BAB IV
IMPLEMENTASI DAN PEMBAHASAN

4.1 Lingkungan Implementasi

Implementasi sistem ini berupa aplikasi pemrograman yang menerapkan metode fuzzy *decision tree* dengan algoritma C4.5 untuk klasifikasi data Haberman's Survival. Range yang dipakai dalam data tersebut yaitu umur, tekanan darah, kolesterol, denyut jantung dan oldpeak. Lingkungan implementasi yang akan dijelaskan meliputi lingkungan implementasi perangkat keras dan perangkat lunak.

4.1.1 Lingkungan Perangkat Keras

Perangkat keras yang digunakan pada penelitian ini adalah sebuah PC(*Personal Computer*) dengan spesifikasi sebagai berikut :

1. *Processor* Intel® Core2 Duo 2.10GHz
2. Memori 2GB
3. *Harddisk* dengan kapasitas 160 GB
4. Monitor 14.0"

4.1.2 Lingkungan Perangkat Lunak

Perangkat lunak yang digunakan pada penelitian ini adalah sebagai berikut :

1. Sistem operasi Windows 7 Home Premium.
2. Borland Delphi 7.0
3. Microsoft Accsess

4.2 Implementasi Program

Berdasarkan analisa dan perancangan proses yang telah dipaparkan pada Bab III, maka pada bab ini akan dijelaskan proses-proses implementasinya.

4.2.1 Implementasi Struktur Data

Struktur data untuk menyimpan data pasien, detail pasien, atribut, Range, node, rule dan detail rule direpresentasikan pada *sourcecode* 4.1 sebagai berikut.

```
{pengelompokan range kelas output}
RRangeOut= Record
     ID_Range:string;
     ID_Atribut:string;
     Keterangan:string;
     Nilai_R1Min:string;
     Nilai_R1Max:string;
     Nilai_RMin:string;
     Nilai_RMax:string;
     MF:real;
     SUM_MF:real;
     Juml_MValue:integer;
     Count_Data:integer;
     Nilai_Proporsi: real;
  end;
{pengelompokan range / Var linguistik}
RRange= Record
     ID_Range:string;
     ID_Atribut:string;
     Keterangan:string;
     Nilai_R1Min:string;
     Nilai_R1Max:string;
     Nilai_RMin:string;
     Nilai_RMax:string;
     MF:real;
     HF:real;
     SUM_MF:real;
     Status_Proses:Boolean;
     NRangeOut: array of RRangeOut;
  end;
{pengelompokan atribut}
RAtribut= Record
    ID_Atribut:string;
    Nama_Atribut:string;
    Keterangan_IO:byte;
    IG: real;
    SI: real;
    GR: real;
    Max_Output:string;
    Status_Active:Boolean;
    Max_Width: integer;
     Juml_MPValue:integer;
    NRange: array of RRange;
  end;
{pengelompokan detail data pasien}
RDetailPasien= Record
    ID_Detail_Pasien: string;
```

```
48      ID_Pasien: string;
49      ID_Atribut:string;
50      NRange: array of RRange;
51      Nilai: string;
52      Status_MPValue: Boolean;
53      Status_Active:Boolean;
54    end;
55  {pengelompokan data pasien}
56  RPasien= Record
57      ID_Pasien: string;
58      Nama_Pasien: string;
59      ID_Range_Output: string;
60      Status_MValue: Boolean;
61      DetailPasien: array of RDetailPasien;
62    end;
63  {pengelompokan range node}
64  RNode = Record
65       ID_Node: integer;
66       ID_NodeInduk:integer;
67       ID_RangeInduk: string;
68       Level: integer;
69       ID_Atribut:string;
70       ID_RangeOut: string;
71       GR: real;
72       HF_Seluruh:real;
73       Juml_MValue:integer;
74       Pasien: array of RPasien;
75       Atribut_Input: array of RAtribut;
76       RangeOut: array of RRangeOut;
77       status_proses: Boolean;
78       RuleNode:string;
79    end;
80  {pengelompokan rule}
81  RRule = Record
82      ID_Rule: string;
83      ID_Range_Output: string;
84      N_FCT: integer;
85      N_LDT: integer;
86    end;
87  {pengelompokan isi rule}
88  RDetailRule = Record
89      ID_Detail_Rule: string;
90      ID_Rule: string;
91      ID_Range_Input: string;
92    end;
93
```

Sourcecode 4.1. Struktur Data Record

4.2.2 Implementasi Pelatihan

4.2.2.1 Implementasi Fuzzifikasi

Proses selanjutnya adalah fuzzifikasi atau perhitungan derajat keanggotaan masing-masing atribut. Implementasi perhitungannya ditunjukkan pada *sourcecode* 4.2 sebagai berikut.

```
{proses menghitung mf yang dibagi jadi 4 titik, R1Min,
R1Max, Rmin dan Rmax}
procedure TFProses.Hitung_MF(Data_Input: real);
var
    i:integer;
    Min_Range1,Max_Range1:real;
    Min_Range,Max_Range:real;
Begin
for i:=Low(Input_Range) to High(Input_Range)do
begin
  if Data_Input >=0 then
  begin
        Min_Range1:=strtofloat(Input_Range[i].Nilai_R1Min;
        Max_Range1:=strtofloat(Input_Range[i].Nilai_R1Max;
         Min_Range:=strtofloat(Input_Range[i].Nilai_RMin);
         Max_Range:=strtofloat(Input_Range[i].Nilai_RMax);
        {bentuk kurva naik}
        if i = High(Input_Range) then
        begin
            if Data_Input >= Min_Range1 then
                Input_Range[i].MF:=1
            else
            begin
                if (Data_Input >= Min_Range) And
                    (Data_Input < Min_Range1) then
                begin
                    Input_Range[i].MF:= (Data_Input-
                    Min_Range)/(Min_Range1-Min_Range);
                end
                else
                   Input_Range[i].MF:=0;
            end;
        end
        else {bentuk kurva segitiga dan kurva turun}
        begin
            if (Data_Input >= Min_Range)  And (Data_Input
               <= Max_Range) then
            begin
                if (Data_Input >= Min_Range1)  And
                    (Data_Input <= Max_Range1)then
                       Input_Range[i].MF:=1
                else if (Data_Input >= Min_Range) And
                    (Data_Input < Min_Range1)then
                    Input_Range[i].MF:= (Data_Input-
                      Min_Range)/(Min_Range1-Min_Range)
```

```
46                  else if (Data_Input > Max_Range1) And
47                      (Data_Input <= Max_Range)then
48                       Input_Range[i].MF:= (Max_Range-
49                         Data_Input)/(Max_Range-Max_Range1);
50              end
51              else
52                      Input_Range[i].MF:=0;
53              end;
54      end
55      else
56      begin
57        Input_Range[i].MF:= 1/(High(Input_Range)+1);
58      end;
59    end;
60  end;
```

Sourcecode 4.2. Perhitungan Fuzzifikasi

4.2.2.2 Implementasi Perhitungan *Fuzzy Entropy, Information Gain, Split Info* dan *Gain Ratio*

Perhitungan *fuzzy entropy*, *information gain, split info* dan *gain ratio* digunakan untuk memilih atribut terbaik dari atribut-atribut yang tersedia yang akan dijadikan sebagai *root* dari *decision tree*. Adapun *function* perhitungan *fuzzy entropy* untuk seluruh data dapat dilihat pada *sourcecode* 4.3 sebagai berikut.

```
1   {proses menghitung nilai HF seluruh data}
2   function TFProses.Hitung_HF_Seluruh(No_Node,Juml_Data:
3   integer): real;
4   var
5     Hasil:real;
6     j:integer;
7   begin
8     Hasil:=0;
9     with Node[No_Node] do
10    begin
11       For j:=Low(RangeOut) to High(RangeOut) do
12          if not((Node[No_Node].RangeOut[j].Count_Data-
13            Node[No_Node].RangeOut[j].Juml_MValue=0) or
14            ((Juml_Data-Node[No_Node].Juml_MValue)=0)) then
15               Hasil:= Hasil- (((RangeOut[j].Count_Data-
16                  Node[No_Node].RangeOut[j].Juml_MValue)/
17                  (Juml_Data-Juml_MValue))*
18                  log2((RangeOut[j].Count_Data-
19                  RangeOut[j].Juml_MValue)/(Juml_Data-
20                  Juml_MValue)));
21              end;
22            result:=Hasil;
23  end;
```

Sourcecode 4.3. Perhitungan *Fuzzy Entropy* Seluruh Data

Adapun *function* perhitungan *fuzzy entropy* untuk tiap atribut dapat dilihat pada *sourcecode* 4.4 sebagai berikut.

```
{proses menghitung nilai HF tiap var. Linguistik}
Function TFProses.Hitung_HF(No_Node:integer;
ID_RangeInput: string): real;
Var
  i,j,k:integer;
  ketemu:boolean;
  Hasil:real;
begin
  ketemu:=false;
  Hasil:=0;
  with Node[No_Node] do
  begin
    For i:=low(Atribut_Input) to high(Atribut_Input) do
    Begin
      For j:=low(Atribut_Input[i].NRange) to
         high(Atribut_Input[i].NRange) do
       begin
         if Atribut_Input[i].NRange[j].ID_Range =
               ID_RangeInput then
         begin
            ketemu:=true;
            break;
         end;
       end;
          if ketemu then break;
      end;
      if ketemu then
      begin
        For k:=low(Atribut_Input[i].NRange[j].NRangeOut)
         to high(Atribut_Input[i].NRange[j].NRangeOut)do
         begin
           if not ((Atribut_Input[i].NRange[j].SUM_MF=0)
             or(Atribut_Input[i].NRange[j].NRangeOut[k].SUM
             _MF=0)) then
                Hasil:= Hasil-
                  ((Atribut_Input[i].NRange[j].NRangeOut[k]
                  .SUM_MF)/(Atribut_Input[i].NRange[j].SUM_
                  MF)*log2(Atribut_Input[i].NRange[j].NRang
                  eOut[k].SUM_MF/Atribut_Input[i].NRange[j]
                  .SUM_MF));
         end;
      end;
  end;
  result:=Hasil;
end;
```

Sourcecode 4.4. Perhitungan *Fuzzy Entropy* Variabel Linguistik

Perhitungan *information gain* dilakukan pada masing-masing atribut. Kemudian nilai *information gain* yang diperoleh tersebut digunakan untuk proses perhitungan nilai *gain ratio* atribut. Implementasi perhitungan *information gain* seperti pada *sourccode* 4.5 sebagai berikut.

```
{proses menghitung information gain masing2 atribut,
dengan input hf seluruh data dan jumlah data}
function TFProses.Hitung_IG(No_Node:integer;ID_Atribut1:
string; HF_Seluruh: real;  Juml_data: integer): real;
Var
  i,j:integer;
  ketemu:boolean;
  Hasil:real;
begin
  ketemu:=false;
  Hasil:=0;
  with Node[No_Node] do
  begin
     For i:=low(Atribut_Input)to high(Atribut_Input) do
     begin
       if Atribut_Input[i].ID_Atribut = ID_Atribut1 then
       begin
          ketemu:=true;
          break;
       end;
       if ketemu then break;
     end;
     if ketemu then
      begin
        For j:=low(Atribut_Input[i].NRange) to
        high(Atribut_Input[i].NRange)do
        begin
          if not (Juml_data =0) then
            Hasil:= Hasil +(Atribut_Input[i].NRange[j].
              SUM_MF * Atribut_Input[i].NRange[j].HF /
              (Juml_data-Atribut_Input[i].Juml_MPValue));
        end;
        if Atribut_Input[i].Juml_MPValue > 0 then
          Hasil:= ((Juml_data-Juml_MValue)/Juml_data) *
              (HF_Seluruh -Hasil)
        else
          Hasil:= HF_Seluruh -Hasil;
      end;
  end;
  result:=hasil;
end;
```

Sourcecode 4.5. Perhitungan *Information Gain* Setiap Atribut

Perhitungan *split info* dilakukan pada masing-masing atribut. Kemudian nilai *split info* yang diperoleh digunakan untuk proses perhitungan nilai *gain ratio*. Implementasi perhitungan *split info* seperti pada *sourccode* 4.6 sebagai berikut.

```
{proses menghitung split info, dengan input atribut dan
jumlah data}
function TFProses.Hitung_SI(No_Node: integer; ID_Atribut1:
string; Juml_data: integer): real;
Var
  i,j:integer;
  ketemu:boolean;
  Hasil:real;
begin
  ketemu:=false;
  Hasil:=0;
  with Node[No_Node] do
  begin
   For i:=low(Atribut_Input) to high(Atribut_Input) do
   begin
     if Atribut_Input[i].ID_Atribut = ID_Atribut1 then
     begin
       ketemu:=true;
       break;
     end;
     if ketemu then break;
   end;
   if ketemu then
   begin
     For j:=low(Atribut_Input[i].NRange) to
        high(Atribut_Input[i].NRange)do
     begin
       if not ((Atribut_Input[i].NRange[j].SUM_MF=0) or
         (Juml_data =0)) then
          Hasil:= Hasil-((Atribut_Input[i].NRange[j].
             SUM_MF)/(Juml_data)*log2(Atribut_Input[i].
             NRange[j].SUM_MF/Juml_data));
        end;
     end;
  end;
  result:=hasil;
end;
```

Sourcecode 4.6. Perhitungan *Split Info* Setiap Atribut

Perhitungan *gain ratio* dilakukan pada masing-masing atribut. Kemudian nilai *gain ratio* yang diperoleh digunakan untuk proses pemilihan root dengan mengambil nilai *gain ratio* tertinggi. Implementasi perhitungan *gain ratio* seperti pada *sourccode* 4.7 sebagai berikut.

```
{proses menghitung gain ratio, dengan input atribut yg
akan dihitung}
function TFProses.Hitung_GR(No_Node: integer; ID_Atribut1:
string): real;
Var
  i:integer;
  ketemu:boolean;
  Hasil:real;
begin
  ketemu:=false;
  Hasil:=0;
  with Node[No_Node] do
  begin
    For i:=low(Atribut_Input) to high(Atribut_Input) do
    begin
      if Atribut_Input[i].ID_Atribut = ID_Atribut1 then
      begin
        ketemu:=true;
        break;
      end;
      if ketemu then break;
    end;
  if ketemu then
  begin
    if not(Atribut_Input[i].SI = 0) then
       Hasil:= Atribut_Input[i].IG/Atribut_Input[i].SI;
  end;  end;  result:=hasil;
end;
```

Sourcecode 4.7. Perhitungan *Gain Ratio* Setiap Atribut

Setelah didapatkan nilai *gain ratio* dari masing-masing atribut kemudian dicari nilai *gain ratio* tertinggi untuk menentukan atribut yang terbaik.

4.2.2.3 Implementasi Pembentukan *Tree*

Tahap pembentukan *decision tree* dapat dilakukan setelah didapatkan nilai *gain ratio* tertinggi dari atribut-atribut yang digunakan. Atribut yang memiliki nilai *gain ratio* tertinggi akan digunakan sebagai *root* dari tiap *node* atau *subnode*. Implementasi pembentukan *tree* seperti pada *sourcecode* 4.8 sebagai berikut.

```
{proses pembentukan node tree}
procedure TFProses.Proses;
Var
  i,j,k,l,No_Node,Node_Induk:integer;
  Proses_Lanjut,Ketemu: Boolean;
  MF_Atribut:real;
  Nilai_Proporsi:real;
```

```
8     Status_nol:Boolean;
9   begin
10    No_Node:=0;
11  {perulangan membaca atribut input untuk masing2 var.
12  Linguistiknya. Cari nilai hf seluruh, ig, si dan gr}
13    For i:=low(node[No_node].Atribut_Input) to
14        high(node[No_node].Atribut_Input) do
15    begin
16      if node[No_node].Atribut_Input[i].Status_Active then
17      begin
18        For j:=low(node[No_node].Atribut_Input[i]. NRange)
19          to high(node[No_node]. Atribut_Input[i].NRange) do
20
21          node[No_node].Atribut_Input[i].NRange[j].HF:=
22          RoundTo(Hitung_HF(No_Node,node[No_node].Atribut_In
23          put[i].NRange[j].ID_Range),-5);
24
25          node[No_node].Atribut_Input[i].IG:=RoundTo(Hitung_
26          IG(No_Node,node[No_node].Atribut_Input[i].ID_Atrib
27          ut,node[No_node].HF_Seluruh,high(node[No_node].Pas
28          ien)+1),-5);
29
30          node[No_node].Atribut_Input[i].SI:=RoundTo(Hitung_
31          SI(No_Node,node[No_node].Atribut_Input[i].ID_Atrib
32          ut,high(node[No_node].Pasien)+1),-5);
33
34          node[No_node].Atribut_Input[i].GR:=RoundTo(Hitung_
35          GR(No_Node,node[No_node].Atribut_Input[i].ID_Atrib
36          ut),-5);
37        end;
38    end;
39  Node[No_Node].ID_Atribut:=Get_ID_Atribut_MaxGR(No_Node);
40  {pembentukan node 1 dst}
41    While (not node[0].status_proses) do
42    begin
43      Proses_Lanjut:= false;
44     {jika ada varling yg status prosesnya masih false}
45      For i:= low(Node[No_Node].Atribut_Input) to
46          high(Node[No_Node].Atribut_Input) do
47      begin
48          if Node[No_Node].Atribut_Input[i].ID_Atribut=
49          Node[No_Node].ID_Atribut then
50          begin
51            for j:=low(Node[No_Node].Atribut_Input[i].
52              NRange) to
53              high(Node[No_Node].Atribut_Input[i]. NRange)
54              do
5               begin
5                 if not Node[No_Node].Atribut_Input[i].
5                    NRange[j].Status_Proses then
                  begin
6                    Proses_Lanjut:= true;
57                   break;
58                end;
```

```
59              end;
60            end;
61            if Proses_Lanjut then break;
62          end;
63          {status proses lanjut true}
64          if Proses_Lanjut then
65          begin
66            Node_Induk:= Node[No_Node].ID_Node;
67            No_Node:= high(Node)+1;
68            setlength(node,No_Node+1);
69            Node[No_Node].ID_Node:=No_Node;
70           Node[No_Node].ID_NodeInduk:=Node_Induk;
71           Node[No_Node].ID_Atribut:=Node[Node_Induk].
72           ID_Atribut;
73           Node[No_Node].ID_RangeInduk:=Node[Node_Induk].
74           Atribut_Input[i].NRange[j].ID_Range;
75
76            {membentuk rule}
77            if Node[Node_induk].RuleNode <> '' then
78             Node[No_Node].RuleNode:=Node[Node_induk].
79             RuleNode + ',' +  Node[No_Node].ID_RangeInduk
80           else
81             Node[No_Node].RuleNode:=Node[No_Node].
82             ID_RangeInduk;
83            Node[No_Node].status_proses:=false;
84            Load_Data_Node(No_Node);
85          end;
86      end;
87  end;
```

Sourcecode 4.8. Proses Pembentukan *Tree*

4.2.2.4 Implementasi Perhitungan Proporsi

Proporsi didapatkan dari jumlah nilai derajat keanggotaan dari *record* yang terpilih dibandingkan dengan jumlah derajat keanggotaan semua *record* pada masing-masing kelas. Adapun prosedur perhitungan proporsi seperti pada *sourcecode* 4.9 sebagai berikut.

```
1   {proses menghitung proporsi tiap rangeoutput yg diambil
2   dari sum mf detail pasien. Kemudian dikalikan 100 dengan
3   membagi sum mf keseluruhan}
4   function TFProses.Hitung_Proporsi(No_Node:integer;
5   ID_Range,ID_Range_out: string; SUM_MF_All: real): real;
6   Var
7     x,i,j,k:integer;
8     ketemu:boolean;
9     Hasil:real;
10  begin
11    For i:=low(Node[No_Node].Pasien) to
12       high(Node[No_Node].Pasien)do
```

```
13    begin
14      if Node[No_Node].Pasien[i].ID_Range_Output =
15       ID_Range_out then
16      begin
17         For j:=low(Node[No_Node].Pasien[i].DetailPasien) to
18             high(Node[No_Node].Pasien[i].DetailPasien)do
19         begin
20           if Node[No_Node].Pasien[i].DetailPasien[j].
21           Status_Active then
22            begin
23              for k:= low(Node[No_Node].Pasien[i].
24                 DetailPasien[j].NRange) to
25                 high(Node[No_Node].Pasien[i].
26                 DetailPasien[j].NRange)do
27                 if Node[No_Node].Pasien[i].
28                    DetailPasien[j].NRange[k].ID_Range=ID_Ra
29                    nge then
30                    Node[No_Node].RangeOut[x].SUM_MF:=
31                    Node[No_Node].RangeOut[x].SUM_MF +
32                    Node[No_Node].Pasien[i].DetailPasien[j].
33                    NRange[k].MF;
34                end;
35             end;
36           end;
37       end;
38       if SUM_MF_All > 0 then
39         Hasil:=((Node[No_Node].RangeOut[x].SUM_MF)/
40         SUM_MF_All)*100;
41    end;
42    result:=Hasil;
43  end;
44
```

Sourcecode 4.9. Proses Perhitungan Proporsi

4.2.3 Implementasi Pengujian

Tahap pengujian merupakan proses membandingkan kelas output berdasarkan *rule* atau aturan yang telah dihasilkan dari proses pelatihan dengan data uji. Pengujian dilakukan menggunakan metode inferensi Mamdani. Inferensi Mamdani terdapat 4 tahap, yaitu fuzzifikasi, implikasi, komposisi, dan defuzzifikasi. Implementasi pengujian masing-masing tahap adalah sebagai berikut.

4.2.3.1 Implementasi Fuzzifikasi

Fuzzifikasi merupakan proses mengubah data uji yang masih berupa himpunan *crisp* menjadi himpunan fuzzy berdasarkan derajat keanggotaan yang telah ditentukan. Implementasi fuzzifikasi data uji sama dengan implementasi fuzzifikasi pada saat pelatihan.

4.2.3.2 Implementasi Fungsi Implikasi dan Komposisi

Fungsi implikasi, proses pencarian nilai minimum dari derajat keanggotaan yang terdapat di dalam *rule fuzzy*. Sedangkan fungsi komposisi, solusi himpunan fuzzy diperoleh dengan cara mengambil nilai maksimum aturan. Adapun implementasi fungsi implikasi dan komposisi dapat dilihat pada *Sourcecode* 4.10 sebagai berikut.

```
{proses setelah mendapatkan mf, diambil nilai paling min
disetiap rule yg memenuhi}
procedure TFPengujian.Min_Max;
var
  i,j,m,n:integer;    NMin: array of real;
begin
  For m:= low(Rule) To high(Rule) do
  begin
    Fillchar(NMin,sizeof(NMin),#0);
     For n:= low(Rule[m].AnggotaRule) To
        high(Rule[m].AnggotaRule) do
     begin
       setlength(NMin,n+1);
       NMin[n]:= Rule[m].AnggotaRule[n].Nilai;
    end;
    Rule[m].NMin:= Get_Min(NMin);
  end;
{proses komposisi, mencari nilai max mf untuk tiap kelas
output}
  For m:= low(Output.Range)To high(Output.Range)do
  begin
    j:= -1;
   Fillchar(Output.Range[m].NMinRule,sizeof(Output.
   Range[m].NMinRule),#0);
    For i:= low(Rule) To high(Rule) do
    begin
      If Rule[i].ID_Range_Output = Output.Range[m].
         ID_Range Then
      begin
        inc(j);
        setlength(Output.Range[m].NMinRule,j+1);
        Output.Range[m].NMinRule[j]:= Rule[i].NMin;
      end;
    end;
    Output.Range[m].NMax:= Get_Max(Output.Range[m].
       NMinRule);
  end;
end;
```

Sourcecode 4.10. Fungsi Implikasi dan Komposisi

4.2.3.3 Implementasi Defuzzifikasi

Metode defuzzyfikasi yang dipakai pada komposisi aturan Mamdani, yaitu metode *centroid (center of gravity)*. Implementasi perhitungan titik potong dapat dilihat pada *sourcecode* 4.11.

```
{proses menghitung titik potong}
function TFPengujian.Hitung_TitikPotong(Min_Range,
Max_Range, MF: real;Status_NaikTurun: Boolean): real;
begin
  If Status_NaikTurun Then //Naik
     Result:=((Max_Range-Min_Range)*MF)+Min_Range
  Else //Turun
     Result:= Max_Range-((Max_Range-Min_Range)* MF);
end;

```

Sourcecode 4.11. Fungsi Perhitungan Titik Potong

Pada metode ini solusi crisp diperoleh dengan cara mengambil titik pusat daerah *fuzzy*. Implementasi *Center Of Gravity* dapat dilihat pada *sourcecode* 4.12.

```
{proses menghitung nilai COG dengan skala 0,2 dan
dikalikan dengan mf. Kemudian menghitung luas daerah
sesuai dg titik potong.}
function TFPengujian.Hitung_COG: real;
Var
  R,ELuas1,ELuas2:real;
  j:integer;
  Luas_1: array of real;
  Skala_COG:real;
begin
  //Hitung COG(Defuzzy)
  ELuas1:= 0;
  ELuas2:= 0;
  R:= 0;
  Skala_COG:=0.2;
  While R < Titik_Potong.Elements[0] do
  begin
    ELuas1:= ELuas1 + R;
    If R > 0 Then
       ELuas2:= ELuas2 + Titik_Potong.MF[0];
    R:= R + Skala_COG;
  end;

  R:= R - Skala_COG;
  If R < Titik_Potong.Elements[0] Then
  begin
    ELuas1:= ELuas1 + Titik_Potong.Elements[0];
    ELuas2:= ELuas2 + Titik_Potong.MF[0];
```

```
29      R:= R + Skala_COG;
30      If R = Titik_Potong.Elements[0] Then
31         R:= R + Skala_COG;
32    End;
33
34    ELuas1:= ELuas1 * Titik_Potong.MF[0];
35    For j:= low(Titik_Potong.Elements) To
36         high(Titik_Potong.Elements) do
37    begin
38      setlength(Luas_1,j+1);
39      If (j + 1) <= high(Titik_Potong.Elements) Then
40      begin
41        While R < Titik_Potong.Elements[j + 1] do
42        begin
43          Luas_1[j]:= Luas_1[j] + R;
44          ELuas2:= ELuas2 + Titik_Potong.MF[j];
45          R:= R + Skala_COG;
46        end;
47
48        R:= R - Skala_COG;
49        Luas_1[j]:= Luas_1[j] * Titik_Potong.MF[j];
50
51        If R < Titik_Potong.Elements[j + 1] Then
52        begin
53          Luas_1[j]:= Luas_1[j] + (Titik_Potong.Elements[j]
54             + 1] * Titik_Potong.MF[j + 1]);
55          ELuas2:= ELuas2 + Titik_Potong.MF[j + 1];
56          R:= R + Skala_COG;
57          If R = Titik_Potong.Elements[j + 1] Then
58           R:= R + Skala_COG;
59        end;
60      end
61    end;
62    For j:= low(Luas_1) To high(Luas_1) do
63       ELuas1:= ELuas1 + Luas_1[j];
64    Result:= ELuas1 / ELuas2;
65  end;
```

Sourcecode 4.12. Perhitungan Nilai COG

4.3 Implementasi Antarmuka

Implementasi antarmuka sistem terdiri 3 bagian utama, yaitu:

1. Form Data Pasien

   Form data pasien digunakan sebagai antarmuka untuk memasukkan data baru pasien, mengubah dan menghapus data pasien yang tersimpan dalam *database*.

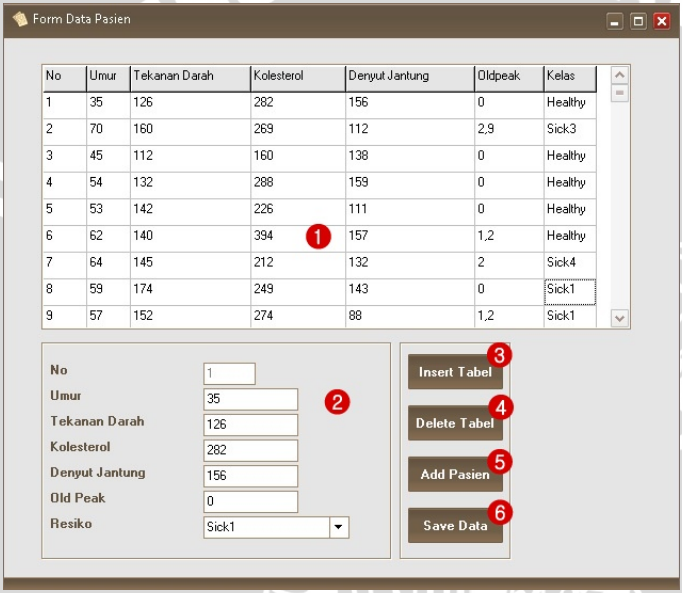
2. Form Pelatihan

   Form data uji digunakan sebagai antarmuka untuk melakukan pembentukan *decision tree* dan menampilkan *rule* yang terbentuk.

3. Form Pengujian
   Form pengujian digunakan sebagai antarmuka untuk melakukan pengujian *rule* atau aturan yang terbentuk dengan menggunakan data uji serta perhitungan tingkat akurasinya.

4.3.1 Form Data Pasien

Terdapat 1 halaman pada form data pasien yang menampilkan isi data pasien yang tersimpan dalam *database*. Hasil perancangan antarmuka ditunjukan seperti pada Gambar 4.1 sebagai berikut.



Gambar 4.1. Antarmuka Data Pasien

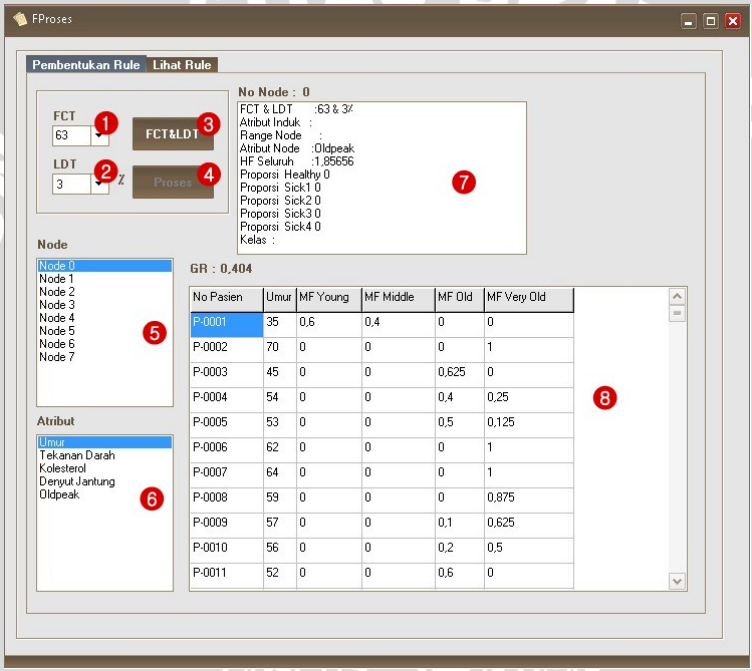
Keterangan Gambar 4.1:

1. Tabel yang menampilkan nilai atribut setiap pasien yang tersimpan dalam *database*.
2. *Field-field* untuk memasukkan nilai atribut pasien seperti umur, tekanan darah, kolesterol, denyut jantung, oldpeak dan kelas.
3. Tombol untuk memasukkan nilai atribut yang telah diisikan pada *field* kedalam tabel.
4. Tombol untuk menghapus data pasien yang tampil pada tabel.
5. Tombol untuk menambah data pasien.

6. Tombol untuk menyimpan nilai atribut pasien yang telah dimasukkan ke dalam *database*.

4.3.2 Form Pelatihan

Terdapat 2 halaman inti pada form pelatihan, yaitu halaman pembentukan *decision tree* dan halaman *rule* yang terbentuk. Hasil perancangan antarmuka pelatihan pembentukan *decision tree* ditunjukkan seperti pada Gambar 4.2 sebagai berikut.



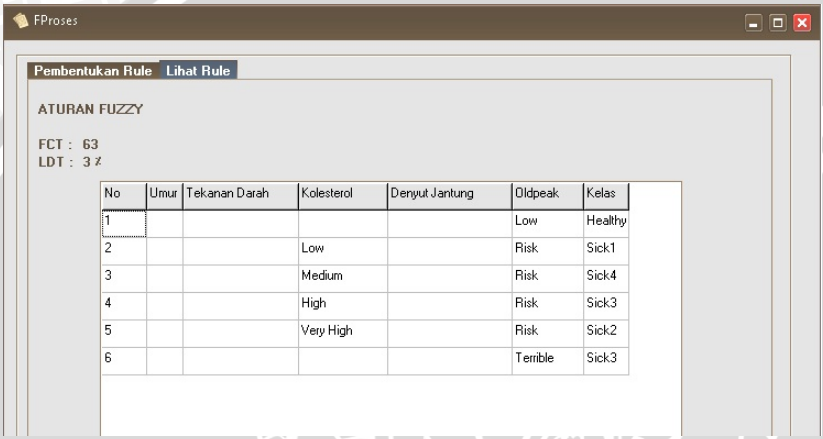
Gambar 4.2. Antarmuka Pembentukan Aturan

Keterangan Gambar 4.2:

1. ComboBox untuk memilih besar FCT dengan nilai 50-90%.
2. ComboBox untuk memilih besar LDT dengan pilihan nilai 3%, 5%, 8%, 10% dan 15%.
3. Tombol untuk mengatur besar FCT dan LDT kedalam program setelah nilainya dimasukkan.
4. Tombol memulai proses pembentukan *tree*.
5. List menampilkan node yang terbentuk dari hasil pelatihan.

6. List menampilkan atribut sesuai dengan node yang terbentuk.
7. List menampilkan keterangan setiap node yang terbentuk.
8. Tabel menampilkan derajat keanggotaan variabel linguistik.

Hasil perancangan antarmuka halaman aturan ditunjukkan seperti pada Gambar 4.3 sebagai berikut.

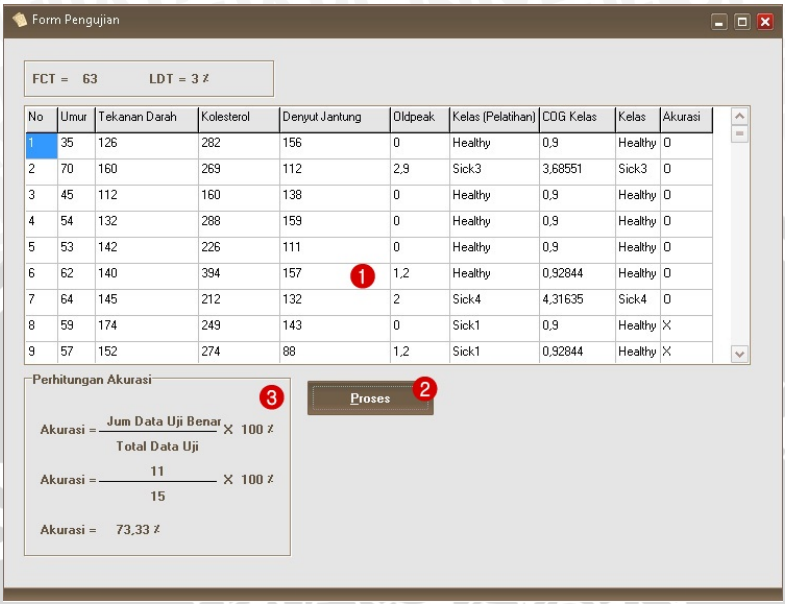


No	Umur	Tekanan Darah	Kolesterol	Denyut Jantung	Oldpeak	Kelas
1					Low	Healthy
2			Low		Risk	Sick1
3			Medium		Risk	Sick4
4			High		Risk	Sick3
5			Very High		Risk	Sick2
6					Terrible	Sick3

Gambar 4.3. Antarmuka Lihat Aturan

4.3.3 Form Pengujian

Hasil perancangan antarmuka pengujian ditunjukkan seperti pada Gambar 4.4 sebagai berikut.



Gambar 4.4. Antarmuka Pengujian

Keterangan Gambar 4.4:

1. Tabel yang menampilkan data pasien, nilai COG, hasil klasifikasi dan akurasi.
2. Tombol memulai proses pengujian.
3. *GroupBox* yang menampilkan perhitungan akurasi.

4.4 Sistematika Pengujian

Pada pengujian tingkat akurasi, kelas output yang dihasilkan dibandingkan dengan kelas output pada data asli. Untuk mendapatkan kelas output, dilakukan proses inferensi mamdani yang disesuaikan dengan *rule-rule* yang terbentuk dalam proses pengujian sebelumnya. Pengujian ini dilakukan sebanyak 40 kali pada 3 jumlah data latih yang berbeda yaitu 70, 140 dan 210 data. Tingkat akurasi ini kemudian di rata-rata berdasarkan kombinasi nilai FCT dan LDT nya untuk mendapatkan tingkat akurasi sistem.

4.4.1 Uji Coba

4.4.1.1 Data Uji 1

Data uji pertama yang digunakan adalah data uji dengan jumlah data sebanyak 30 data. Data latih sebanyak 70 data yang telah mengalami proses pelatihan akan membangkitkan aturan atau *rule*. Aturan tersebut akan disesuaikan dengan data uji. Hasil pengujian akurasi dengan data uji sebanyak 30 data ditunjukkan pada Tabel 4.1.

Tabel 4.1. Pengujian Akurasi dengan 30 Data Uji

Akurasi					
FCT	LDT				
	3 %	5 %	8 %	10 %	15 %
50 %	73,33%	73,33%	73,33%	73,33%	73,33%
60 %	70%	70%	70%	70%	73,33%
65 %	63,33%	63,33%	63,33%	63,33%	76,67%
70 %	56,67%	60%	60%	60%	76,67%
75 %	56,67%	60%	60%	60%	76,67%
80 %	60%	60%	60%	60%	76,67%
85 %	60%	60%	60%	60%	76,67%
90 %	60%	60%	60%	60%	76,67%

Pada tabel ditunjukkan bahwa akurasi paling rendah adalah 56,67% yaitu pada nilai FCT 70% dan 75% dengan nilai LDT yang sama 3%. Sedangkan akurasi paling tinggi adalah 76,67% yaitu diperoleh pada nilai FCT 65% sampai 90% dengan nilai LDT yang sama sebesar 15%.

4.4.1.2 Data Uji 2

Data uji kedua yang digunakan adalah data uji dengan jumlah data sebanyak 60 data. Data latih sebanyak 140 data yang telah mengalami proses pelatihan akan membangkitkan aturan atau *rule* seperti pada pengujian sebelumnya. Aturan tersebut akan disesuaikan dengan data uji. Hasil pengujian akurasi dengan data uji sebanyak 60 data ditunjukkan pada tabel 4.2 sebagai berikut.

Tabel 4.2. Pengujian Akurasi dengan 60 Data Uji

Akurasi					
FCT	LDT				
	3 %	5 %	8 %	10 %	15 %
50 %	55%	56,67%	56,67%	56,67%	56,67%
60 %	55%	55%	55%	55%	53,33%
65 %	51,67%	53,33%	55%	55%	53,33%
70 %	51,67%	53,33%	53,33%	51,67%	50%
75 %	51,67%	53,33%	51,67%	50%	48,33%
80 %	51,67%	53,33%	53,33%	51,67%	50%
85 %	48,33%	51,67%	53,33%	53,33%	50%
90 %	48,33%	53,33%	53,33%	53,33%	50%

Pada tabel di atas ditunjukkan bahwa akurasi yang dihasilkan tidak sama dengan pengujian sebelumnya yang menggunakan data latih sebanyak 70 data. Akurasi paling rendah adalah 48,33% pada nilai FCT 75%, 85%, 90% dengan kombinasi nilai LDT masing-masing 15%, 3%, 3%. Sedangkan akurasi paling tinggi adalah 56,67% pada nilai FCT 50% dan nilai LDT 5% sampai 15%.

4.4.1.3 Data Uji 3

Data uji ketiga yang digunakan adalah data uji dengan jumlah data sebanyak 210 data. Data latih yang telah mengalami proses pelatihan akan membangkitkan aturan atau *rule* seperti pada pengujian sebelumnya. Aturan tersebut akan disesuaikan dengan data uji. Hasil pengujian akurasi dengan data latih sebanyak 210 data ditunjukkan pada Tabel 4.3 sebagai berikut.

Tabel 4.3. Pengujian Akurasi dengan 90 Data Uji

Akurasi					
FCT	LDT				
	3 %	5 %	8 %	10 %	15 %
50 %	62,22%	62,22%	61,11%	61,11%	61,11%
60 %	65,56%	62,22%	64,44%	64,44%	62,22%
65 %	65,56%	62,22%	64,44%	64,44%	62,22%
70 %	60%	56,67%	62,22%	62,22%	60%
75 %	61,11%	57,78%	63,33%	63,33%	61,11%
80 %	62,22%	55,56%	61,11%	61,11%	58,89%

85 %	62,22%	55,56%	61,11%	61,11%	58,89%
90 %	62,22%	55,56%	61,11%	61,11%	58,89%

Pada tabel di atas ditunjukkan bahwa akurasi yang dihasilkan tidak sama dengan pengujian sebelumnya yang menggunakan data latih 70 dan 140 data. Akurasi paling rendah adalah 55,56 % pada nilai FCT 80% sampai 85% dan nilai LDT 5%. Sedangkan akurasi paling tinggi adalah 65,56% yang diperoleh pada nilai FCT 60% sampai dengan 65% untuk nilai LDT yang sama yaitu 3%.

4.4.2 Analisa Hasil

Proses pengujian menggunakan 3 macam data uji dengan jumlah data yang berbeda. Akurasi yang dihasilkan memiliki nilai yang berbeda untuk setiap kombinasi nilai FCT dan LDT. Hal ini disebabkan karena pada pelatihan yang dilakukan sebelumnya menghasilkan *rule-rule* dengan kelas output yang berbeda dengan kelas output pada data uji yang sebenarnya, sehingga ketidakcocokan ini mempengaruhi nilai akurasi klasifikasi. Semakin besar ketidakcocokan dengan *rule*, maka akurasi akan semakin menurun. Hasil pengujian pada 3 macam data uji diatas, dapat diambil nilai rata-rata akurasi klasifikasi seperti ditunjukkan pada Tabel 4.4 sebagai berikut.

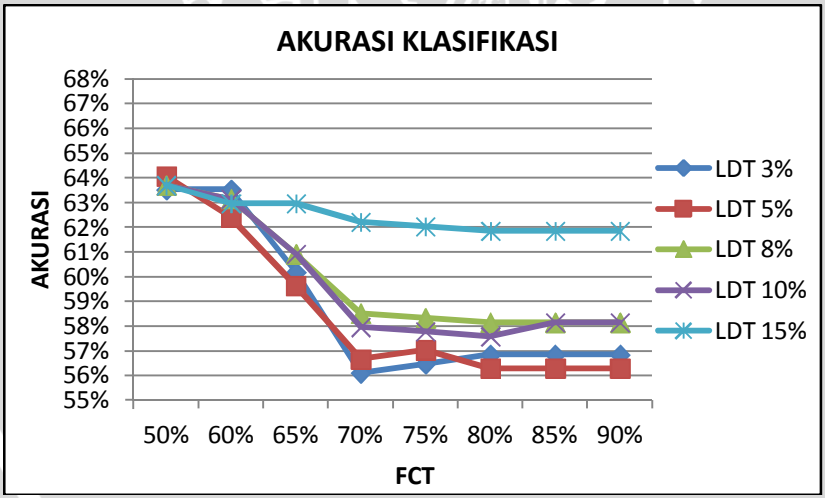
Tabel 4.4 Rata-rata Pengujian Akurasi Klasifikasi

Akurasi					
FCT	LDT				
	3 %	5 %	8 %	10 %	15 %
50 %	63,52%	64,07%	63,70%	63,70%	63,70%
60 %	63,52%	62,41%	63,15%	63,15%	62,96%
65 %	60,19%	59,63%	60,92%	60,92%	62,96%
70 %	56,11%	56,67%	58,52%	57,96%	62,22%
75 %	56,48%	57,04%	58,33%	57,78%	62,04%
80 %	56,85%	56,30%	58,15%	57,59%	61,85%
85 %	56,85%	56,30%	58,15%	58,15%	61,85%
90 %	56,85%	56,30%	58,15%	58,15%	61,85%

Dari Tabel 4.4 diatas, dapat dilihat bahwa kinerja algoritma FC4.5 mengalami penurunan jika nilai FCT semakin besar dan atau nilai LDT yang semakin kecil, walaupun penurunan yang terjadi tidaklah signifikan sehingga masih dapat ditoleransi. Kondisi ini disebabkan karena terjadinya *overfitting*. *Overfitting* adalah terlalu tingginya nilai FCT yang digunakan pada saat pelatihan, sehingga *tree* akan terus diekspansi sampai betul-betul sesuai dengan pelatihan. Akibatnya *tree* memiliki node-node yang mengandung data yang mengalami kesalahan klasifikasi.

Nilai FCT terlalu tinggi atau nilai LDT terlalu rendah dapat menghasilkan *tree* dengan ukuran yang besar dan *rule* yang dihasilkan banyak dan bervariasi karena *tree* akan diekspansi sampai *leaf-node* terdalam atau sampai tidak ada atribut lagi. Sebaliknya, nilai FCT yang terlalu rendah dan atau nilai LDT yang terlalu tinggi akan menghasilkan *tree* dengan ukuran yang kecil sehingga *rule* yang dihasilkan juga sedikit. Hal ini terjadi karena *tree* yang sedang dibangun mengalami *pruning* atau pemotongan.

Grafik perbandingan akurasi klasifikasi menggunakan data uji dapat ditunjukkan pada Gambar 4.5 sebagai berikut.



Gambar 4.5 Grafik Tingkat Akurasi Klasifikasi

Akurasi tertinggi pada pengujian akurasi dicapai pada nilai FCT sebesar 50% dengan nilai LDT sebesar 5% yaitu 64,07%. Sedangkan

akurasi terendah adalah 56,11% pada nilai FCT sebesar 70% dengan nilai LDT sebesar 3%. Akurasi yang diperoleh hanya berkisar antara 64% sampai 56%, hal ini dapat tergolong akurasi yang rendah jika dibandingkan dengan metode klasifikasi yang lain seperti *fuzzy decision tree* dengan algoritma ID3 pada data diabetes dengan akurasi sebesar 94,15%. Setelah dilakukan analisa, hal ini disebabkan karena ketidakcocokan kelas output *rule* yang terbentuk dengan kelas output data sebenarnya. Berikut ini adalah beberapa contoh *rule* yang terbentuk pada 10 data latih.

1. IF tekanan darah low AND oldpeak low THEN healthy
2. IF tekanan darah medium AND oldpeak low THEN healthy
3. IF tekanan darah high AND oldpeak low THEN sick1
4. IF tekanan darah very high AND oldpeak low THEN sick1
5. IF kolesterol low AND oldpeak risk THEN sick1
6. IF kolesterol medium AND oldpeak risk THEN sick4
7. IF kolesterol high AND oldpeak risk THEN sick3
8. IF oldpeak terrible THEN sick3

Dari *rule* diatas jika diberikan sebuah data pengujian umur 48, tekanan darah 124, kolesterol 274, denyut jantung 166, *oldpeak* 0,5 dan kelas outputnya sick3, apabila dibandingkan dengan *rule* yang terbentuk diatas maka hasil klasifikasinya harusnya masuk kelas sehat (*rule* 1). Ketidakcocokan kelas output *rule* yang terbentuk dengan data sebenarnya inilah yang menyebabkan turunnya akurasi.

Selain faktor ketidakcocokan *rule*, faktor data penyakit jantung yang digunakan untuk pelatihan dan pengujian terdapat beberapa data yang jelek, atau hasil klasifikasinya meleset jauh tidak sesuai dengan perbandingan dari kelima parameter yang digunakan. Misalnya, parameter masih tergolong kategori normal atau sehat tetapi kelas output datanya termasuk kelas sick3. Hal ini juga bisa menurunkan akurasi, karena nantinya *rule* yang terbentuk akan masuk kedalam kelas sick3, sehingga pada saat pengujian jika terdapat data yang sama atau kelas outputnya dalam kategori sehat maka kelas output klasifikasinya akan menghasilkan kelas sick3.

Terdapat beberapa hasil akurasi yang sama untuk kombinasi FCT dan LDT, seperti pada FCT 80% dan 90% dengan masing-masing kombinasi nilai LDT dari 3% sampai 15% Persamaan nilai akurasi ini disebabkan karena perbedaan *rule* yang terbentuk tidak

berbeda jauh sehingga perbedaan *rule* tersebut tidak mempengaruhi proses pengujian pada data uji yang digunakan.

Dari keseluruhan uji coba, nilai FCT dan LDT sangat berpengaruh terhadap *rule* yang dihasilkan. Semakin bervariasi *rule* maka mempengaruhi besarnya akurasi pada proses pengujian. Semakin besar nilai FCT belum tentu semakin besar tingkat akurasi klasifikasi. Begitu juga dengan perubahan nilai LDT yang semakin rendah, akurasi klasifikasi juga belum tentu akurasinya tinggi.


UNIVERSITAS BRAWIJAYA