BAB IV
IMPLEMENTASI

4.1 Lingkungan Implementasi

Dalam implementasi metode *Fuzzy K-nearest Neighbor (FK-NN)* untuk diagnosa penderita liver dibutuhkan beberapa aspek yang perlu diperhatikan yaitu segi perangkat keras dan perangkat lunak.

4.1.1 Lingkungan Implementasi Perangkat Keras

Dalam mengembangkan sistem dan penerapan metode penelitian ini digunakan beberapa komponen perangkat keras sebagai berikut :

1. Processor : Intel ® Core™ 2Duo CPU @2.10GHz
2. Memory : 2.00 GB
3. Hard disk : 300 GB

4.1.2 Lingkungan Implementasi Perangkat Lunak

Pengembangan sistem dan penelitian ini dibutuhkan beberapa perangkat lunak yang digunakan sebagai berikut ini :

1. Sistem Operasi yang digunakan Windows 7 Ultimate 32bit
2. Aplikasi pembangunan GUI dan code menggunakan NetBeans IDE 7.3.1
3. Bahasa pemrograman yang dipakai yaitu bahasa pemrograman java.
4. Komponen java yang digunakan yaitu JDK 1.7

4.2 Implementasi Program

Berdasarkan metode penelitian dan perancangan proses yang terdapat dalam bab 3, maka pada sub bab ini akan dijelaskan implementasi proses-proses tersebut.

4.2.1 Proses Baca File

Proses baca file digunakan untuk membaca file data yang akan digunakan dalam pelatihan dan pengujian. Tahapan proses baca file ditunjukan pada source code 4.1

Proses Baca File

```
package skripsifknn;
import java.io.File;
import java.util.Arrays;
import javax.swing.JFileChooser;
import javax.swing.table.DefaultTableModel;
import jxl.Sheet;
import jxl.Workbook;
public class skripsifknngui extends
javax.swing.JFrame {

Object [][]isitabel;
Object [][]isitabeluji ;
String []namakolom=new String[11];
double[]umur;
double[]tb;
double[]db;
double[]ap;
double[]aia;
double[]apa;
double[]tp;
double[]alb;
double[]ag;
int[]target;

double[]umuruji;
double[]tbuji;
double[]dbuji;
double[]apuji;
double[]aiauji;
double[]apauji;
double[]tpuji;
double[]albuji;
double[]aguji;
int[]targetuji;
int[][]targeturut;

    double minumur;
    double maxumur;
    double mintb;
    double maxtb;
    double mindb;
    double maxdb;
    double minap;
    double maxap;
    double minaia;
    double maxaia;
    double minapa;
    double maxapa;
    double mintp;
    double maxtp;
    double minalb;
```

```
        double maxalb;
        double minag;
        double maxag;
        double mintarget;
        double maxtarget;

        double [] norumur;
        double [] nortb;
        double [] nordb;
        double [] norap;
        double [] noraia;
        double [] norapa;
        double [] nortp;
        double [] noralb;
        double [] norag;

        double [] norumuruji;
        double [] nortbuji;
        double [] nordbuji;
        double [] norapuji;
        double [] noraiauji;
        double [] norapauji;
        double [] nortpuji;
        double [] noralbuji;
        double [] noraguji;

        double total;
        double total2;

        double[]botumur;
        double[]bottb;
        double[]botdb;
        double[]botap;
        double[]botaia;
        double[]botapa;
        double[]bottp;
        double[]botalb;
        double[]botag;

        double [][] ed ;//untuk menampung ecludian
        double [][] we ;//untuk menampung nilai weight
        double [][] we1;//untuk menampung nilai weight yang
telah disortir

        double miu11;
        double miu12;
        double miu22;
        double miu21;

        int k;
        double[][] fknn1;//pembilang u1
        double[][] fknn2;//pembilang u2
```

```
        double[][] penyebut1;
        double[][] hasil2; //hasil
        double[][] hasil1; //hasil
        double[][] we2;
        int [] hasilprediksi;

        double akurasi=0;
        double totaldata=0;

    public skripsifknngui() {
        initComponents();
    }

JFileChooser pilih = new JFileChooser(); //ganti array yang
sebelumnya dengan jxl sehingga lebih ringkas dan lebih
sedikit
        pilih.showOpenDialog(null);
        File a=pilih.getSelectedFile();
        String alamat = a.getAbsolutePath();
        File hardika = new File(alamat);

        try
        {
            Workbook ilpd =
jxl.Workbook.getWorkbook(hardika);
            Sheet [] data = ilpd.getSheets();
            int jumlahbaris= data[0].getRows();
            int jumlahkolom= data[0].getColumns();

            umur= new double[jumlahbaris];
            tb= new double[jumlahbaris];
            db= new double[jumlahbaris];
            ap= new double[jumlahbaris];
            aia= new double[jumlahbaris];
            apa= new double[jumlahbaris];
            tp= new double[jumlahbaris];
            alb= new double[jumlahbaris];
            ag= new double[jumlahbaris];
            target= new int[jumlahbaris];

            for(int i=1;i<jumlahbaris;i++)
            {
               double umur1 =
Double.parseDouble(data[0].getCell(0, i).getContents());
               double tb1 =
Double.parseDouble(data[0].getCell(1, i).getContents());
               double db1 =
Double.parseDouble(data[0].getCell(2, i).getContents());
               double ap1 =
Double.parseDouble(data[0].getCell(3, i).getContents());
               double aia1 =
Double.parseDouble(data[0].getCell(4, i).getContents());
```

```
                double apa1 =
Double.parseDouble(data[0].getCell(5, i).getContents());
                double tp1 =
Double.parseDouble(data[0].getCell(6, i).getContents());
                double alb1 =
Double.parseDouble(data[0].getCell(7, i).getContents());
                double ag1 =
Double.parseDouble(data[0].getCell(8, i).getContents());
                int target1 =
Integer.parseInt(data[0].getCell(9, i).getContents());

                umur[i]=umur1;
                tb[i]=tb1;
                db[i]=db1;
                ap[i]=ap1;
                aia[i]=aia1;
                apa[i]=apa1;
                tp[i]=tp1;
                alb[i]=alb1;
                ag[i]=ag1;
                target[i]=target1;

            }

            namakolom[0]="no";
            namakolom[1]="umur";
            namakolom[2]="tb";
            namakolom[3]="db";
            namakolom[4]="ap";
            namakolom[5]="aia";
            namakolom[6]="apa";
            namakolom[7]="tp";
            namakolom[8]="alb";
            namakolom[9]="ag";
            namakolom[10]="target";

            isitabel=new Object [umur.length][11];

            for(int i=0;i<umur.length;i++)
            {
                isitabel[i][0]=i+1;
                isitabel[i][1]=umur[i];
                isitabel[i][2]=tb[i];
                isitabel[i][3]=db[i];
                isitabel[i][4]=ap[i];
                isitabel[i][5]=aia[i];
                isitabel[i][6]=apa[i];
                isitabel[i][7]=tp[i];
                isitabel[i][8]=alb[i];
                isitabel[i][9]=ag[i];
                isitabel[i][10]=target[i];
            }
```

```
        tblDataLatih.setModel(new
DefaultTableModel(isitabel,namakolom));//untuk menampilkan
tabel
```

Source Code 4.1 Baca File

4.2.2 Proses Normalisasi Data

Proses normalisasi data digunakan untuk normalisasi data latih atau data uji untuk bisa digunakan pada sistem ketika melakukan perhitungan. Source code normalisasi data ditunjukan pada source code 4.2

```
Proses Normalisasi data
        minumur=umur[1];
        maxumur=umur[1];
        mintb=tb[1];
        maxtb=tb[1];
        mindb=db[1];
        maxdb=db[1];
        minap=ap[1];
        maxap=ap[1];
        minaia=aia[1];
        maxaia=aia[1];
        minapa=apa[1];
        maxapa=apa[1];
        mintp=tp[1];
        maxtp=tp[1];
        minalb=alb[1];
        maxalb=alb[1];
        minag=ag[1];
        maxag=ag[1];
        mintarget=target[1];
       maxtarget=target[1];

   //mencari minimum dan maximum
        for(int i=1;i<umur.length;i++)
        {
            if(minumur>umur[i])
            {
                minumur=umur[i];
            }
        }
        for(int i=1;i<umur.length;i++)
        {
            if(maxumur<umur[i])
            {
                maxumur=umur[i];
            }
        }
        for(int i=1;i<umur.length;i++)
```

```
    {
        if(mintb>tb[i])
        {
            mintb=tb[i];
        }
    }
    for(int i=1;i<umur.length;i++)
    {
        if(maxtb<tb[i])
        {
            maxtb=tb[i];
        }
    }
    for(int i=1;i<umur.length;i++)
    {
        if(mindb>db[i])
        {
            mindb=db[i];
        }
    }
    for(int i=1;i<umur.length;i++)
    {
        if(maxdb<db[i])
        {
            maxdb=db[i];
        }
    }
    for(int i=1;i<umur.length;i++)
    {
        if(minap>ap[i])
        {
            minap=ap[i];
        }
    }
    for(int i=1;i<umur.length;i++)
    {
        if(maxap<ap[i])
        {
            maxap=ap[i];
        }
    }
    for(int i=1;i<umur.length;i++)
    {
        if(minaia>aia[i])
        {
            minaia=aia[i];
        }
    }
    for(int i=1;i<umur.length;i++)
    {
        if(maxaia<aia[i])
        {
```

```
            maxaia=aia[i];
        }
    }
    for(int i=1;i<umur.length;i++)
    {
        if(minapa>apa[i])
        {
            minapa=apa[i];
        }
    }
        for(int i=1;i<umur.length;i++)
        {
            if(maxapa<apa[i])
            {
                maxapa=apa[i];
            }
        }
        for(int i=1;i<umur.length;i++)
        {
            if(mintp>tp[i])
            {
                mintp=tp[i];
            }
        }
        for(int i=1;i<umur.length;i++)
        {
            if(maxtp<tp[i])
            {
                maxtp=tp[i];
            }
        }
        for(int i=1;i<umur.length;i++)
        {
            if(minalb>alb[i])
            {
                minalb=alb[i];
            }
        }
        for(int i=1;i<umur.length;i++)
        {
            if(maxalb<alb[i])
            {
                maxalb=alb[i];
            }
        }
        for(int i=1;i<umur.length;i++)
        {
            if(minag>ag[i])
            {
                minag=ag[i];
            }
        }
```

```
            for(int i=1;i<umur.length;i++)
            {
                if(maxag<ag[i])
                {
                    maxag=ag[i];
                }
            }

            //normalisasi
            norumur=new double[umur.length];
            for(int i=1;i<umur.length;i++)
            {
                norumur[i]=(umur[i]-minumur)/(maxumur-
    minumur);
            }
            nortb=new double[umur.length];
            for(int i=1;i<umur.length;i++)
            {
                nortb[i]=(tb[i]-mintb)/(maxtb-mintb);
            }
            nordb=new double[umur.length];
            for(int i=1;i<umur.length;i++)
            {
                nordb[i]=(db[i]-mindb)/(maxdb-mindb);
            }
            norap=new double[umur.length];
            for(int i=1;i<umur.length;i++)
            {
                norap[i]=(ap[i]-minap)/(maxap-minap);
            }
            noraia=new double[umur.length];
            for(int i=1;i<umur.length;i++)
            {
                noraia[i]=(aia[i]-minaia)/(maxaia-
    minaia);
            }
            norapa=new double[umur.length];
            for(int i=1;i<umur.length;i++)
            {
                norapa[i]=(apa[i]-minapa)/(maxapa-
    minapa);
            }
            nortp=new double[umur.length];
            for(int i=1;i<umur.length;i++)
            {
                nortp[i]=(tp[i]-mintp)/(maxtp-mintp);
            }
            noralb=new double[umur.length];
            for(int i=1;i<umur.length;i++)
            {
                noralb[i]=(alb[i]-minalb)/(maxalb-
    minalb);
```

```
            }
            norag=new double[umur.length];
            for(int i=1;i<umur.length;i++)
            {
                norag[i]=(ag[i]-minag)/(maxag-minag);
            }

                for(int i=0;i<umur.length;i++)
                {
                    isitabel[i][0]=i+1;
                    isitabel[i][1]=norumur[i];
                    isitabel[i][2]=nortb[i];
                    isitabel[i][3]=nordb[i];
                    isitabel[i][4]=norap[i];
                    isitabel[i][5]=noraia[i];
                    isitabel[i][6]=norapa[i];
                    isitabel[i][7]=nortp[i];
                    isitabel[i][8]=noralb[i];
                    isitabel[i][9]=norag[i];
                    isitabel[i][10]=target[i];
                }

            tblnormalisasidatalatih.setModel(new
     DefaultTableModel(isitabel,namakolom));
```

Source Code 4.2 Normalisasi Data

4.2.3 Proses Pembobotan

Proses pembobotan digunakan untuk melakukan pembobotan data latih agar mendapatkan bobot akhir yang akan digunakan untuk KNN. Source code pelatihan ditunjukan pada source code 4.3 berikut

```
Proses Pembobotan
      //pembobotan data latih di tab 1
            total=0;
            total2=0;

            botumur=new double[2];
            botumur[0]=0;
            botumur[1]=0;
            bottb=new double[2];
            bottb[0]=0;
            bottb[1]=0;
            botdb=new double[2];
            botdb[0]=0;
            botdb[1]=0;
            botap=new double[2];
            botap[0]=0;
            botap[1]=0;
            botaia=new double[2];
```

```
botaia[0]=0;
botaia[1]=0;
botapa=new double[2];
botapa[0]=0;
botapa[1]=0;
bottp=new double[2];
bottp[0]=0;
bottp[1]=0;
botalb=new double[2];
botalb[0]=0;
botalb[1]=0;
botag=new double[2];
botag[0]=0;
botag[1]=0;

for(int i=1;i<umur.length;i++)
{
    if(target[i]!=1)
    {
        botumur[1]=botumur[1]+norumur[i];
        bottb[1]=bottb[1]+nortb[i];
        botdb[1]=botdb[1]+nordb[i];
        botap[1]=botap[1]+norap[i];
        botaia[1]=botaia[1]+noraia[i];
        botapa[1]=botapa[1]+norapa[i];
        bottp[1]=bottp[1]+nortp[i];
        botalb[1]=botalb[1]+noralb[i];
        botag[1]=botag[1]+norag[i];
        total=total+1;
    }
    else
    {
        botumur[0]=botumur[0]+norumur[i];
        bottb[0]=bottb[0]+nortb[i];
        botdb[0]=botdb[0]+nordb[i];
        botap[0]=botap[0]+norap[i];
        botaia[0]=botaia[0]+noraia[i];
        botapa[0]=botapa[0]+norapa[i];
        bottp[0]=bottp[0]+nortp[i];
        botalb[0]=botalb[0]+noralb[i];
        botag[0]=botag[0]+norag[i];
        total2=total2+1;
    }
}

botumur[0]=botumur[0]/total2;
botumur[1]=botumur[1]/total;

bottb[0]=bottb[0]/total2;
bottb[1]=bottb[1]/total;

botdb[0]=botdb[0]/total2;
```

```
            botdb[1]=botdb[1]/total;

            botap[0]=botap[0]/total2;
            botap[1]=botap[1]/total;

            botaia[0]=botaia[0]/total2;
            botaia[1]=botaia[1]/total;

            botapa[0]=botapa[0]/total2;
            botapa[1]=botapa[1]/total;

            bottp[0]=bottp[0]/total2;
            bottp[1]=bottp[1]/total;

            botalb[0]=botalb[0]/total2;
            botalb[1]=botalb[1]/total;

            botag[0]=botag[0]/total2;
            botag[1]=botag[1]/total;

            }

            catch(Exception ex)
            {
                System.out.println("eror om
    "+ex.getMessage());
      }
```

Source code 4.3 Pembobotan

4.2.4 Proses KNN

Proses KNN digunakan untuk menghitung nilai jarak kedekatan tetangga data uji terhadap data latih menggunakan *Euclidean Distance*. Source code KNN ditunjukan pada source code 4.4 berikut ini.

```
Proses KNN
      JFileChooser pilih = new JFileChooser(); //ganti
      array yang sebelumnya dengan jxl sehingga lebih
      ringkas dan lebih sedikit
             pilih.showOpenDialog(null);
             File a=pilih.getSelectedFile();
             String alamat = a.getAbsolutePath();
             File hardika = new File(alamat);

             try
             {
                 Workbook ilpd =
      jxl.Workbook.getWorkbook(hardika);
                 Sheet [] data = ilpd.getSheets();
                 int jumlahbaris= data[0].getRows();
```

```
                int jumlahkolom= data[0].getColumns();

                umuruji= new double[jumlahbaris];
                tbuji= new double[jumlahbaris];
                dbuji= new double[jumlahbaris];
                apuji= new double[jumlahbaris];
                aiauji= new double[jumlahbaris];
                apauji= new double[jumlahbaris];
                tpuji= new double[jumlahbaris];
                albuji= new double[jumlahbaris];
                aguji= new double[jumlahbaris];
                targetuji= new int[jumlahbaris];

                for(int i=1;i<jumlahbaris;i++)
                {
                    double umur1 =
Double.parseDouble(data[0].getCell(0,
i).getContents());
                    double tb1 =
Double.parseDouble(data[0].getCell(1,
i).getContents());
                    double db1 =
Double.parseDouble(data[0].getCell(2,
i).getContents());
                    double ap1 =
Double.parseDouble(data[0].getCell(3,
i).getContents());
                    double aia1 =
Double.parseDouble(data[0].getCell(4,
i).getContents());
                    double apa1 =
Double.parseDouble(data[0].getCell(5,
i).getContents());
                    double tp1 =
Double.parseDouble(data[0].getCell(6,
i).getContents());
                    double alb1 =
Double.parseDouble(data[0].getCell(7,
i).getContents());
                    double ag1 =
Double.parseDouble(data[0].getCell(8,
i).getContents());
                    int target1 =
Integer.parseInt(data[0].getCell(9,
i).getContents());

                    umuruji[i]=umur1;
                    tbuji[i]=tb1;
                    dbuji[i]=db1;
                    apuji[i]=ap1;
                    aiauji[i]=aia1;
                    apauji[i]=apa1;
```

```
                tpuji[i]=tp1;
                albuji[i]=alb1;
                aguji[i]=ag1;
                targetuji[i]=target1;
            }
            isitabeluji=new
Object[umuruji.length][11];
            for(int j=0;j<umuruji.length;j++)
            {
                isitabeluji[j][0]=j+1;
                isitabeluji[j][1]=umuruji[j];
                isitabeluji[j][2]=tbuji[j];
                isitabeluji[j][3]=dbuji[j];
                isitabeluji[j][4]=apuji[j];
                isitabeluji[j][5]=aiauji[j];
                isitabeluji[j][6]=apauji[j];
                isitabeluji[j][7]=tpuji[j];
                isitabeluji[j][8]=albuji[j];
                isitabeluji[j][9]=aguji[j];
                isitabeluji[j][10]=targetuji[j];
            }

        tblDataUji.setModel(new
DefaultTableModel(isitabeluji,namakolom));//untuk
menampilkan tabel

        //menentukan minimum dan minimum data uji
        for(int j=1;j<umuruji.length;j++)
        {
            if(minumur>umuruji[j])
            {
                minumur=umuruji[j];
            }
        }

        for(int j=1;j<umuruji.length;j++)
        {
            if(maxumur<umuruji[j])
            {
                maxumur=umuruji[j];
            }
        }

        for(int j=1;j<umuruji.length;j++)
        {
            if(mintb>tbuji[j])
            {
                mintb=tbuji[j];
            }
        }

        for(int j=1;j<umuruji.length;j++)
```

```
{
    if(maxtb<tbuji[j])
    {
        maxtb=tbuji[j];
    }
}

for(int j=1;j<umuruji.length;j++)
{
    if(mindb>dbuji[j])
    {
        mindb=dbuji[j];
    }
}

for(int j=1;j<umuruji.length;j++)
{
    if(maxdb<dbuji[j])
    {
        maxdb=dbuji[j];
    }
}

for(int j=1;j<umuruji.length;j++)
{
    if(minap>apuji[j])
    {
        minap=apuji[j];
    }
}

for(int j=1;j<umuruji.length;j++)
{
    if(maxap<apuji[j])
    {
        maxap=apuji[j];
    }
}

for(int j=1;j<umuruji.length;j++)
{
    if(minaia>aiauji[j])
    {
        minaia=aiauji[j];
    }
}

for(int j=1;j<umuruji.length;j++)
{
    if(maxaia<aiauji[j])
    {
        maxaia=aiauji[j];
```

```
        }
    }

    for(int j=1;j<umuruji.length;j++)
    {
        if(minapa>apauji[j])
        {
            minapa=apauji[j];
        }
    }

    for(int j=1;j<umuruji.length;j++)
    {
        if(maxapa<apauji[j])
        {
            maxapa=apauji[j];
        }
    }

    for(int j=1;j<umuruji.length;j++)
    {
        if(mintp>tpuji[j])
        {
            mintp=tpuji[j];
        }
    }

    for(int j=1;j<umuruji.length;j++)
    {
        if(maxtp<tpuji[j])
        {
            maxtp=tpuji[j];
        }
    }

    for(int j=1;j<umuruji.length;j++)
    {
        if(minalb>albuji[j])
        {
            minalb=albuji[j];
        }
    }

    for(int j=1;j<umuruji.length;j++)
    {
        if(maxalb<albuji[j])
        {
            maxalb=albuji[j];
        }
    }

    for(int j=1;j<umuruji.length;j++)
```

```
            {
                if(minag>aguji[j])
                {
                    minag=aguji[j];
                }
            }

            for(int j=1;j<umuruji.length;j++)
            {
                if(maxag<aguji[j])
                {
                    maxag=aguji[j];
                }
            }

            //normalisasi data uji
            norumuruji=new double[umuruji.length];
            for(int i=1;i<umuruji.length;i++)
            {
                norumuruji[i]=(umuruji[i]-
minumur)/(maxumur-minumur);
            }

            nortbuji=new double[umuruji.length];
            for(int i=1;i<umuruji.length;i++)
            {
                nortbuji[i]=(tbuji[i]-mintb)/(maxtb-
mintb);
            }

            nordbuji=new double[umuruji.length];
            for(int i=1;i<umuruji.length;i++)
            {
                nordbuji[i]=(dbuji[i]-mindb)/(maxdb-
mindb);
            }

            norapuji=new double[umuruji.length];
            for(int i=1;i<umuruji.length;i++)
            {
                norapuji[i]=(apuji[i]-minap)/(maxap-
minap);
            }

            noraiauji=new double[umuruji.length];
            for(int i=1;i<umuruji.length;i++)
            {
                noraiauji[i]=(aiauji[i]-
minaia)/(maxaia-minaia);
            }

            norapauji=new double[umuruji.length];
```

```
            for(int i=1;i<umuruji.length;i++)
            {
                norapauji[i]=(apauji[i]-
minapa)/(maxapa-minapa);
            }

            nortpuji=new double[umuruji.length];
            for(int i=1;i<umuruji.length;i++)
            {
                nortpuji[i]=(tpuji[i]-mintp)/(maxtp-
mintp);
            }

            noralbuji=new double[umuruji.length];
            for(int i=1;i<umuruji.length;i++)
            {
                noralbuji[i]=(albuji[i]-
minalb)/(maxalb-minalb);
            }

            noraguji=new double[umuruji.length];
            for(int i=1;i<umuruji.length;i++)
            {
                noraguji[i]=(aguji[i]-minag)/(maxag-
minag);
            }

                for(int i=0;i<umuruji.length;i++)
                {
                    isitabeluji[i][0]=i+1;
                    isitabeluji[i][1]=norumuruji[i];
                    isitabeluji[i][2]=nortbuji[i];
                    isitabeluji[i][3]=nordbuji[i];
                    isitabeluji[i][4]=norapuji[i];
                    isitabeluji[i][5]=noraiauji[i];
                    isitabeluji[i][6]=norapauji[i];
                    isitabeluji[i][7]=nortpuji[i];
                    isitabeluji[i][8]=noralbuji[i];
                    isitabeluji[i][9]=noraguji[i];
                    isitabeluji[i][10]=targetuji[i];
                }

            tblNormalisasiDataUji.setModel(new
DefaultTableModel(isitabeluji,namakolom));

            //euclidean distance di tab 2
            ed = new double
[umuruji.length][umur.length];

            for(int j=1;j<umuruji.length;j++)
            {
                for(int i=1;i<umur.length;i++)
```

```
                {
                        if(target[i]!=1)
                        {
                        ed[j][i]=(Math.pow((norumuruji[j]-
norumur[i]), 2)*botumur[1])
                                +(Math.pow((nortbuji[j]-
nortb[i]), 2)*bottb[1])
                                +(Math.pow((nordbuji[j]-
nordb[i]), 2)*botdb[1])
                                +(Math.pow((norapuji[j]-
norap[i]), 2)*botap[1])
                                +(Math.pow((noraiauji[j]-
noraia[i]), 2)*botaia[1])
                                +(Math.pow((norapauji[j]-
norapa[i]), 2)*botapa[1])
                                +(Math.pow((nortpuji[j]-
nortp[i]), 2)*bottp[1])
                                +(Math.pow((noralbuji[j]-
noralb[i]), 2)*botalb[1])
                                +(Math.pow((noraguji[j]-
norag[i]), 2)*botag[1]);
                        }
                        else
                        {
ed[j][i]=(Math.pow((norumuruji[j]-norumur[i]),
2)*botumur[0])
                                +(Math.pow((nortbuji[j]-
nortb[i]), 2)*bottb[0])
                                +(Math.pow((nordbuji[j]-
nordb[i]), 2)*botdb[0])
                                +(Math.pow((norapuji[j]-
norap[i]), 2)*botap[0])
                                +(Math.pow((noraiauji[j]-
noraia[i]), 2)*botaia[0])
                                +(Math.pow((norapauji[j]-
norapa[i]), 2)*botapa[0])
                                +(Math.pow((nortpuji[j]-
nortp[i]), 2)*bottp[0])
                                +(Math.pow((noralbuji[j]-
noralb[i]), 2)*botalb[0])
                                +(Math.pow((noraguji[j]-
norag[i]), 2)*botag[0]);

                        }
                        ed[j][i]=Math.sqrt(ed[j][i]);

                }
        }

        for(int j=1;j<umuruji.length;j++)
        {
            for (int i=1;i<umur.length;i++)
```

```
                {
                    System.out.println("jarak euclidean
distance data ke "+j+" adalah: "+ed[j][i]);
                }
            }
            //mencari nilai weight
            we = new
double[umuruji.length][umur.length];
            we2 = new
double[umuruji.length][umur.length];
                    for(int j=1;j<umuruji.length;j++)
                    {
                        for(int i=1;i<umur.length;i++)
                        {
we[j][i]=1/(Math.pow(ed[j][i], 2));

we2[j][i]=1/(Math.pow(ed[j][i], 2));
                        }
                    }
            //mengurutkan ke nilai terbesar
            we1 = new
double[umuruji.length][umur.length];
            for(int j=1;j<umuruji.length;j++)
            {
                Arrays.sort(we[j]);//sorting array 2
dimensi
            }
            for(int j=1;j<umuruji.length;j++)
                    {
                        for(int i=1;i<umur.length;i++)
                        {

we1[j][i]=we[j][we[j].length-i];
                        }
                    }
            targeturut = new
int[umuruji.length][target.length];

            for(int k=1;k<umuruji.length; k++)
            {
                for(int j=1;j<target.length;j++)
                {
                    for(int i=1;i<target.length;i++)
                    {
                        if(we1[k][j]==we2[k][i])
                        {
targeturut[k][j]=target[i];
                        }
                    }
                }
            }
```

Source Code 4.4 KNN

4.2.5 Proses FKNN

Proses FKNN merupakan proses perhitungan nilai membership dengan 2 kelas (penderita liver dan bukan penderita liver). Source code FKNN dapat dilihat pada source code 4.5

Proses FKNN

```
        //mencari nilai membership
                miu11=0.51+((total2/(total+total2))*0.49);
                miu12=((total2/(total+total2))*0.49);
                miu22=0.51+((total/(total+total2))*0.49);
                miu21=((total/(total+total2))*0.49);
                }
                catch(Exception ex)
                {
                    System.out.println("eror om dika
     "+ex.getMessage());
                }

    //menggunakan rumus fknn
                k=Integer.parseInt(txtnilaik.getText());

                fknn1=new double[umuruji.length][2];
                //fknn2=new double[umuruji.length//][2];
                hasil1 = new double[umuruji.length][2];
                //hasil2 = new double[umuruji.length][k+1];
                penyebut1 = new double[umuruji.length][2];

                for(int j=1;j<umuruji.length;j++)
                {
                    fknn1[j][0]=0;
                    fknn1[j][1]=0;
                    hasil1[j][0]=0;
                    hasil1[j][1]=0;
                    penyebut1[j][0]=0;
                    penyebut1[j][1]=0;
                    for(int i=0;i<k;i++)
                    {
                        //fknn1[j][i]=0;
                        //fknn2[j][i]=0;
                        //hasil1[j][i]=0;
                        //hasil2[j][i]=0;
                        //penyebut1[j][i]=0;

                    }
                }
                for(int j=1;j<umuruji.length;j++)
                {
                    for(int i=1;
      i<targeturut[j].length;i++)
                    {
```

```
                }
            }

            for(int
j=1;j<umuruji.length;j++)//pembilang
            {
                for(int i=1;i<k+1;i++)
                {
                    if(targeturut[j][i]==1)
                    {

fknn1[j][0]=fknn1[j][0]+(miu11*(1/(Math.pow(we1[j][
i], 2))));

fknn1[j][1]=fknn1[j][1]+(miu21*(1/(Math.pow(we1[j][
i], 2))));
                    }
                    else
                    {

fknn1[j][0]=fknn1[j][0]+(miu12*(1/(Math.pow(we1[j][
i], 2))));

fknn1[j][1]=fknn1[j][1]+(miu22*(1/(Math.pow(we1[j][
i], 2))));
                    }
                }
            }

for(int j=1;j<umuruji.length;j++)//pembilang2
            {
                for(int i=1;i<k+1;i++)
                {

penyebut1[j][0]=fknn1[j][0]+(1/(Math.pow(we1[j][i],
2)));

penyebut1[j][1]=fknn1[j][1]+(1/(Math.pow(we1[j][i],
2)));

                }
            }

            for(int j=1;j<umuruji.length;j++)//hasile
            {

hasil1[j][0]=fknn1[j][0]/penyebut1[j][0];
hasil1[j][1]=fknn1[j][1]/penyebut1[j][1];
            }
            hasilprediksi=new int[umuruji.length];
            for(int i=1;i<umuruji.length;i++)
            {
```

```
                if(hasil1[i][0]>hasil1[i][1])
                {
                    hasilprediksi[i]=1;
                }
                else
                {
                    hasilprediksi[i]=2;
                }
            }
            for(int i=1;i<umuruji.length;i++)
            {
                if(targetuji[i]==hasilprediksi[i])
                {
                    akurasi=akurasi+1;
                            totaldata=totaldata+1;
                }
                else
                {
                    totaldata=totaldata+1;
                }
            }
            akurasi=(akurasi/totaldata)*100;
            int a = (int) akurasi;
    txtakurasi.setText(String.valueOf(a+"%"));
```

Source code 4.5 FKNN