BAB IV

IMPLEMENTASI

Implementasi merupakan proses transformasi representasi rancangan ke bahasa pemrograman yang dapat dimengerti oleh komputer. Pada bab ini akan dibahas hal-hal yang berkaitan dengan implementasi *clustering* dengan algoritma *fuzzy c-means*.

4.1 Lingkungan Implementasi

Implementasi merupakan proses transformasi representasi rancangan ke dalam bahasa pemrograman yang dapat dimengerti oleh komputer. Pada bab ini, lingkungan implementasi yang akan dijelaskan meliputi lingkungan implementasi perangkat keras dan perangkat lunak.

4.1.1 Lingkungan Perangkat Keras

Perangkat keras yang digunakan dalam pengembangan dan pengujian sistem *Clustering* ini adalah sebuah *notebook* dengan spesifikasi sebagai berikut :

1. Prosesor Intel Dual Core @1.86GHz
2. Memori 1 GB
3. Harddisk dengan kapasitas 120 GB
4. Monitor 14”
5. Keyboard dan Mouse.

Perangkat keras ini akan difungsikan sebagai tempat perangkat lunak untuk penelitian dijalankan.

4.1.2 Lingkungan Perangkat Lunak

Perangkat lunak yang digunakan dalam pengembangan sistem *clustering* ini adalah :

1. Sistem operasi *Windows 7™ Ultimate 32-bit* sebagai tempat aplikasi dijalankan.
2. *Microsoft Visual C# 2010* sebagai *programming softwaredevelopment* dalam pembuatan sistem *clustering*.
3. *Microsoft office excel 2007.*

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 Penanganan Data

Dalam kelas ini dilakukan pengolahan data yang akan digunakan dalam proses *clustering*. Data yang digunakan memiliki format csv. Pengolahan data yang dilakukan yaitu proses pengambilan data dan proses pembacaan data.

4.2.1.1 Proses Pengambilan Data

No	Source Code
1	`DialogResult file = openFile.ShowDialog();`
2	`file == DialogResult.OK)`
3	`{`
4	`List<string[]> parsedData = newList<string[]>();`

Kode Sumber 4.1 *Load csv*

Data yang akan diolah diambil melalui form.cs dengan prosedur *button browse*.

4.2.1.2 Proses Pembersihan Dan Pembacaan Data

No	Source Code
1	`using (StreamReader readFile =`
2	`newStreamReader(openFile.FileName))`
3	`{`
4	`while ((line = readFile.ReadLine()) != null)`
5	`{`
6	`row = line.Split(',');`
7	`parsedData.Add(row);`
8	`}`
9	`}`
10	`}`
11	`catch (Exception er)`
12	`{`
13	`MessageBox.Show(er.Message);`
14	`}`
15	`.......`
16	`DataTable testTable = newDataTable();`
17	`foreach (string column in realdata[0])//membuat kolom`
18	`sebanyak jumlah items dalam sebaris dataset`
19	`{`
20	`testTable.Columns.Add();`
21	`att++;`
22	`}`

```
23  foreach (string[] r in realdata)//memasukkan
24                  {
25                      testTable.Rows.Add(r);
26                      rec++;
27                  }
28                 DGset.DataSource = testTable;//memasukkan
29  item dalam testTable kedalam datagridview DGset
```

Kode Sumber 4.2 Pembacaan dan Pembersihan Data

Proses pembersihan data dilakukan setelah pembacaan data. Pada saat program mengambil file dalam format csv, maka data akan dibaca per baris dan dipisahkan oleh koma. Untuk itu perlu dilakukan proses pembersihan koma agar dapat ditempatkan dalam array.

Setelah data diperoleh dan dilakukan pembersihan, maka proses selanjutnya adalah pembacaan data. Data ditempatkan dalam *array Index* Posisi yang nantinya akan ditampilkan dalam Data *Grid View* sehingga dapat dibaca oleh user.

4.2.2 Implementasi Algoritma

4.2.2.1 Penanganan Data Input

Fungsi ini berisi proses pengambilan data input pelanggan yang telah disimpan sementara dalam array. Data ini yang nantinya akan digunakan dalam perhitungan.

No	Source Code
1	`doubledata = newdouble[rec][];`
2	`for (int a = 0; a < rec; a++)`
3	`{`
4	`doubledata[a] = newdouble[att];`
5	`}`
6	`for (int a = 0; a < rec; a++)`
7	`{`
8	`for (int b = 0; b < att; b++)`
9	`{`
10	`doubledata[a][b] = Convert.ToDouble(realdata[a][b]);`
11	`} }`

Kode Sumber 4.3 Pengambilan *Input* Data

4.2.2.2 Inisialisasi Derajat Keanggotaan Awal

Dalam proses algoritma *fuzzy c-means* langkah awal dilakukan pembangkitan bilangan *random* berupa matriks partisi awal derajat keanggotaan.

```
Random random = newRandom();
double[,] randomMatrix =
newdouble[rec,Convert.ToInt16(NUDcluster.Value)];
for (int a = 0; a < rec; a++)
            {
for (int b = 0; b < k; b++)
                {
while (randomMatrix[a, b] == 0 || randomMatrix[a, b] == 1)
                    {
                        randomMatrix[a, b] =
Math.Round(Math.Pow(random.NextDouble(), bobot), 5);
                    }
                }
            }
double[][][] dK =
newdouble[Convert.ToInt32(NUDcluster.Value)][][];
for (int a = 0; a < k; a++)
            {
                dK[a] = newdouble[rec][];
for (int b = 0; b < rec; b++)
                {
                    dK[a][b] = newdouble[att];
                }
            }
for (int a = 0; a < k; a++)
            {
for (int b = 0; b < rec; b++)
                {
for (int c = 0; c < att; c++)
                    {
     dK[a][b][c] = doubledata[b][c] * randomMatrix[b, a];
                    }
                }
            }
```

Kode Sumber 4.4 Pembangkitan Nilai Awal Derajat Keanggotaan

4.2.2.3 Menghitung Nilai Pusat Cluster

Dalam fungsi ini dilakukan perhitungan pusat icluster dari setiap iterasi. Proses ini dapat dilihat dalam kode sumber 4.5

```
double[] jumMIK =
newdouble[Convert.ToInt32(NUDcluster.Value)];
double[,] vKJ = newdouble[Convert.ToInt32(NUDcluster.Value),
att];

for (int a = 0; a < k; a++)
            {
for (int b = 0; b < rec; b++)
                {
                    jumMIK[a] += randomMatrix[b, a];
                }
            }
for (int a = 0; a < k; a++)
            {
for (int b = 0; b < att; b++)
                {
for (int c = 0; c < rec; c++)
                    {
                        vKJ[a, b] += dK[a][c][b];
                    }
                }
            }
for (int a = 0; a < k; a++)
            {
for (int b = 0; b < att; b++)
                {
                    vKJ[a, b] /= jumMIK[a];
                }
            }
```

No	Source Code

Kode Sumber 4.5 Perhitungan Nilai Pusat Cluster

4.2.2.4 Menghitung Fungsi Objektif

Nilai fungsi objektif nantinya akan digunakan sebagai salah satu syarat untuk berhentinya proses *clustering* pada algoritma FCM.

```
double[,] totalFO =
newdouble[Convert.ToInt32(NUDcluster.Value), rec];
double[][][] FO =
newdouble[Convert.ToInt32(NUDcluster.Value)][][];
for (int a = 0; a < k; a++)
            {
                FO[a] = newdouble[rec][];
for (int b = 0; b < rec; b++)
                {
                    FO[a][b] = newdouble[att];
```

No	Source Code

```
11                  }
12              }
13  for (int a = 0; a < k; a++)
14              {
15  for (int b = 0; b < rec; b++)
16                  {
17  for (int c = 0; c < att; c++)
18                      {
19          FO[a][b][c] = Math.Abs(Math.Pow((doubledata[b][c] -
20  vKJ[a, c]), 2));
21                      }
22                  }
23              }
24  for (int a = 0; a < k; a++)
25              {
26  for (int b = 0; b < rec; b++)
27                  {
28              totalFO[a, b] = FO[a][b].Sum();
29                  }
30              }
31  double[,] p = newdouble[rec,
32  Convert.ToInt16(NUDcluster.Value)];
33  double[] pKlaster = newdouble[rec];
34  double pTotal = 0;
35  for (int a = 0; a < rec; a++)
36              {
37  for (int b = 0; b < k; b++)
38                  {
39          p[a, b] = totalFO[b, a] * randomMatrix[a, b];
40                  }
41              }
42  for (int a = 0; a < rec; a++)
43              {
44  for (int b = 0; b < k; b++)
45                  {
46                      pKlaster[a] += p[a, b];
47                  }
48                  pTotal += pKlaster[a];
              }
```

Kode Sumber 4.6 Perhitungan Nilai Fungsi Objektif

4.2.2.5 Cek Kondisi Berhenti

Proses akan berhenti apabila nilai fungsi objektif pada iterasi ke t dan t-1 nilainya lebih kecil daripada nilai minimum erorr. Selain itu pengecekan dilakukan apabila iterasi lebih kecil dari maksimum iterasi.

No	Source Code
1	do
2	.....
3	while ((Math.Abs(pTotal - pTotalBaru) <Math.Pow(10, -
4	Convert.ToDouble(NUDepsilon.Value))) || (iterasi
5	<Convert.ToInt32(NUDiterasi.Value)));

Kode Sumber 4.7 Cek Kondisi Berhenti

4.2.2.6 Update Matrix U Baru

Proses berikutnya adalah pembentukan matriks U baru yang nantinya akan digunakan dalam perulangan perhitungan. Perulangan sebanyak iterasi ini bertujuan untuk memperbaiki pusat *cluster* dan derajat keanggotaan tiap titik data. Perubahan ini diharapkan pusat *cluster* akan bergeser ke titik nilai yang tepat.

No	Source Code
1	double[] jumlahFO = newdouble[rec];
2	double[,] newMatriks = newdouble[rec,
3	Convert.ToInt16(NUD*cluster*.Value)];
4	for (int a = 0; a < rec; a++)
5	{
6	for (int b = 0; b < k; b++)
7	{
8	jumlahFO[a] +=
9	Math.Pow(totalFO[b, a], -1 / (bobot - 1));
10	
11	}
12	}
13	for (int a = 0; a < rec; a++)
14	{
15	for (int b = 0; b < k; b++)
16	{
17	newMatriks[a, b] =
18	Math.Round(Math.Pow(totalFO[b, a], -1 / (bobot - 1)) /
19	jumlahFO[a], 5);
20	}
21	}

Kode Sumber 4.8 *Update* Matriks U Baru

Setelah proses pembentukan matriks di atas, maka akan dilakukan perulangan untuk mendapatkan nilai derajat keanggotaan, nilai pusat *cluster* dan fungsi objektif yang baru sampai memenuhi kondisi perualangan berhenti. Output FCM adalah deretan pusat *cluster* dan derajat keanggotaan data terhadap setiap *cluster*.

4.3 Implementasi Validasi Data

Pada kelas ini akan dilakukan operasi perhitungan rasio, yang merupakan perhitungan validasi dari data yang telah dikelompokkan.

4.3.1 Menghitung *Compactness* Data

Fungsi ini dilakukan untuk mengetahui rasio kepadatan data dari *cluster* yang telah terbentuk. Perhitungan dilakukan berdasar jarak antara data input serat pusat *cluster* dan dibandingkan dengan derajat keanggotaan *cluster*.

No	Source Code
1	`double[][][] compactness =`
2	`newdouble[Convert.ToInt16(NUDcluster.Value)][][];`
3	`double[][][] compact =`
4	`newdouble[Convert.ToInt16(NUDcluster.Value)][][];`
5	`double[][] totalCompact = newdouble[rec][];`
6	`double[] totalCompactness =`
7	`newdouble[Convert.ToInt16(NUDcluster.Value)];`
8	`for (int a = 0; a < k; a++)`
9	`{`
10	`compactness[a] = newdouble[rec][];`
11	`compact[a] = newdouble[rec][];`
12	`for (int b = 0; b < rec; b++)`
13	`{`
14	`compactness[a][b] = newdouble[att];`
15	`compact[a][b] = newdouble[att];`
16	`}`
17	`}`
18	`for (int a = 0; a < k; a++)`
19	`{`
20	`for (int b = 0; b < rec; b++)`
21	`{`
22	`for (int c = 0; c < att; c++)`
23	`{`
24	`compactness[a][b][c] = Math.Abs(Math.Pow(FO[a][b][c] -`
25	`vKJ[a, c], 2));`
26	`}`

```
27  }
28  }
29
30  for (int a = 0; a < k; a++)
31  {
32  for (int b = 0; b < rec; b++)
33  {
34  for (int c = 0; c < att; c++)
35  {
36  compact[a][b][c] = newMatriks[b, a] * compactness[a][b][c] /
37  rec;
38  }
39  }
40  }
41
42  for (int a = 0; a < rec; a++)
43  {
44  totalCompact[a] =
45  newdouble[Convert.ToInt16(NUDcluster.Value)];
46  }
47
48  for (int a = 0; a < rec; a++)
49  {
50  for (int b = 0; b < k; b++)
51  {
52  totalCompact[a][b] = compact[b][a].Sum();
53  }
54  }
55
56  for (int a = 0; a < k; a++)
57  {
58  for (int b = 0; b < rec; b++)
59  {
60  textBox2.Text = Convert.ToString(totalCompactness[a] +=
61  totalCompact[b][a]);
62  }
    }
```

Kode Sumber 4.9 *Compactness* Data

4.3.2 Menghitung *Separation* Data

Fungsi ini bertujuan untuk mengetahui rasio keterpisahan antar data *cluster*. Perhitungan dicari melalui jarak minimum antar vektor pusat *cluster* yang telah terbentuk.

No	Source Code
1	`List<double> Dmin = newList<double>();`
2	`for (int a = 0; a < k; a++)`
3	`{`
4	`for (int b = 0; b < att; b++)`
5	`{`
6	`for (int c = b; c < att; c++)`
7	`{`
8	`Dmin.Add(Math.Pow(vKJ[a, b] * vKJ[a,`
9	`c], 2));`
10	`}`
11	`}`
12	`}`

Kode Sumber 4.10 *Separation Data*

4.3.3 Perhitungan Indeks Xie dan Beni

Fungsi ini digunakan untuk mengetahui rasio dari validitas data *cluster* yang telah terbentuk. Dalam perhitungan rasio ini dilakukan perbandingan antara kepadatan dan keterpisahan jarak antar data *cluster* dan jumlah data.

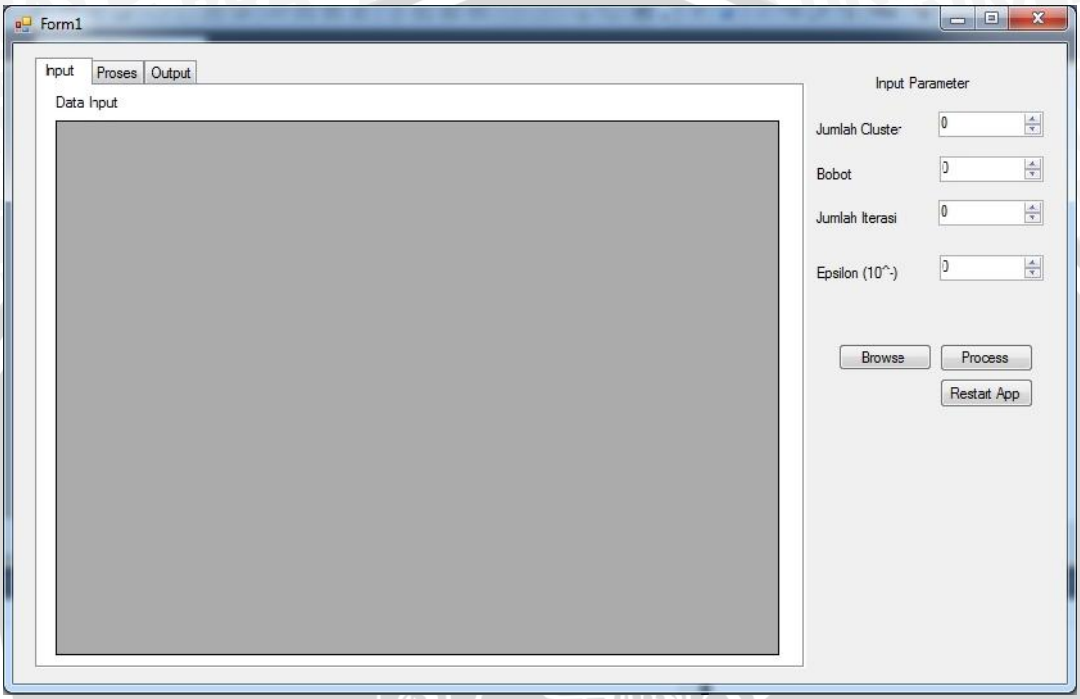
No	Source Code
1	`double XB = totalCompactness.Sum() / (rec *`
2	`Dmin.Min());`
3	`textBox1.Text = Convert.ToString(XB);`

Kode Sumber 4.11 Perhitungan Rasio Indeks Xie dan Beni

4.4 Implementasi Antar Muka

Berdasarkan perancangan antar muka pada bab 3 maka dihasilkan antar muka sebagai berikut :

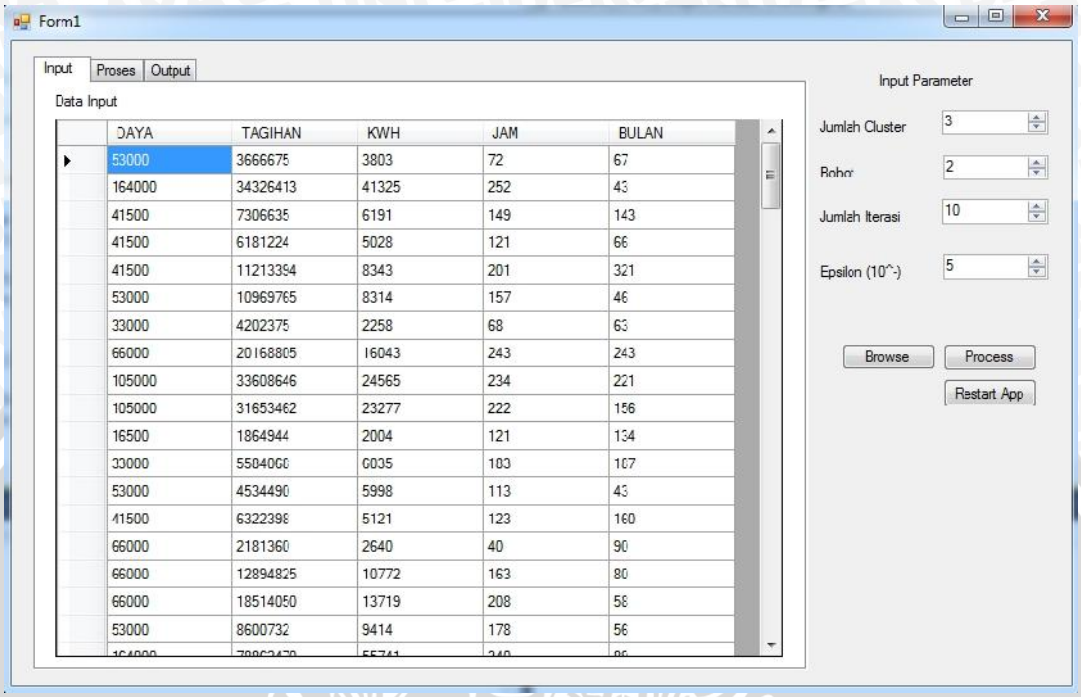
1. Antar muka *Input* Data



Gambar 4.1 Antar Muka *Input* Data

Antar muka *input* data merupakan antar muka yang dapat digunakan oleh *user* untuk memasukkan data yang akan di *cluster*. Untuk mengambil dan membaca data, *user* dapat menekan tombol *browse*. Antar muka tersebut dapat dilihat pada gambar 4.1.

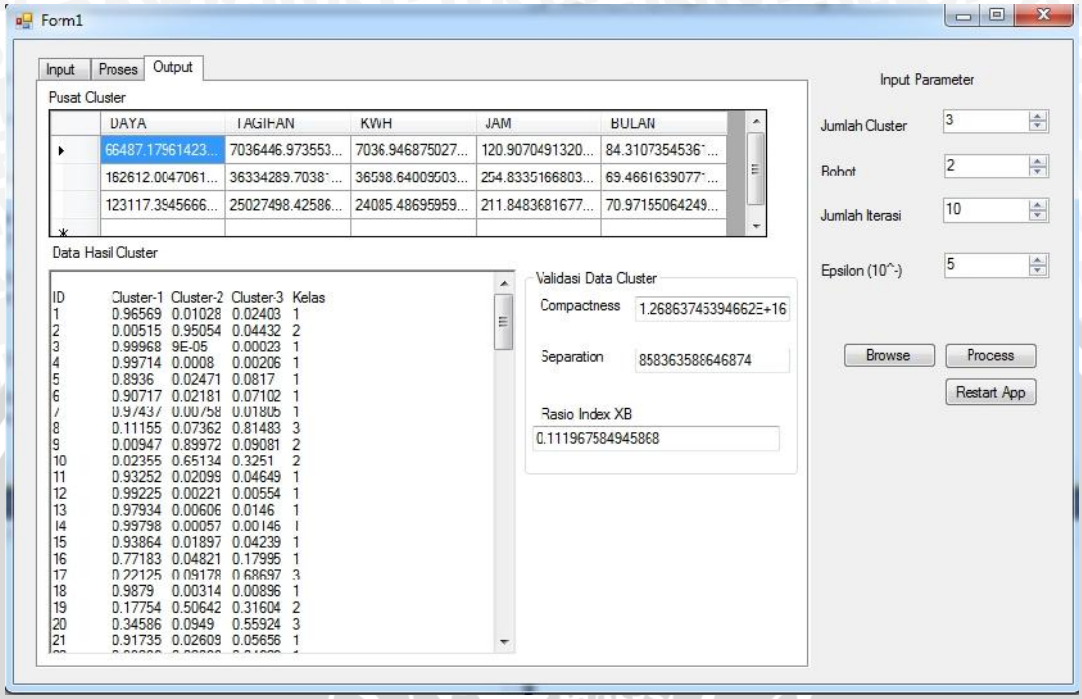
2. Antar muka *Clustering*



Gambar 4.2 Antar Muka *Clustering*

Dalam antar muka *clustering* user dapat memasukkan parameter-parameter yang akan digunakan dalam perhitungan *cluster*. Selain itu dalam antar muka ini juga dilakukan pembangkitan bilangan *random*(acak) sebagai nilai derajat keanggotaan awal. Antar muka tersebut dapat dilihat pada gambar 4.2

3. Antar muka *output* dan Rasio Validitas



Gambar 4.3 Antar Muka *Output*

Dalam antar muka *output* ditampilkan data hasil *cluster* yang berisi derajat keanggotaan akhir. Selain itu juga ditampilkan hasil akhir pusat *cluster*. Antar muka tersebut dapat dilihat pada gambar 4.3.