BAB 5 IMPLEMENTASI

Pada bab ini akan membahas tentang implementasi sistem yang telah dibuat. Untuk menjelaskan secara rinci tentang tahapan implementasi sistem di bab ini akan dijelaskan 3 sub bab utama yaitu lingkungan implementasi, implementasi program dan implementasi antarmuka.

5.1 Lingkungan Implementasi

Spesifikasi kebutuhan perangkat dalam pembuatan perangkat lunak yang digunakan untuk penelitian ini agar dapat berjalan dengan baik saat proses implementasi dibutuhkan beberapa hal, antara lain:

1. Spesifikasi Perangkat Keras (*Hardware*):
   a. *Processor* Intel® Corei3-3120M 2.50GHz
   b. RAM 8 GB
   c. *Harddisk* dengan kapasitas 550 GB
2. Spesifikasi Perangkat Lunak (*Software*):
   a. Sistem operasi Mi*Cr*osoft Windows 8.1 64 bit
   b. Bahasa pe*Mr*ograman java dengan *editor* NetBeans IDE 7.3.1
   c. *MiCrosoft Office Word* 2016 untuk proses pembuatan dokumen
   d. *MiCrosoft Office Excel* 2016 untuk melakukan proses perhitungan
   e. Format *file* .txt untuk menampung data

5.2 Impementasi Program

Pada sub-bab ini akan dijelaskan mengenai proses implementasi pembangunan optimasi FIS *Tsukamoto* untuk diagnosis autisme pada anak menggunakan algoritme genetika. Implementasi ini dibangun berdasarkan perancangan pada bab sebelumnya dengan menggunakan bahasa pemrograman Java.

5.2.1 Implementasi Proses Representasi Kromosom

Proses ini merupakan proses representasi kromosom dengan nama *method* inisialisasi_populasi. Berikut *sourcecode* dari representasi kromosom yang dapat dilihat pada *Sourcecode* 5.1.

```
for (int i = 0; i < popsize; i++) {
            for (int j = 0; j < panjangK; j++) {
                if (j < 2) { // j == 0 || j == 1 khusus
untuk kolom umur
                    kromosom[i][j] = random.nextInt((int)
(max_umur - min_umur + 1)) + min_umur;
                } else if (j >= 2 && j < 5) { // j == 2 || j
== 3 || j == 4 khusus untuk kolom kolesterol
                    kromosom[i][j] = random.nextInt((int)
(max_kolesterol - min_kolesterol + 1)) + min_kolesterol;
                } else if (j == 5) { //khusus untuk kolom
HDL
                    kromosom[i][j] = random.nextInt((int)
(max_HDL - min_HDL + 1)) + min_HDL;
                } else if (j >= 6 && j < 9) { // j == 6 || j
== 7 || j == 8khusus untuk LDL
                    kromosom[i][j] = random.nextInt((int)
(max_LDL - min_LDL + 1)) + min_LDL;
                } else if (j >= 9 && j < 12) { // j == 9 ||
j == 10 || j == 11 khusus untuk trigliserida
                    kromosom[i][j] = random.nextInt((int)
(max_trigliserida - min_trigliserida + 1)) +
min_trigliserida;
                } else if (j >= 12 && j < 16) //{j == 12 ||
j == 13 || j == 14 || j == 15
                {
                    kromosom[i][j] = random.nextInt((int)
(max_stroke - min_stroke + 1)) + min_stroke;
                }
            }

            //System.out.print(kromosom[i][j] + " ");
            double temp = 0;
            double a = kromosom[i][0];
            double b = kromosom[i][1];
            if (a > b) {
                temp = kromosom[i][0];
                kromosom[i][0] = kromosom[i][1];
                kromosom[i][1] = temp;
            }
            temp = 0;
            for (int k = 0; k < 3; k++) {
                for (int l = 2; l < (5 - 1); l++) {
                    if (kromosom[i][l] > kromosom[i][l + 1])
{
                        temp = kromosom[i][l];
                        kromosom[i][l] = kromosom[i][l + 1];
                        kromosom[i][l + 1] = temp;
                    }
                }
            }
            temp = 0;
            for (int k = 0; k < 3; k++) {
                for (int l = 6; l < (9 - 1); l++) {
                    if (kromosom[i][l] > kromosom[i][l + 1])
{
                        temp = kromosom[i][l];
                        kromosom[i][l] = kromosom[i][l + 1];
                        kromosom[i][l + 1] = temp;
```

```
60                          }
61
62                      }
63                  }
64
65                  temp = 0;
66                  for (int k = 0; k < 3; k++) {
67                      for (int l = 9; l < (12 - 1); l++) {
68                          if (kromosom[i][l] > kromosom[i][l + 1])
69  {
70                              temp = kromosom[i][l];
71                              kromosom[i][l] = kromosom[i][l + 1];
72                              kromosom[i][l + 1] = temp;
73                          }
74
75                      }
76                  }
77                  temp = 0;
78                  for (int k = 0; k < 4; k++) {
79                      for (int l = 12; l < (16 - 1); l++) {
80                          if (kromosom[i][l] > kromosom[i][l + 1])
81  {
82                              temp = kromosom[i][l];
83                              kromosom[i][l] = kromosom[i][l + 1];
84                              kromosom[i][l + 1] = temp;
85                          }
86
87                      }
88                  }
89
90              }
```

***Sourcecode* 5.1 Representasi Kromosom**

5.2.2 Implementasi Proses *Crossover*

Proses ini merupakan proses reproduksi *Crossover* dengan metode *one cut point*. Berikut proses tersebut yang ditampilkan pada *Sourcecode* 5.2.

```
1   public void CrossOver() {
2           int jumlahCross = (int) Math.ceil(Cr * popsize);
3          // System.out.println("jumlah Cross : " +
4   jumlahCross);
5           int sisa = jumlahCross % 2;
6           int z = 0;
7           if (sisa == 1) {
8               jumlahCross = jumlahCross + 1;
9           }
10          double[][] tampungArray = new
11  double[jumlahCross][panjangK];
12          childC = new double[jumlahCross][panjangK];
13  //menyimpan anak child
14          count = jumlahCross; // menyimpan jumlah
15  Crossovernya
16          int jumlahCrossB = jumlahCross / 2;
17         // System.out.println("jumlah Cross b : " +
18  jumlahCrossB);
19          for (int i = 0; i < jumlahCrossB; i++) {
20              int rand = random.nextInt(popsize);//untuk
```

```
pemilihan individu yang di Crossover
            int rand2 = random.nextInt(popsize);//untuk
pemilihan individu yang di Crossover
            int rand3 = random.nextInt(panjangK - 1);//untuk
pemotongan individu
    int b = 0;
            while (b < 1) {
                if (rand == rand2) {
                    rand = random.nextInt(popsize);
                    rand2 = random.nextInt(popsize);
                } else {

                    int q = 0;
                    for (int j = 0; j <= rand3; j++)
{ //untuk masukan kepala parent
                        tampungArray[q][j] =
kromosom[rand][j];
                        tampungArray[q + 1][j] =
kromosom[rand2][j];

                    }

                    int r = 0;
                    for (int j = rand3 + 1; j < panjangK;
j++) { //untuk menyilang ke ekor
                        tampungArray[r][j] =
kromosom[rand2][j];
                        tampungArray[r + 1][j] =
kromosom[rand][j];
                    }
                    b++;
                }
            }
        int x = 0;
            for (int k = z; k < z + 2; k++) {
                for (int j = 0; j < panjangK; j++) {
                    childC[k][j] = tampungArray[x][j];
                }
                x++;
                //System.out.println("");
            }
            x = 0;
            z = z + 2;
        }
}
```

Sourcecode 5.2 One Cut Point

5.2.3 Implementasi Proses Mutasi

Bagian ini menampilkan proses dari mutasi yaitu salah satu proses reproduksi yang ada di dalam Algoritme Genetika. Mutasi yang dilakukan pada proses ini adalah *random mutation* dengan kode program yang ditunjukkan pada *Sourcecode* 5.3.

```
public double[] mutasi(int data1) {
        double[] p1 = new double[panjangK];
        double[] child = new double[panjangK];
        for (int i = 0; i < panjangK; i++) {
```

```
5               p1[i] = kromosom[data1][i];
6               child[i] = kromosom[data1][i];
7           }
8           int random1;
9           random1 = random.nextInt(panjangK); // menentukan
10  index mana yang terpilih
11          double tampa = Math.random() * (0.1 - (-0.1));
12          double alpha = -0.1 + tampa;
13          double tamp1 = p1[random1];
14          double hasil1;
15          if (random1 < 2) {//Umur
16              hasil1 = tamp1 + (alpha * (max_umur -
17  min_umur));
18              child[random1] = hasil1;
19          }
20          if (random1 > 1 && random1 < 5) {//Kolesterol
21              hasil1 = tamp1 + (alpha * (max_kolesterol -
22  min_kolesterol));
23              child[random1] = hasil1;
24          }
25          if (random1 == 5) {//HDL
26              hasil1 = tamp1 + (alpha * (max_HDL - min_HDL));
27              child[random1] = hasil1;
28          }
29          if (random1 > 5 && random1 < 9) {//LDL
30              hasil1 = tamp1 + (alpha * (max_LDL - min_LDL));
31              child[random1] = hasil1;
32          }
33          if (random1 > 8 && random1 < 12) {//Triglisida
34              hasil1 = tamp1 + (alpha * (max_trigliserida -
35  min_trigliserida));
36              child[random1] = hasil1;
37          }
38          if (random1 > 11 && random1 < 16) {//Tingkat Resiko
39              hasil1 = tamp1 + (alpha * (max_stroke -
40  min_stroke));
41              child[random1] = hasil1;
42          }
43          return child;
44      }
```

Sourcecode 5.3 Random Mutation

5.2.4 Implementasi Proses Menghitung *Fitness*

Bagian ini menampilkan proses dari perhitungan *fitness* yaitu dengan *input array data* yang didapatkan dari proses fuzzifikasi yang terdapat dalam proses *Fuzzy Tsukamoto*. Berikut kode program yang ditunjukkan pada *Sourcecode* 5.4 Perhitungan *Fitness*.

```
1   public double hitungfitness(double a) {
2           double fitt;
3           fitt = (a / 105) * 100;
4           return fitt;
5       }
```

Sourcecode 5.4 Perhitungan *Fitness*

5.2.5 Implementasi Proses Seleksi *Elitism*

Dalam proses implementasi seleksi pada Algoritme Genetika ini menggunakan metode *elitism*. Proses implementasi pada *Sourcecode* 5.5.

```
1   public void seleksi() {
2           double swap;
3           double[][] swap2 = new double[1][panjangK];;
4           for (int i = 0; i < popsize + count + anakMutasi;
5   i++) {
6               for (int j = 0; j < popsize + count + anakMutasi
7   - 1; j++) {
8                   if (fitnessGabungan[j] < fitnessGabungan[j +
9   1]) {
10                      swap = fitnessGabungan[j];
11                      fitnessGabungan[j] = fitnessGabungan[j +
12  1];
13                      fitnessGabungan[j + 1] = swap;
14                      swap2[0] = kromosomGabungan[j];
15                      kromosomGabungan[j] = kromosomGabungan[j
16  + 1];
17                      kromosomGabungan[j + 1] = swap2[0];
18                  }
19              }
20          }
21  for (int i = 0; i < popsize; i++) {
22              kromosom[i] = kromosomGabungan[i];
23              double temp = 0;
24              double a = kromosom[i][0];
25              double b = kromosom[i][1];
26              if (a > b) {
27                  temp = kromosom[i][0];
28                  kromosom[i][0] = kromosom[i][1];
29                  kromosom[i][1] = temp;
30              }
31              temp = 0;
32              for (int k = 0; k < 3; k++) {
33                  for (int l = 2; l < (5 - 1); l++) {
34                      if (kromosom[i][l] > kromosom[i][l + 1])
35  {
36                          temp = kromosom[i][l];
37                          kromosom[i][l] = kromosom[i][l + 1];
38                          kromosom[i][l + 1] = temp;
39                      }
40
41                  }
42              }
43              temp = 0;
44              for (int k = 0; k < 3; k++) {
45                  for (int l = 6; l < (9 - 1); l++) {
46                      if (kromosom[i][l] > kromosom[i][l + 1])
47  {
48                          temp = kromosom[i][l];
49                          kromosom[i][l] = kromosom[i][l + 1];
50                          kromosom[i][l + 1] = temp;
51                      }
52
53                  }
54              }
```

```
55
56              temp = 0;
57              for (int k = 0; k < 3; k++) {
58                  for (int l = 9; l < (12 - 1); l++) {
59                      if (kromosom[i][l] > kromosom[i][l + 1])
60  {
61                          temp = kromosom[i][l];
62                          kromosom[i][l] = kromosom[i][l + 1];
63                          kromosom[i][l + 1] = temp;
64                      }
65
66                  }
67              }
68              temp = 0;
69              for (int k = 0; k < 4; k++) {
70                  for (int l = 12; l < (16 - 1); l++) {
71                      if (kromosom[i][l] > kromosom[i][l + 1])
72  {
73                          temp = kromosom[i][l];
74                          kromosom[i][l] = kromosom[i][l + 1];
75                          kromosom[i][l + 1] = temp;
76                      }
77
78                  }
79              }
80          }
81
82          for (int i = 0; i < popsize; i++) {
83              System.out.print("Fitness " + i + " : " +
84  fitnessGabungan[i]);
85              for (int j = 0; j < panjangK; j++) {
86                  System.out.print(" " + kromosom[i][j]);
87              }
88              System.out.println("");
89          }
90
91      }
```

Sourcecode 5.5 Perhitungan *Elitism*

5.2.6 Implementasi Proses Menghitung Fuzzifikasi

Prosses pada fuzzifikasi ini dilakukan disetiap himpunan *Fuzzy*. Berikut merupakan salah satu proses fuzzifikasi pada variabel dari *Fuzzy* yaitu variabel umur dengan masing masing kelas, umur muda dan umur tua yang ditunjukkan pada *Sourcecode* 5.6 Fuzzifikasi.

```
1   public double umurMuda(double x) {
2           if (x >= umur2) {
3               miuUmur1 = 0;
4           } else if (x > umur1 && x < umur2) {
5               miuUmur1 = (umur2 - x) / selisihUmur;
6           } else if (x <= umur1) {
7               miuUmur1 = 1;
8           } return miuUmur1;
9       }
10      public double umurTua(double x) {
11          if (x <= umur1) {
12              miuUmur2 = 0;
```

```
13          } else if (x > umur1 && x < umur2) {
14              miuUmur2 = (x - umur1) / selisihUmur;
15          } else if (x >= umur2) {
16              miuUmur2 = 1;
17          } return miuUmur2;}
```

Sourcecode 5.6 Fuzzifikasi

5.2.7 Implementasi Proses Implikasi

Proses implikasi pada penelitian ini menggunakan metode *Min*. Selain itu, fungsi yang akan dijelaskan pada *Sourcecode* 5.7 Impikasi, juga menampilkan *sourcecode* proses perhitungan derajat keanggotaan (fuzzifikasi), perhitungan *rule* serta defuzzifikasi.

```
1   public double rule(double j, double k, double l, double m,
2   double n) {
3           for (int i = 0; i < 108; i++) { //method rulenya
4   dilakukan sebanyak individu dan perindividu dilakukan
5   sebanyak data
6               a[i] = 0.0;
7               z[i] = 0.0;
8               az[i] = 0.0;
9           }
10          a[108] = 0.0;
11          az[108] = 0.0;
12
13          a[0] = Min(umurMuda(j), kolesRendah(k),
14  trigliRendah(l), LDLRendah(m), HDLBaik(n));
15          z[0] = strokeRendah(a[0]);
16          az[0] = z[0] * a[0];
17
18          a[1] = Min(umurTua(j), kolesRendah(k),
19  trigliRendah(l), LDLRendah(m), HDLBaik(n));
20          z[1] = strokeRendah(a[1]);
21          az[1] = z[1] * a[1];
22
23          a[2] = Min(umurMuda(j), kolesSedang(k),
24  trigliRendah(l), LDLRendah(m), HDLBaik(n));
25          z[2] = strokeRendah(a[2]);
26          az[2] = z[2] * a[2];
27
28          a[3] = Min(umurTua(j), kolesSedang(k),
29  trigliRendah(l), LDLRendah(m), HDLBaik(n));
30          z[3] = strokeRendah(a[2]);
31          az[3] = z[2] * a[2];
32
33          a[4] = Min(umurMuda(j), kolesTinggi(k),
34  trigliRendah(l), LDLRendah(m), HDLBaik(n));
35          z[4] = strokeRendah(a[4]);
36          az[4] = z[4] * a[4];
37
38          a[5] = Min(umurTua(j), kolesTinggi(k),
39        .
40        .
41        .
42        .
43        for (int i = 0; i < 108; i++) {
44                      a[108] += a[i];
```

```
45                  }
46
47                  for (int i = 0; i < 108; i++) {
48                      az[108] += az[i];
49                  }
50                  zFinal = az[108] / a[108];
51                  return zFinal;
52              }
```

Sourcecode 5.7 Implikasi