length - 1][0] = jumlahPlus;
65      w[w.length - 1][1] = jumlahMin;
66      System.out.println(w[w.length - 1][0]+"|"+w[w.length-1][1]);
67
68      B = -0.5 * (w[w.length - 1][0] + w[w.length - 1][1]);
69      nilaiB.setText("Nilai B = " + B);
70
71      try{
72         Statement s = c.createStatement();
73         System.out.println("nilai B : "+B);
74
75         String sql = "update nilai_b set nilai = '"+B+"' where level
76         = 1";
77
78         s.executeUpdate(sql);
79
80      }catch(SQLException ex){
81         Logger.getLogger(MainPrograms.class.getName())
82         .log(Level.SEVERE, null, ex);
83      }
```

Kode Program 5. 6 Implementasi Algoritme Perhitungan Nilai *b*

Berikut penjelasan Kode Program 5.6.

1. Baris 1-3 merupakan kode program untuk menampilkan tulisan “Tabel K+ dan Tabel K-”.
2. Baris 5 merupakan kode program untuk memasukkan nilai kpluskmin.
3. Baris 14-27 merupakan kode program untuk melakukan perulangan pada data ke-*i*, sepanjang banyaknya data.
4. Baris 29-32 merupakan kode program untuk menampilkan nilai K+ dan K- yang dipakai.
5. Baris 34 merupakan kode program untuk melakukan perulangan pada data ke-*i*, sepanjang banyaknya data.
6. Baris 35 merupakan kode program untuk mengambil nilai *kernel* SVM berdasarkan nilai maximum yang bernilai positif.

7. Baris 36 merupakan kode program untuk mengambil nilai *kernel* SVM berdasarkan nilai maximum yang bernilai negatif.
8. Baris 37-39 merupakan kode program untuk menampilkan nilai K+ dan nilai K-.
9. Baris 41-43 merupakan kode program untuk menampilkan tulisan “Tabel W”.
10. Baris 45 merupakan kode program untuk memasukkan nilai w.
11. Baris 50-51 merupakan kode program untuk melakukan perulangan pada data ke-*i*, dan data ke-*j* sepanjang banyaknya data.
12. Baris 52-53 merupakan kode program untuk menghitung nilai w dan menampilkan nilai yang didapatkan.
13. Baris 55-66 merupakan kode program untuk menghitung jumlah total nilai w positif dan jumlah total nilai w negatif dan menampilkan nilai yang didapatkan.
14. Baris 68-69 merupakan kode program untuk menghitung nilai b dan menampilkan nilai yang didapatkan.
15. Baris 75-78 merupakan kode program untuk mengupdate nilai b kedalam database dengan nama tabelnya adalah nilai_b.

5.3.5 Implementasi Algoritme Perhitungan Nilai *f(x)*

Algoritme perhitungan nilai *f(x)* berada pada tahap pengujian hasil dari sistem yang nilainya diperoleh dari penjumlahan antara nilai bias dengan jumlah perhitungan pada data uji. Ada 2 hasil dari nilai *f(x)* yang akan didapatkan yaitu bernilai positif atau bernilai negatif. Nilai positif atau nilai negatif menentukan data uji tersebut tergolong kedalam kelas positif atau negatif. Apabila nilai *f(x)* yang didapatkan bernilai positif maka data uji tersebut akan masuk kedalam kelas 1 dan jika nilai *f(x)* bernilai negatif maka data uji tersebut akan masuk kedalam kelas -1. Kode program untuk algoritme perhitungan nilai *f(x)* ditunjukkan pada Kode Program 5.7.

```
1     if (pilihan_kernel == 2) {
2     System.out.println("-------------------------------------");
3     System.out.println("        Kernel RBF");
4     System.out.println("-------------------------------------");
5
6     jumlahKanan = 0;
7     kernelSVM = new double[dataLatih.length][2];
8     for(int j = 0; j < kernelSVM.length; j++) {
9        double jumlahPangkat = 0;
10       for (int k = 0; k < dataLatih[0].length - 1; k++) {
11         jumlahPangkat+=Math.pow((dataUji[d][k]-
12    dataLatih[j][k],2);
13       }
14       kernelSVM[j][0] = Math.exp(-1*jumlahPangkat/(2 * Math.pow
15                         (DataConnection.sigma, 2)));
16       System.out.print(kernelSVM[j][0] + "|");
17
18
```

```
19      kernelSVM[j][1]                                                 =
20  alfa[j]*dataLatih[j][32]*kernelSVM[j][0];
21
22      jumlahKanan += kernelSVM[j][1];
23      System.out.printf("%.12f |", kernelSVM[j][1]);
24
25      System.out.println("");
26    }
27  }
28
29  System.out.printf("Jumlah Kanan : %.12f\n", jumlahKanan);
30  System.out.println("-------------------------------------");
31  System.out.println("        Nilai F(x)");
32  System.out.println("-------------------------------------");
33
34  try {
35
36      Connection c = DataConnection.getKoneksi();
37
38      Statement s = c.createStatement();
39
40      String sql = "select nilai from nilai_b where level = " +
41                   (a+1);
42
43      ResultSet r = s.executeQuery(sql);
44
45      while (r.next()) {
46            b = Double.parseDouble(r.getString("nilai"));
47      }
48
49  }catch (SQLException ex) {
50
51          Logger.getLogger(MainPrograms.class.getName())
52          .log(Level.SEVERE, null, ex);
53    }
54
55  System.out.println(b);
56
57  fx = jumlahKanan + b;
58
59  System.out.printf("Nilai F{x} : %.12f\n", fx);
```

Kode Program 5. 7 Algoritme Perhitungan Nilai *f(x)*

Berikut penjelasan Kode Program 5.7.

1. Baris 1 merupakan kode program jika pilihan_*kernel* bernilai 2 maka akan masuk kedalam perhitungan *kernel* RBF untuk menghitung *kernel* pada data ujinya.
2. Baris 2-4 merupakan kode program untuk menampilkan tulisan "*Kernel* RBF".
3. Baris 7-17 merupakan kode program untuk menghitung *kernel* pada data uji dan menampilkan nilai *kernel* yang didapatkan.
4. Baris 19-25 merupakan kode program untuk menghitung nilai *kernel* dari data uji yang dikalikan dengan nilai alpha yang didapatkan sebelumnya kemudian menjumlahkan semua nilai yang dapatkan dari hasil perkalian tersebut.

5. Baris 30-32 merupakan kode program untuk menampilkan tulisan “Nilai F(x)”.
6. Baris 40-55 merupakan kode program untuk mengambil nilai b yang sudah disimpan didalam database sebelumnya dan menampilkan nilai b tersebut.
7. Baris 57-59 merupakan kode program untuk menghitung nilai *f(x)* dan menampilkan nilai *f(x)* yang didapatkan.

5.3.6 Implementasi Algoritme *One-Against-All*

Algoritme perhitungan *One-Against-All* proses pengklasifikasian dari sistem yang dibangun. Kode program perhitungan *One-Against-All* ditunjukkan pada Kode Program 5.8.

```
1   System.out.println("-----------------------------------");
2   System.out.println("        Sign Nilai F(x)");
3   System.out.println("-----------------------------------");
4   if (fx > 0) {
5      System.out.println("Positif");
6      sign[a] = "1";
7      nilaiFx = String.valueOf(fx);
8      break;
9   }else {
10     System.out.println("Negatif");
11     sign[a] = "-1";
12  }
13    if (a == 7 && fx < 0) {
14    kondisi = 8;
15    nilaiFx = String.valueOf(fx);
16  }
17
18  switch (kondisi) {
19
20    case 0:
21      status = "Scabies";
22      KelasSistem = "Scabies";
23      break;
24    case 1:
25      status = "Gastritis";
26      KelasSistem = "Gastritis";
27      break;
28    case 2:
29      status = "Helminthiasis";
30      KelasSistem = "Helminthiasis";
31      break;
32    case 3:
33      status = "Rhinitis";
34      KelasSistem = "Rhinitis";
35      break;
36    case 4:
37      status = "Dermatophytosis";
38      KelasSistem = "Dermatophytosis";
39      break;
40    case 5:
41      status = "Dermatitis";
42      KelasSistem = "Dermatitis";
43      break;
44    case 6:
      status = "Otitis";
```

```
45      KelasSistem = "Otitis";
46      break;
47    case 7:
48      status = "Enteritis";
49      KelasSistem = "Enteritis";
50      break;
51    default:
52      status = "Sehat";
53      KelasSistem = "Sehat";
54      break;
55  }
56
```

Kode Program 5. 8 Algoritme *One-Against-All*

Berikut penjelasan Kode Program 5.7.

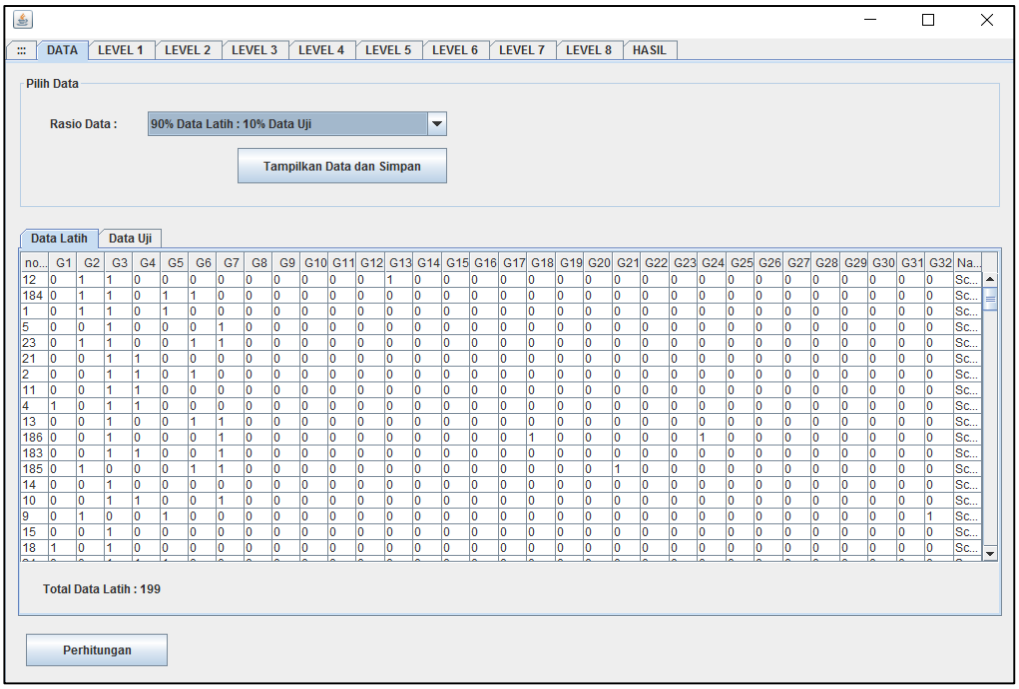
1. Baris 1-3 merupakan kode program untuk menampilkan tulisan “Sign Nilai F(x)”.
2. Baris 4-8 merupakan kode program untuk kondisi dimana nilai fx bernilai lebih besar dari 0 maka akan ditampilkan tulisan positif dan menampilkan nilai fx dan nilai signnya bernilai 1 dan proses dihentikan.
3. Baris 9-12 merupakan kode program untuk kondisi dimana nilai fx bernilai negatif dan nilai signnya bernilai -1.
4. Baris 13-15 merupakan kode program untuk kondisi dimana ketika nilai a bernilai 7 dan nilai fx yang didapatkan bernilai kurang dari 0 maka akan masuk pada kondisi 8 dan akan ditampilakan nilai fx.
5. Baris 17-22 merupakan kode program untuk kondisi dimana *case* 0 memiliki perintah untuk status *scabies* dan kelas sistemnya *scabies*.
6. Baris 23-26 merupakan kode program untuk kondisi dimana *case* 1 memiliki perintah untuk status *gastritis* dan kelas sistemnya *gastritis*.
7. Baris 27-30 merupakan kode program untuk kondisi dimana *case* 2 memiliki perintah untuk status *helminthiasis* dan kelas sistemnya *helminthiasis*.
8. Baris 31-34 merupakan kode program untuk kondisi dimana *case* 3 memiliki perintah untuk status *rhinitis* dan kelas sistemnya *rhinitis*.
9. Baris 35-39 merupakan kode program untuk kondisi dimana *case* 4 memiliki perintah untuk status *dermatitis* dan kelas sistemnya *dermatitis*.
10. Baris 40-43 merupakan kode program untuk kondisi dimana *case* 5 memiliki perintah untuk status *dermaphytosis* dan kelas sistemnya *dermaphytosis*.
11. Baris 44-47 merupakan kode program untuk kondisi dimana *case* 6 memiliki perintah untuk status *otitis* dan kelas sistemnya *otitis*.
12. Baris 48-51 merupakan kode program untuk kondisi dimana *case* 7 memiliki perintah untuk status *enteritis* dan kelas sistemnya *enteritis*.
13. Baris 52-56 merupakan kode program untuk kondisi dimana *default* memiliki perintah untuk status sehat dan kelas sistemnya sehat.

5.4 Implementasi Antarmuka Sistem

Implementasi antarmuka pada sistem yang akan dibangun merupakan penghubung secara langsung antara pengguna dengan aplikasi yang akan dibangun. Ada 10 bagian utama dari tampilan halaman antarmuka sistem untuk pengklasifikasian penyakit kucing dengan menggunakan algoritme *Support Vector Machine* yaitu halaman pilih data, halaman *Support Vector Machine* level 1, halaman *Support Vector Machine* level 2, halaman *Support Vector Machine* level 3, halaman *Support Vector Machine* level 4, halaman *Support Vector Machine* level 5, halaman *Support Vector Machine* level 6, halaman *Support Vector Machine* level 7, halaman *Support Vector Machine* level 8, dan halaman hasil klasifikasi sistem.

5.4.1 Implementasi Antarmuka Halaman Pilih Data

Antarmuka untuk halaman pilih data merupakan halaman yang berfungsi untuk menampilkan data yang akan digunakan untuk proses perhitungan dengan menggunakan algoritme *Support Vector Machine*. Data yang digunakan pada sistem ini untuk data latih ataupun data uji memiliki format .sql. Tampilan untuk antarmuka halaman pilih data dapat dilihat pada Gambar 5.1.



Gambar 5. 1 Implementasi Antarmuka Halaman Pilih Data

5.4.2 Implementasi Antarmuka Halaman SVM Level 1

Antarmuka untuk halaman *Support Vector Machine* level 1 merupakan halaman yang berfungsi untuk menampilkan hasil perhitungan dengan menggunakan algoritme *Support Vector Machine* pada level 1. Proses yang ada pada perhitungan *Support Vector Machine* pada level 1 yaitu perhitungan *kernel* RBF, *matriks hessian*, *sequential training Support Vector Machine*. Nilai *sequential training Support Vector Machine* adalah nilai nilai E_i, nilai $\delta\alpha_i$ dan nilai α_i.

Tampilan untuk antarmuka halaman *Support Vector Machine* level 1 dapat dilihat pada Gambar 5.2.

5.4.3 Implementasi Antarmuka Halaman SVM Level 2

Antarmuka untuk halaman *Support Vector Machine* level 2 merupakan halaman yang berfungsi untuk menampilkan hasil perhitungan dengan menggunakan algoritme *Support Vector Machine* pada level 2. Proses yang ada pada perhitungan *Support Vector Machine* pada level 2 yaitu perhitungan *kernel* RBF, *matriks hessian*, *sequential training Support Vector Machine*. Nilai *sequential training Support Vector Machine* adalah nilai nilai E_i, nilai $\delta\alpha_i$ dan nilai α_i. Tampilan untuk antarmuka halaman *Support Vector Machine* level 2 dapat dilihat pada Gambar 5.3.

5.4.4 Implementasi Antarmuka Halaman SVM Level 2

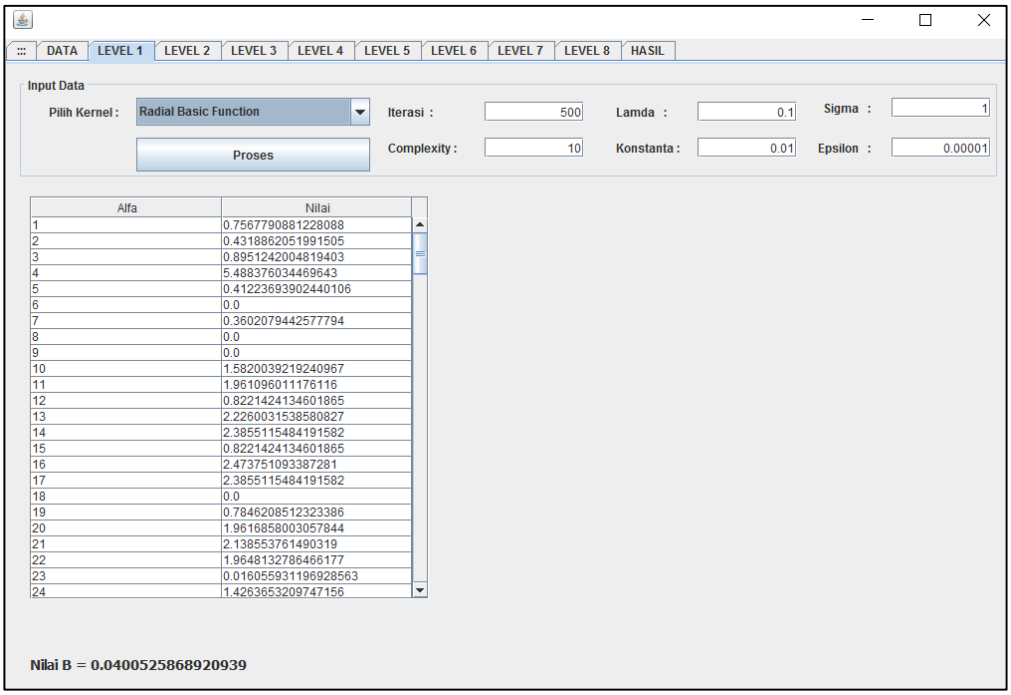
Antarmuka untuk halaman *Support Vector Machine* level 2 merupakan halaman yang berfungsi untuk menampilkan hasil perhitungan dengan menggunakan algoritme *Support Vector Machine* pada level 2. Proses yang ada pada perhitungan *Support Vector Machine* pada level 2 yaitu perhitungan *kernel* RBF, *matriks hessian*, *sequential training Support Vector Machine*. Nilai *sequential training Support Vector Machine* adalah nilai nilai E_i, nilai $\delta\alpha_i$ dan nilai α_i. Tampilan untuk antarmuka halaman *Support Vector Machine* level 2 dapat dilihat pada Gambar 5.3.

5.4.5 Implementasi Antarmuka Halaman SVM Level 3

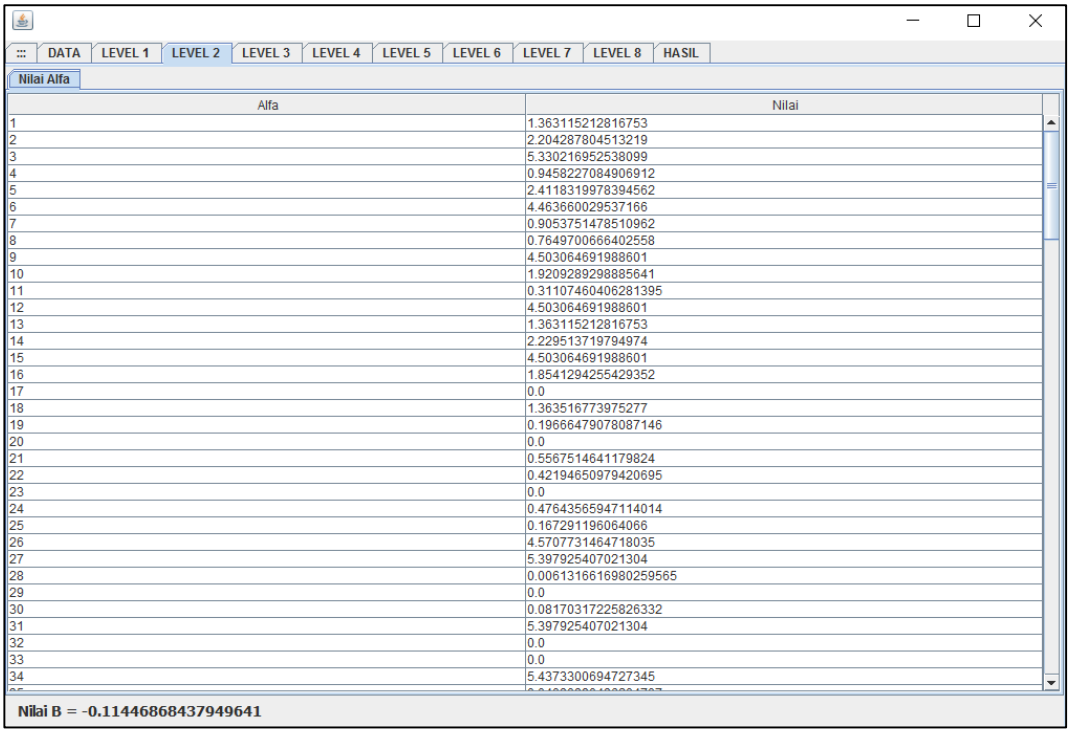
Antarmuka untuk halaman *Support Vector Machine* level 3 merupakan halaman yang berfungsi untuk menampilkan hasil perhitungan dengan menggunakan algoritme *Support Vector Machine* pada level 3. Proses yang ada pada perhitungan *Support Vector Machine* pada level 3 yaitu perhitungan *kernel* RBF, *matriks hessian*, *sequential training Support Vector Machine*. Nilai *sequential training Support Vector Machine* adalah nilai nilai E_i, nilai $\delta\alpha_i$ dan nilai α_i. Tampilan untuk antarmuka halaman *Support Vector Machine* level 3 dapat dilihat pada Gambar 5.4. Tampilan untuk halaman *Support Vector Machine* level 4, halaman *Support Vector Machine* level 5, halaman *Support Vector Machine* level 6, halaman *Support Vector Machine* level 7, dan halaman *Support Vector Machine* level 8 memiliki tampilan yang sama seperti dengan tampilan halaman *Support Vector Machine* level 2 dan 3. Perbedaan yang ada pada halaman *Support Vector Machine* disetiap level terletak dari hasil perhitungan disetiap levelnya, karena perhitungan setiap levelnya menggunakan jumlah data yang berbeda.

5.4.6 Implementasi Antarmuka Halaman Hasil Klasifikasi Sistem

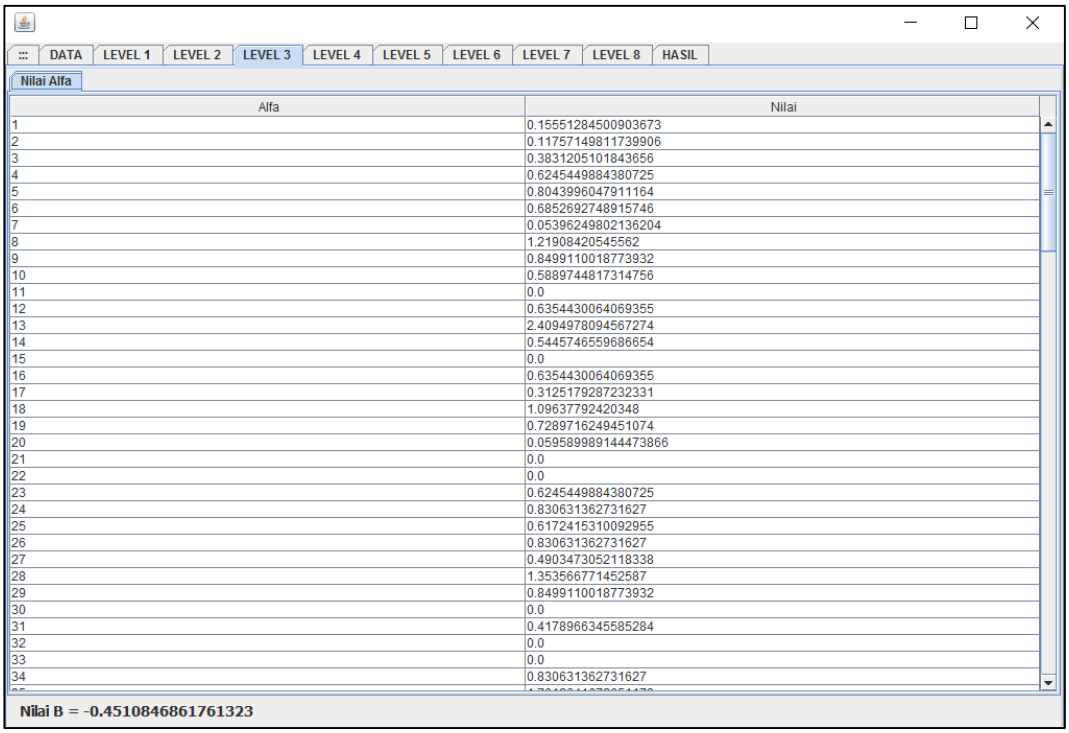
Antarmuka untuk halaman hasil klasifikasi sistem merupakan halaman yang berfungsi untuk menampilkan hasil pengklasifikasi sistem dengan menggunakan algoritme *Support Vector Machine* dan menampilkan data yang sesuai dengan kelas penyakitnya juga hasil dari akurasi sistem yang dibangun. Tampilan untuk antarmuka halaman hasil klasifikasi sistem dapat dilihat pada Gambar 5.5.



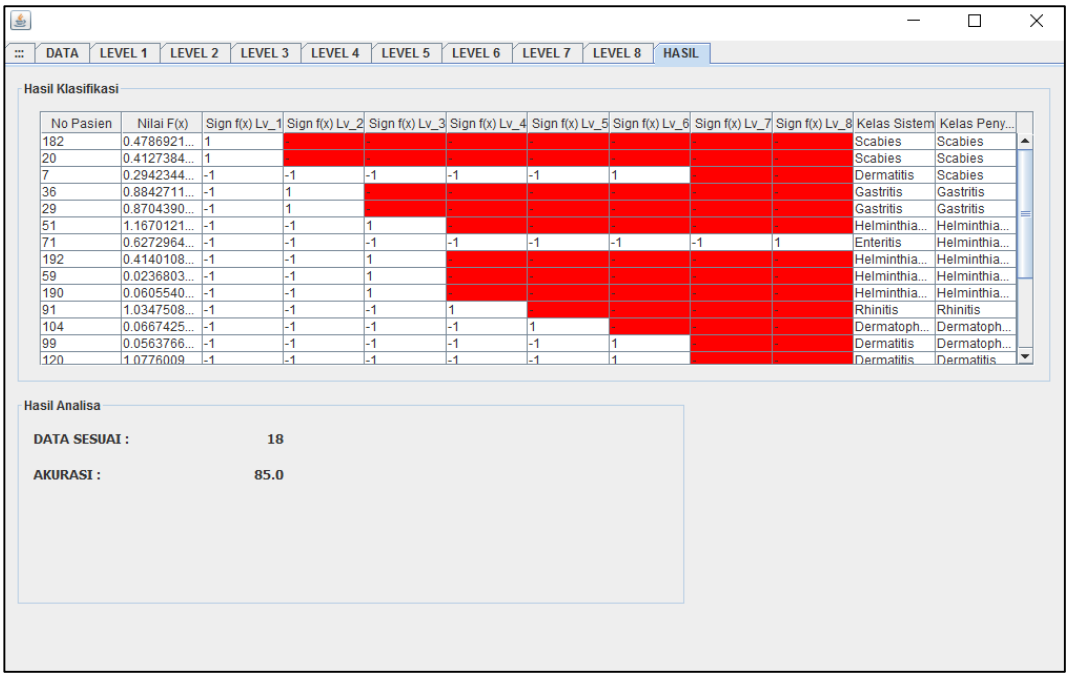
Gambar 5. 2 Implementasi Antarmuka Halaman SVM Level 1



Gambar 5. 3 Implementasi Antarmuka Halaman SVM Level 2



Gambar 5. 4 Implementasi Antarmuka SVM Level 3



Gambar 5. 5 Implementasi Antarmuka Halaman Hasil Klasifikasi Sistem