BAB 5 IMPLEMENTASI

Bab ini menjelaskan implementasi sistem berdasarkan analisis kebutuhan dan proses perancangan sistem sebelumnya. Pembahasan pada bab ini terdiri dari penjelasan implementasi algoritma *Extreme Learning Machine* untuk penyelesaian klasifikasi penyakit IMS berdasarkan dataset dari pakar.

5.1 Spesifikasi Sistem

Tahap implementasi sistem membutuhkan spesifikasi perangkat yang sesuai agar sistem yang dirancang bangun sesuai kebutuhan spesifikasi supaya program dapat berfungsi lancar. Spesifikasi perangkat yang dibutuhkan oleh sistem terdiri dari spesifikasi perangkat keras dan spesifikasi perangkat lunak.

5.1.1 Spesifikasi Perangkat Keras

Dalam implemantasi ini dibutuhkan perangkat keras berupa PC yang akan dipakai untuk proses implementasi program dengan metode *extreme learning machine* dan kebutuhan perangkatnya adalah sebagai berikut :

1. Dengan processor Intel core i3-2230M 2,2 GHz
2. Ukuran memory RAM 4 GB
3. Kapasitas Hard Disk sebesar 640 GB

5.1.2 Spesifikasi Perangkat Lunak

Spesifikasi perangkat lunak yang dugunakan selama proses implementasi metode rough set dan certainty factor untuk mendeteksi dini penyakit menular seksual sebagai berikut :

1. *Operating System* yang digunakan adalah *OS Windows 10 Pro 64-bit*.
2. Aplikasi penunjang pembuatan dokumen dan perhitungan adalah *Microsoft Office 2016*.
3. *Microsoft Visio 2007* adalah penunjang penulis dalam membuat *flowchart*.
4. *Microsoft Visual Studio 2013* merupkan tool untuk mengimplemantasikan sistem ke dalam aplikasi dekstop.

5.1.3 Spesifikasi Perangkat Lunak

Terdapat beberapa batasan yang membatasi pengembangan sistem ini, diantaranya adalah :

1. *Input* yang diterima sistem adalah berupa dataset dari pakar hasil survey gejala IMS.
2. *Output* yang diberikan berupa hasil akurasi testing dan klasifikasi IMS.
3. Bahasa Pemrograman C# yang digunakan untuk pengembangan sistem.
4. *Extreme Learning Machin*e merupakan single metode yang dipakai untuk implementasi algoritma.

5.2 Implementasi Algoritma

Berdasarkan pembahasan perancangan *software* pada Bab IV ,maka implementasi program dalam pembuatan aplikasi implemantasi metode ELM dengan menggunakan bahasa pemrogaman C# terdapat pada sub bab ini.

5.2.1 Implementasi *Generate Random*

Implementasi generate random digunakan untuk mencari nilai random yang akan dipakai pada proses *weight* dan bias. Generate random merupakan proses yang bertujuan untuk membentuk matriks dengan variable *input* batas atas bawah dan jumlah ordo matriknya. Nilai yang ada dalam matriks bobot *input* diperoleh dari nilai acak dengan *range* yang telah ditentukan. Implementasi *generate random* ditunjukkan pada Tabel 5.1 berikut.

Kode Program 5.1 *Generate Random*

No	Kode Program
1	`double[,] GenerateRandom(uint row, uint column, double min, double`
2	`max)`
3	`        {`
4	`            double[,] randData;`
5	`            if (row > 0 && column > 0)`
6	`            {`
7	`                randData = new double[row, column];`
8	`            }`
9	`            else`
10	`            {`
11	`                randData = new double[1, 1];`
12	`            }`
13	`            for (int rowIndex = 0; rowIndex < randData.GetLength(0);`
14	`rowIndex++)`
15	`            {`
16	`                for (int columnIndex = 0; columnIndex <`
17	`randData.GetLength(1); columnIndex++)`
18	`                {`
19	`                    randData[rowIndex, columnIndex] = min +`
20	`rand.NextDouble() * (max - min);`
21	`                }`
22	`            }`
23	`            return randData;`
24	`        }`

Penjelasan kode program untuk implementasi inisialisasi *random* adalah sebagai berikut:

Baris 1 : Fungsi GenerateRandom menerima parameter masukan yaitu uint row, uint column, double min, double max.

Baris 4 s.d 11 : Proses random dari batasan maksimal dan minimal.

Baris 13 s.d 20 : Proses perulangan untuk mengisi matriks bobot_input sesuai baris dan kolomnya dengan nilai yang telah diacak dari batas maksimal dan minimal

Baris 14 : Mengembalikan nilai matriks random.

5.2.2 Implementasi Transpose Matriks

Implementasi transpose matriks digunakan dalam proses perhitungan ELM. Transpose matrik mengubah bentuk matrik ke dalam bentuk baru yaitu membalik posisi indek matrik sehingga indek kolom akan menjadi baris begitu juga sebaliknnya. Implementasi transpose matriks ditunjukkan dengan kode program pada Tabel 5.2.

Kode Program 5.2 Transpose Matriks

No	Kode Program
1	`public double[,] matriks_transpose(double[,] matriks)`
2	`{`
3	`  double[,]matriks_transpose=new double[matriks.GetLength(1),`
4	`matriks.GetLength(0)];`
5	
6	`  for (int i = 0; i < matriks_transpose.GetLength(0); i++)`
7	`  {`
8	`   for (int j = 0; j < matriks_transpose.GetLength(1); j++)`
9	`   {`
10	`     matriks_transpose[i, j] = matriks[j, i];`
11	`   }`
12	`  }`
13	`  return matriks_transpose;`
14	`}`

Penjelasan kode program untuk implementasi transpose matriks adalah sebagai berikut:

Baris 1 : Fungsi matriks_transpose menerima parameter matriks masukan sebagai matriks.

Baris 3 : Proses pembentukan matriks baru dengan nama matriks_transpose dengan ordo kolom matriks masukan x baris matriks masukan.

Baris 6 s.d 12 : Proses perulangan untuk melakukan transpose matriks dengan cara mengubah baris dan kolom matriks masukan menjadi kolom dan baris baru yang disimpan pada matriks baru matriks_transpose.

Baris 13 : Mengembalikan nilai matriks_transpose.

5.2.3 Implementasi Inverse Matrik OBE

Implementasi inverse matrik termasuk dalam proses ELM dan merupakan proses rumit jika memiliki jumlah ordo banyak maka dari itu akan memakai metode OBE sesuai dengan bab perancangan. Implementasi perhitugan inverse matrik OBE ditunjukkan dengan kode program pada Tabel 5.3 berikut.

Kode Program 5.3 Inverse Matrik OBE

No	Kode Program
1	`public static double[,] inverse_matriks(double[,] matriksA)`
2	`        {`
3	

```
            double[,] matriks = new double[matriksA.GetLength(0),
matriksA.GetLength(1)];
            double[,] matriks_invers = new
double[matriks.GetLength(0), matriks.GetLength(1)];

            for (int i = 0; i < matriks.GetLength(0); i++)
            {
                for (int j = 0; j < matriks.GetLength(1); j++)
                {
                    matriks[i, j] = matriksA[i, j];
                }
            }

            for (int i = 0; i < matriks.GetLength(0); i++)
            {
                for (int j = 0; j < matriks.GetLength(1); j++)
                {
                    if (i == j)
                    {
                        matriks_invers[i, j] = 1;
                    }
                    else
                    {
                        matriks_invers[i, j] = 0;
                    }
                }
            }

            for (int i = 0; i < matriks.GetLength(0); i++)
            {
                double t = matriks[i, i];

                for (int k = 0; k < matriks.GetLength(1); k++)
                {
                    matriks_invers[i, k] = matriks_invers[i, k] / t;
                    matriks[i, k] = matriks[i, k] / t;

                }

                for (int j = 0; j < matriks.GetLength(1); j++)
                {
                    double c = matriks[j, i];
                    for (int l = 0; l < matriks.GetLength(1); l++)
                    {
                        if (i != j)
                        {
                            matriks_invers[j, l] = matriks_invers[j,
l] - c * matriks_invers[i, l];
                            matriks[j, l] = matriks[j, l] - c *
matriks[i, l];

                        }

                    }
                }
            }


            return matriks_invers;
```

```
63          }
```

Penjelasan kode program untuk implementasi penjumlahan matriks dengan bias adalah sebagai berikut:

Baris 1 s.d 7 : Fungsi inisialisasi menerima parameter untuk oprasi perhitungan selanjutnya

Baris 9 s.d 30 : Proses perulangan untuk membentuk matrik identitas OBE.

Baris 30 s.d 60 : Proses perulangan untuk memindahkan matrik identitas ke sebelah kiri dan matrik awal di sebelah kanan.

Baris 63 : Mengembalikan nilai matriks inverse.

5.2.4 Implementasi *Training*

Implementasi *training* merupakan implementasi yang menunjukkan proses-proses yang ada dalam tahap *training* sebagaimana telah dijabarkan pada perancangan bab 4. Dalam proses *training* ini memiliki output yaitu bobot keluaran training. Selain itu nilai random weight dan bias juga masih dipakai dalam perhitungan selanjutnya yaitu testing. Implementasi perhitungan H_{init} ditunjukkan dengan kode program pada Tabel 5.4 berikut.

Kode Program 5.4 Perhitungan *Training*

```
1   var weight_training = GenerateRandom(hidden_neuron, input_layer,
2   weightMin, weightMax);
3                   var bias_training = GenerateRandom(1, hidden_neuron,
4   0, 1);
5   var weight_transpose =
6   MatrixMath.matriks_transpose(weight_training);
7                   var H_init =
8   MatrixMath.matriks_perkalian(x_training, weight_transpose);
9                   H_init = MatrixMath.ones_bias(H_init,
10  bias_training);
11                  var H = Calculate_H(H_init);
12                  var H_transpose = MatrixMath.matriks_transpose(H);
13                  var _HT_H =
14  MatrixMath.matriks_perkalian(H_transpose, H);
15                  //Console.Write(">");
16                  var HT_H = MatrixMath.inverse_matriks(_HT_H);
17                  //Console.Write(">");
18                  var H_plus = MatrixMath.matriks_perkalian(HT_H,
19  H_transpose);
20                  var beta_training =
21  MatrixMath.matriks_perkalian(H_plus, y_training);
22                  //Console.Write(" Testing " + trainCount + " Data,
23  ");
24
25                  //======================TESTING Y TRAINING PHASE
26                  //klasifikasi training
27                  var y_topi = MatrixMath.matriks_perkalian(H,
28  beta_training);
29                  var accuracy = new double[y_topi.GetLength(0), 17];
```

```
30                  for (int i = 0; i < accuracy.GetLength(0); i++)
31                  {
32                      for (int j = 0; j < accuracy.GetLength(1); j++)
33                      {
34                          accuracy[i, j] = Math.Abs(y_topi[i, 0] - j +
35  1);
36                      }
37                  }
38                  //mencari indeks nilai terkecil utk klasifikasi
39                  var predict = new double[accuracy.GetLength(0)];
40                  for (int i = 0; i < predict.Length; i++)
41                  {
42                      predict[i] = double.MaxValue;
43                      int idxMin = 0;
44                      for (int j = 0; j < accuracy.GetLength(1); j++)
45                      {
46                          if (predict[i] > accuracy[i, j])
47                          {
48                              idxMin = j;
49                              predict[i] = accuracy[i, j];
50                          }
51                      }
52                      predict[i] = (double)idxMin;
53                  }
```

Penjelasan kode program untuk implementasi perhitungan H_{init} adalah sebagai berikut:

Baris 1 s.d 4 : Mencari nilai random *weight* dan bias yang akan digunakan dalam proses perhitungan ELM

Baris 5 s.d 23 : Melakukan proses perhitungan ELM dari step pertama sampai dengan proses perhitungan bobot beta *training*.

Baris 27 s.d 53 : mencari Y_topi *training* kemudian mengklasifikasikan kedalam kelas-kelas untuk mengetahui bobot yang baik.

5.2.5 Implementasi *Testing*

Imlementasi *testing* merupakan implementasi yang menunjukkan proses-proses yang ada dalam tahap *testing* sebagaimana telah dijabarkan pada perancangan bab 4. Data masukan dari testing berupa matrik X_testing, beta_topi, *weight* dan bias *random*. Keluaran akhir menghasilkan klasifikasi data dan menghitung akurasi kesesuaian dengan data aktual. Implementasi perhitungan H_{init} ditunjukkan dengan kode program pada Tabel 5.5 berikut.

Kode Program 5.5 Perhitungan *Testing*

```
No  Kode Program
1               var weight_transpose_test =
2   MatrixMath.matriks_transpose(weight);
3               var H_init_test =
4   MatrixMath.matriks_perkalian(x_testing, weight_transpose_test);
5               H_init_test = MatrixMath.ones_bias(H_init_test, bias);
6               var H_test = Calculate_H(H_init_test);
7               var y_topi_test = MatrixMath.matriks_perkalian(H_test,
8   beta);
```

```
9               var accuracy_test = new double[y_topi_test.GetLength(0),
10  17];
11              //klasifikasi testing
12              for (int i = 0; i < accuracy_test.GetLength(0); i++)
13              {
14                  for (int j = 0; j < accuracy_test.GetLength(1); j++)
15                  {
16                      accuracy_test[i, j] = Math.Abs(y_topi_test[i, 0]
17  - j + 1);
18                  }
19              }
20              for (int i = 0; i < y_testing.GetLength(0); i++)
21              {
22                  y_testing[i, 1] = double.MaxValue;
23                  int idxMin = 0;
24                  for (int j = 0; j < accuracy_test.GetLength(1); j++)
25                  {
26                      if (y_testing[i, 1] > accuracy_test[i, j])
27                      {
28                          idxMin = j;
29                          y_testing[i, 1] = accuracy_test[i, j];
30                      }
31                  }
32                  y_testing[i, 1] = (double)idxMin;
33              }
```

Penjelasan kode program untuk implementasi perhitungan H_{init} adalah sebagai berikut:

Baris 1 s.d 9 : Perhitungan Testing dari data training kemudian menghasilkan beta_topi

Baris 12 s.d 33 : Perulangan untuk mengklasifikasikan data ke dalam kelas-kelas penyakit sesuai dengan bobot beta topi training

5.2.6 Implementasi Perhitungan Akurasi

Implementasi perhitungan Akurasi bertujuan untuk melakukan perhitungan nilai evaluasi dengan cara membandingkan hasil prediksi dengan data *testing*. Pada proses ini menghasilkan presentase nilai akurasi kebenaran. Implementasi perhitungan akurasi ditunjukkan dengan kode program pada Tabel 5.6.

Kode Program 5.6 Perhitungan Akurasi

No	Kode Program
1	`int count_test = 0;`
2	`            for (int i = 0; i < y_testing.GetLength(0); i++)`
3	`            {`
4	`                if (Math.Abs(y_testing[i, 1] - y_testing[i, 0]) <`
5	`double.Epsilon)`
6	`                {`
7	`                    count_test++;`
8	`                }`
9	`            }`
10	`            if (count_test > accuracyTesting)`

```
11              {
12                  accuracyTesting = count_test;
13              }
```

Penjelasan kode program untuk implementasi perhitungan $\hat{Y}$ adalah sebagai berikut:

Baris 1 : Inisialisasi variable untuk perhitungan akurasi *testing*

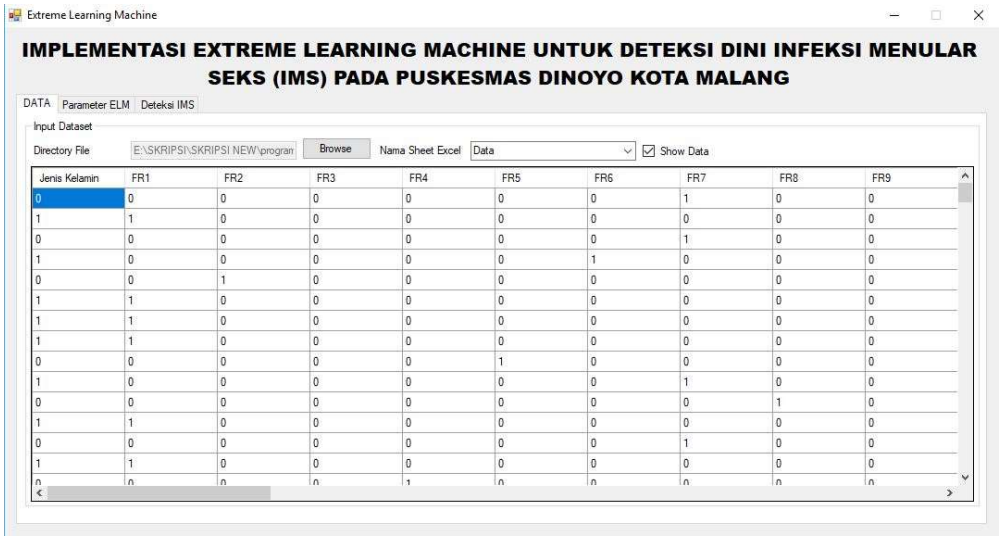
Baris 2 s.d 13 : Perulangan untuk menemukan akurasi terbaik dalam proses klasifikasi *testing*

5.3 Implementasi Antarmuka

Implementasi antarmuka untuk implementasi algoritma *Extreme Learning Machine* untuk deteksi dini IMS digunakan oleh pengguna sistem agar lebih mudah dalam berinteraksi secara langsung dengan sistem. Antarmuka sistem pada penelitian ini terdiri dari halaman data, parameter ELM, dan deteksi IMS.

5.3.1 Antarmuka Halaman Data

Halaman Data merupakan halaman awal dari sistem, halaman ini untuk memasukkan dataset *file Microsoft Excel* (.xls/.xlsx). Pada halaman ini menampilkan tabel dataset yang telah dinormalisasi kemudian dipakai untuk proses ELM. Hasil implementasi antarmuka halaman data ditunjukkan pada gambar 5.1.

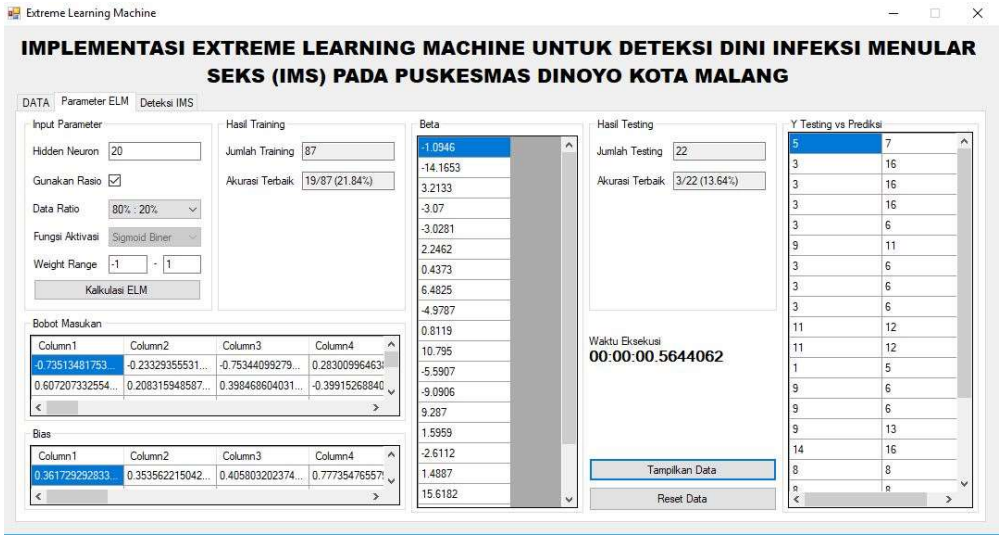


Gambar 5.1 Implementasi Halaman Data

5.3.2 Antarmuka Parameter ELM

Halaman Parameter ELM merupakan halaman kedua dari sistem, berisi halaman untuk mengisi parameter dan perhitungan ELM yaitu *training* dan *testing.* Parameter input pada halaman ini berupa jumlah *hidden neuron*, data rasio *training testing*, *range* nilai bobot *W* random. Hasil ELM yang ditampilkan

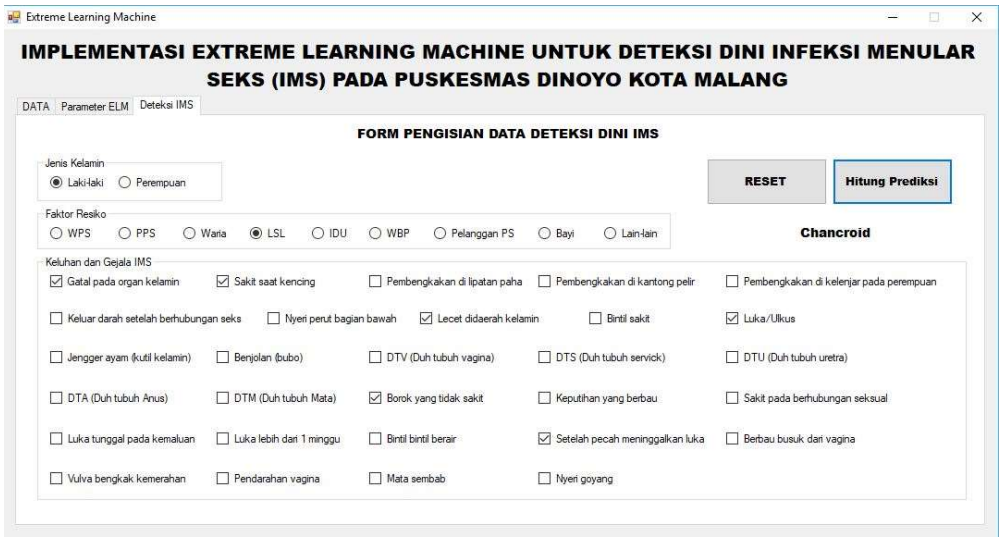
ialah bobot masukan, nilai bias, akurasi training dan testing, $\hat{\beta}_{training}$, $Y_{testing}$, $Y_{prediksi}$ dan waktu eksekusi. Pada halaman ini terdapat opsi untuk menampilkan nilai-nilai perhitungan ELM ataupun pilihan *hidden*. Hasil implementasi antarmuka halaman parameter ELM ditunjukkan pada gambar 5.2.



Gambar 5.2 Implementasi Halaman Parameter ELM

5.3.3 Antarmuka Halaman Deteksi IMS

Antarmuka Halaman Deteksi IMS merupakan halaman terakhir dari sistem, berisi halaman untuk prediksi IMS dengan mengisi form berisi atribut yang diperlukan untuk prediksi. Atribut yang diperlukan ialah jenis kelamin faktor risiko dan keluhan gejala IMS. Pada halaman ini menampilkan prediksi kelas penyakit menggunakan algoritma ELM. Hasil implementasi antarmuka halaman deteksi IMS ditunjukkan pada gambar 5.3.



Gambar 5.3 Implementasi Halaman Prediksi