LAMPIRAN

Lampiran 1. *Source Code* Program

PrediksiMasaStudi.java

```
package prediksimasastudi;


/**
 *
 * @author M. Syukron Azim
 */
public class PrediksiMasaStudi {
    public static void main(String[] args) {
        // TODO code application logic here
        Main n = new Main();
        n.setVisible(true);
    }
}
```

Main.java

```
package prediksimasastudi;

/**
 *
 * @author M. Syukron Azim
 */
public class Main extends javax.swing.JFrame {

    /**
     * Creates new form Main
     */
    public Main() {
        initComponents();
    }

private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {
       Pelatihan n = new Pelatihan("Select *from tabel_alumni");
       n.setVisible(true);
       this.setVisible(false);
    }

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
       Prediksi n = new Prediksi();
       n.setVisible(true);
       this.setVisible(false);
    }
```

Pelatihan.java

```
package prediksimasastudi;

import java.sql.Connection;
import java.sql.Statement;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.SQLException;
import java.sql.ResultSet;
import javax.swing.JOptionPane;
```

```
import javax.swing.JTextField;
import javax.swing.ComboBoxModel;
import javax.swing.table.DefaultTableModel;
import prediksimasastudi.NeuralNetwork;
import java.util.Scanner;
import java.io.*;
import java.text.DecimalFormat;

/**
 *
 * @author M. Syukron Azim
 */
public class Pelatihan extends javax.swing.JFrame {


   public Connection con;
   public Statement stat;
   public ResultSet res;

     Double a,b;
    int c,d,e,f;

  private static DefaultTableModel tabmode;

    /**
     * Creates new form Pelatihan
     */

    public String query;
    public Pelatihan(String query) {
        initComponents();

       this.query = query;
       tampil();

    }

    public void tampil(){
        Connection_mysql classKoneksi = new Connection_mysql();

        Object[]baris = {"ID Alumni","SKS", "IPK", "Jumlah Organisasi", "Ketua Umum", "Wakil Ketua
Umum", "Kepala Dinas", "Staff", "Lama Organisasi", "Semester Ikut Organisasi", "Target",
"Output_BPNN", "Output_Hybrid"};
        tabmode = new DefaultTableModel(null, baris);

        tabel_alumni.setModel(tabmode);

        try{
            con = classKoneksi.getConnection();
            stat = con.createStatement();
            res = stat.executeQuery(query);
                while (res.next()){
          String[] row = {res.getString(1),res.getString(2), res.getString(3), res.getString(4), res.getString(5),
res.getString(6), res.getString(7), res.getString(8), res.getString(9), res.getString(10), res.getString(11)};
                tabmode.addRow(row);
            }
            tabel_alumni.setModel(tabmode);
        }
        catch(SQLException ex){
            System.out.print(ex.getMessage());
        }
        int n = tabmode.getRowCount();
        TotData.setText(""+n);
    }
```

```
    public Pelatihan(){
      initComponents();
    }



public void SimpanParameter(){
     a = Double.parseDouble(jTxt_LearningRate.getText());
     b = Double.parseDouble(jTxt_ErrorTolerance.getText());
     c = Integer.parseInt(jTxt_Epoch.getText());

try{
          Statement statement = (Statement) Connection_mysql.getConnection().createStatement();
          statement.executeUpdate("update parameter SET Learning_Rate
='"+a+"',"+"Error_Tollerance='"+b+"',"+"Epoch='"+c+"'");
          statement.close();
          JOptionPane.showMessageDialog(null, "Parameter Berhasil disimpan");
      } catch (SQLException ex) {
          JOptionPane.showMessageDialog(null, "Parameter Gagal disimpan");
       }
    }

public void SimpanOutputPelatihan_BPNN(NeuralNetwork nn){
      Object [][] data = new Object[tabel_alumni.getRowCount()][11];
       double [][] target = new double[tabel_alumni.getRowCount()][1];

       double output[][];
       output = nn.getResultOutputs();
       int i=0;
       DefaultTableModel model = (DefaultTableModel) tabel_alumni.getModel();

       Connection_mysql classKoneksi = new Connection_mysql();
       try{
           con = classKoneksi.getConnection();
           stat = con.createStatement();
           res = stat.executeQuery("select *from tabel_alumni");
           System.out.println("Jumlah Data = "+tabel_alumni.getRowCount());
           while (res.next()){
              String[] row = {res.getString(1),res.getString(2), res.getString(3), res.getString(4), res.getString(5),
res.getString(6), res.getString(7), res.getString(8), res.getString(9), res.getString(10), res.getString(11)};
              data[i] = row;
              i++;
           }
        }
        catch(SQLException ex){
           System.out.print(ex.getMessage());
        }

       try{
          Statement statement = (Statement) Connection_mysql.getConnection().createStatement();
          statement.executeUpdate("Delete from database_final");
          for(int a = 0 ; a< output.length ; a++){
        statement.executeUpdate("insert into database_final VALUES
('"+data[a][0]+"','"+data[a][1]+"','"+data[a][2]+"','"+data[a][3]+"','"+data[a][4]+"','"+data[a][5]+"','"+data[a][6
]+"','"+data[a][7]+"','"+data[a][8]+"','"+data[a][9]+"','"+data[a][10]+"','"+output[a][0]+"',0)");
          }
          statement.close();
          JOptionPane.showMessageDialog(null, "Output Pelatihan BPNN Berhasil disimpan");
      } catch (SQLException ex) {
          System.out.println("Pesan error = "+ex);
          JOptionPane.showMessageDialog(null, "Output Pelatihan BPNN Gagal disimpan"+ex);

     }
}
```

```
public void SimpanOutputPelatihan_ESABPNN(NeuralNetwork esa_nn){
      Object [][] data = new Object[tabel_alumni.getRowCount()][13];
      double [][] data2 = new double [data.length][9];
      double [][] target = new double[tabel_alumni.getRowCount()][1];
      double output[][];
      output = esa_nn.getResultOutputs();
      int i=0;
      DefaultTableModel model = (DefaultTableModel) tabel_alumni.getModel();

      Connection_mysql classKoneksi = new Connection_mysql();
      try{
          con = classKoneksi.getConnection();
          stat = con.createStatement();
          res = stat.executeQuery("select *from database_esa");
          while (res.next()){
             String[] row = {res.getString(1),res.getString(2), res.getString(3), res.getString(4), res.getString(5),
res.getString(6), res.getString(7), res.getString(8), res.getString(9), res.getString(10), res.getString(11)};
             data[i] = row;
             i++;
          }
          tabel_alumni.setModel(model);
       }
       catch(SQLException ex){
          System.out.print(ex.getMessage());
       }

      //SIMPAN KE DATABASE ESA-BPNN ONLY
      try{
         Statement statement = (Statement) Connection_mysql.getConnection().createStatement();
         System.out.println("jumlah output = "+output.length);
         for(int a = 0 ; a< output.length ; a++){
         statement.executeUpdate("update database_final SET output_ESABPNN='"+output[a][0]+"' where
id_alumni='"+data[a][0]+"'");
         }
         statement.close();
         JOptionPane.showMessageDialog(null, "Output Pelatihan ESA-BPNN Berhasil disimpan");
      } catch (SQLException ex) {
         System.out.println("Pesan Error = "+ex);
         JOptionPane.showMessageDialog(null, "Output Pelatihan ESA-BPNN Gagal disimpan"+ex);

       }

       Object[]baris = {"ID Alumni","SKS", "IPK", "Jumlah Organisasi", "Ketua Umum", "Wakil Ketua
Umum", "Kepala Dinas", "Staff", "Lama Organisasi", "Semester Ikut Organisasi", "Target",
"Output_BPNN", "Output_Hybrid"};
      tabmode = new DefaultTableModel(null, baris);
      tabel_alumni.setModel(tabmode);

      i=0;
      try{
          con = classKoneksi.getConnection();
          stat = con.createStatement();
          res = stat.executeQuery("select *from database_final");
          while (res.next()){
             String[] row = {res.getString(1),res.getString(2), res.getString(3), res.getString(4), res.getString(5),
res.getString(6), res.getString(7), res.getString(8), res.getString(9), res.getString(10), res.getString(11),
res.getString(12), res.getString(13)};
             data[i] = row;
             tabmode.addRow(row);
             i++;
          }
          tabel_alumni.setModel(tabmode);
       }
```

```
        catch(SQLException ex){
            System.out.print(ex.getMessage());
        }
    }

public void Simpan_max_min (double [] min, double [] max){
        int i=0;

            try{
           Statement statement = (Statement) Connection_mysql.getConnection().createStatement();
           statement.executeUpdate("Delete from minmax");
           for(int a = 0 ; a < 9 ; a++){
           statement.executeUpdate("insert into minmax VALUES ('"+min[a]+"','"+max[a]+"')");
                   }
           statement.close();
           JOptionPane.showMessageDialog(null, "Data Min & Max Berhasil disimpan");
       } catch (SQLException ex) {
           JOptionPane.showMessageDialog(null, "Data Min & Max Gagal disimpan");
        }
    }

private void jPelatihanActionPerformed(java.awt.event.ActionEvent evt) {
       int i =0;
       Object [][] data_BPNN = new Object[tabel_alumni.getRowCount()][9];
       Object [][] data_ESABPNN = new Object[(int)Math.round(tabel_alumni.getRowCount()*0.4)][9];
       double [][] target = new double[tabel_alumni.getRowCount()][1];
       a = Double.parseDouble(jTxt_LearningRate.getText());
        System.out.println("data a= "+a);
       b = Double.parseDouble(jTxt_ErrorTolerance.getText());
        System.out.println("data b= "+b);
       c = Integer.parseInt(jTxt_Epoch.getText());
        System.out.println("data c= "+c);

//=============================== BPNN ONLY ================================
       DefaultTableModel model = (DefaultTableModel) tabel_alumni.getModel();

       Connection_mysql classKoneksi = new Connection_mysql();
       try{
           con = classKoneksi.getConnection();
           stat = con.createStatement();
           res = stat.executeQuery("select *from normalisasi");
           while (res.next()){
               String[] row = {res.getString(2), res.getString(3), res.getString(4), res.getString(5), res.getString(6),
res.getString(7), res.getString(8), res.getString(9), res.getString(10)};
               data_BPNN[i] = row;
               target[i][0] = (Double.parseDouble(res.getString(11))) ;
               i++;
           }
        }
        catch(SQLException ex){
            System.out.print(ex.getMessage());
        }

       NeuralNetwork nn = new NeuralNetwork(9,3,1, data_BPNN, target);
       nn.setLearningRate(a);
       nn.setErrorTollerance(b);
       nn.setEpoch(c);
       nn.run_Pelatihan_BPNN(c,b);

       SimpanOutputPelatihan_BPNN(nn);

       jEpoch2.setText(String.valueOf(nn.getW_epoch()));
       jMSE2.setText(String.valueOf(nn.getW_error()));
       jAccuracy2.setText(String.valueOf(nn.getW_accuracy()+"%"));
```

```
//================================ ESA - BPNN ==============================
      System.out.println("\n ELEPHANT SEARCH ALGORITHM \n");

      ElephantSearch_Aji ESA = new ElephantSearch_Aji();
      ESA.setTabel_alumni(tabel_alumni);
      ESA.Apply((int)Math.round(tabel_alumni.getRowCount()*0.6),
(int)Math.round(tabel_alumni.getRowCount()*0.4), 100);
      i=0;
       try{
          con = classKoneksi.getConnection();
          stat = con.createStatement();
          res = stat.executeQuery("select *from database_esa");
          while (res.next()){
             String[] row = {res.getString(2), res.getString(3), res.getString(4), res.getString(5), res.getString(6),
res.getString(7), res.getString(8), res.getString(9), res.getString(10)};
             data_ESABPNN[i] = row;
             target[i][0] = (Double.parseDouble(res.getString(11))) ;
             i++;
          }
       }
       catch(SQLException ex){
          System.out.print(ex.getMessage());
       }
      NeuralNetwork esa_nn = new NeuralNetwork(9,3,1, data_ESABPNN, target);
      esa_nn.setLearningRate(a);
      esa_nn.setErrorTollerance(b);
      esa_nn.setEpoch(c);
      esa_nn.run_Pelatihan_ESABPNN(c,b);

      SimpanOutputPelatihan_ESABPNN(esa_nn);

      jEpoch.setText(String.valueOf(esa_nn.getW_epoch()));
      jMSE.setText(String.valueOf(esa_nn.getW_error()));
      jAccuracy.setText(String.valueOf(esa_nn.getW_accuracy()+"%"));

      SimpanParameter();
   }

   private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {
      Main n = new Main();
      n.setVisible(true);
      this.setVisible(false);
   }

   private void SuntingDataActionPerformed(java.awt.event.ActionEvent evt) {
      EditData n = new EditData(tabel_alumni);
      n.setVisible(true);
      this.setVisible(false);
   }

private void jBtn_NormalisasiActionPerformed(java.awt.event.ActionEvent evt) {
       DecimalFormat df = new DecimalFormat ("#.##");
       double [] sks = new double[tabel_alumni.getRowCount()];
       double [] ipk = new double[tabel_alumni.getRowCount()];
       double [] jml_org = new double[tabel_alumni.getRowCount()];
       double [] ketum = new double [tabel_alumni.getRowCount()];
       double [] waketum = new double[tabel_alumni.getRowCount()];
       double [] kadin = new double[tabel_alumni.getRowCount()];
       double [] staff = new double[tabel_alumni.getRowCount()];
       double [] lama_org = new double[tabel_alumni.getRowCount()];
       double [] smstr_org = new double[tabel_alumni.getRowCount()];

       double minSKS = Double.parseDouble(tabel_alumni.getValueAt(0, 1).toString()) ;
       double maxSKS= Double.parseDouble(tabel_alumni.getValueAt(0, 1).toString()) ;
```

```
double minIPK= Double.parseDouble(tabel_alumni.getValueAt(0, 2).toString()) ;
double maxIPK= Double.parseDouble(tabel_alumni.getValueAt(0, 2).toString()) ;
double min_jml_org= Double.parseDouble(tabel_alumni.getValueAt(0, 3).toString()) ;
double max_jml_org= Double.parseDouble(tabel_alumni.getValueAt(0, 3).toString()) ;
double min_ketum = Double.parseDouble(tabel_alumni.getValueAt(0, 4).toString());
double max_ketum = Double.parseDouble(tabel_alumni.getValueAt(0, 4).toString());
double min_waketum = Double.parseDouble(tabel_alumni.getValueAt(0, 5).toString());
double max_waketum = Double.parseDouble(tabel_alumni.getValueAt(0, 5).toString());
double min_kadin = Double.parseDouble(tabel_alumni.getValueAt(0, 6).toString());
double max_kadin = Double.parseDouble(tabel_alumni.getValueAt(0, 6).toString());
double min_staff = Double.parseDouble(tabel_alumni.getValueAt(0, 7).toString());
double max_staff = Double.parseDouble(tabel_alumni.getValueAt(0, 7).toString());
double min_lama_org= Double.parseDouble(tabel_alumni.getValueAt(0, 8).toString()) ;
double max_lama_org= Double.parseDouble(tabel_alumni.getValueAt(0, 8).toString()) ;
double min_smstr_org= Double.parseDouble(tabel_alumni.getValueAt(0, 9).toString()) ;
double max_smstr_org= Double.parseDouble(tabel_alumni.getValueAt(0, 9).toString()) ;

for(int j =0; j<tabel_alumni.getRowCount(); j++){
   if(minSKS > Double.parseDouble(tabel_alumni.getValueAt(j, 1).toString()) ){
      minSKS = Double.parseDouble(tabel_alumni.getValueAt(j, 1).toString());
   }
   if(maxSKS < Double.parseDouble(tabel_alumni.getValueAt(j, 1).toString())){
      maxSKS = Double.parseDouble(tabel_alumni.getValueAt(j, 1).toString());
   }
   if(minIPK > Double.parseDouble(tabel_alumni.getValueAt(j, 2).toString())){
      minIPK = Double.parseDouble(tabel_alumni.getValueAt(j, 2).toString());
   }
   if(maxIPK < Double.parseDouble(tabel_alumni.getValueAt(j, 2).toString())){
      maxIPK = Double.parseDouble(tabel_alumni.getValueAt(j, 2).toString());
   }
   if(min_jml_org > Double.parseDouble(tabel_alumni.getValueAt(j, 3).toString())){
      min_jml_org = Double.parseDouble(tabel_alumni.getValueAt(j, 3).toString());
   }
   if(max_jml_org < Double.parseDouble(tabel_alumni.getValueAt(j, 3).toString())){
      max_jml_org = Double.parseDouble(tabel_alumni.getValueAt(j, 3).toString());
   }
   if(min_ketum > Double.parseDouble(tabel_alumni.getValueAt(j, 4).toString())){
      min_ketum = Double.parseDouble(tabel_alumni.getValueAt(j, 4).toString());
   }
   if(max_ketum < Double.parseDouble(tabel_alumni.getValueAt(j, 4).toString())){
      max_ketum = Double.parseDouble(tabel_alumni.getValueAt(j, 4).toString());
   }
   if(min_waketum > Double.parseDouble(tabel_alumni.getValueAt(j, 5).toString())){
      min_waketum = Double.parseDouble(tabel_alumni.getValueAt(j, 5).toString());
   }
   if(max_waketum < Double.parseDouble(tabel_alumni.getValueAt(j, 5).toString())){
      max_waketum = Double.parseDouble(tabel_alumni.getValueAt(j, 5).toString());
   }
   if(min_kadin > Double.parseDouble(tabel_alumni.getValueAt(j, 6).toString())){
      min_kadin = Double.parseDouble(tabel_alumni.getValueAt(j, 6).toString());
   }
   if(max_kadin < Double.parseDouble(tabel_alumni.getValueAt(j, 6).toString())){
      max_kadin = Double.parseDouble(tabel_alumni.getValueAt(j, 6).toString());
   }
   if(min_staff > Double.parseDouble(tabel_alumni.getValueAt(j, 7).toString())){
      min_staff = Double.parseDouble(tabel_alumni.getValueAt(j, 7).toString());
   }
   if(max_staff < Double.parseDouble(tabel_alumni.getValueAt(j, 7).toString())){
      max_staff = Double.parseDouble(tabel_alumni.getValueAt(j, 7).toString());
   }
   if(min_lama_org > Double.parseDouble(tabel_alumni.getValueAt(j, 8).toString())){
      min_lama_org = Double.parseDouble(tabel_alumni.getValueAt(j, 8).toString());
   }
   if(max_lama_org < Double.parseDouble(tabel_alumni.getValueAt(j, 8).toString())){
```

```
            max_lama_org = Double.parseDouble(tabel_alumni.getValueAt(j, 8).toString());
        }
        if(min_smstr_org > Double.parseDouble(tabel_alumni.getValueAt(j, 9).toString())){
            min_smstr_org = Double.parseDouble(tabel_alumni.getValueAt(j, 9).toString());
        }
        if(max_smstr_org < Double.parseDouble(tabel_alumni.getValueAt(j, 9).toString())){
            max_smstr_org = Double.parseDouble(tabel_alumni.getValueAt(j, 9).toString());
        }
    }
    double tmpMin[] = {minSKS, minIPK, min_jml_org, min_ketum, min_waketum, min_kadin, min_staff,
min_lama_org, min_smstr_org};
    double tmpMax[] = {maxSKS, maxIPK, max_jml_org, max_ketum, max_waketum, max_kadin,
max_staff, max_lama_org, max_smstr_org};

    Simpan_max_min(tmpMin, tmpMax);

    System.out.println(minSKS + "," + maxSKS);
    System.out.println(minIPK + "," + maxIPK);
    System.out.println(min_jml_org + "," + max_jml_org);
    System.out.println(min_ketum + "," + max_ketum);
    System.out.println(min_waketum + "," + max_waketum);
    System.out.println(min_kadin + "," + max_kadin);
    System.out.println(min_staff + "," + max_staff);
    System.out.println(min_lama_org + "," + max_lama_org);
    System.out.println(min_smstr_org + "," + max_smstr_org);

    System.out.println("");
    System.out.println("=====================================================");
    System.out.println("");

    for(int i =0; i<tabel_alumni.getRowCount(); i++){
        sks[i] = ((((Double.parseDouble(tabel_alumni.getValueAt(i, 1).toString()) - minSKS)/(maxSKS -
minSKS))*(0.9 - 0.1)) + 0.1);
        ipk[i] = ((((Double.parseDouble(tabel_alumni.getValueAt(i, 2).toString()) - minIPK)/(maxIPK -
minIPK))*(0.9 - 0.1)) + 0.1);
        jml_org[i] = ((((Double.parseDouble(tabel_alumni.getValueAt(i, 3).toString()) -
min_jml_org)/(max_jml_org - min_jml_org))*(0.9 - 0.1)) + 0.1);
        ketum[i] = ((((Double.parseDouble(tabel_alumni.getValueAt(i, 4).toString()) -
min_ketum)/(max_ketum - min_ketum))*(0.9 - 0.1)) + 0.1);
        waketum[i] = ((((Double.parseDouble(tabel_alumni.getValueAt(i, 5).toString()) -
min_waketum)/(max_waketum - min_waketum))*(0.9 - 0.1)) + 0.1);
        kadin[i] = ((((Double.parseDouble(tabel_alumni.getValueAt(i, 6).toString()) -
min_kadin)/(max_kadin - min_kadin))*(0.9 - 0.1)) + 0.1);
        staff[i] = ((((Double.parseDouble(tabel_alumni.getValueAt(i, 7).toString()) - min_staff)/(max_staff -
min_staff))*(0.9 - 0.1)) + 0.1);
        lama_org[i] = ((((Double.parseDouble(tabel_alumni.getValueAt(i, 8).toString()) -
min_lama_org)/(max_lama_org - min_lama_org))*(0.9 - 0.1)) + 0.1);
        smstr_org[i] = ((((Double.parseDouble(tabel_alumni.getValueAt(i, 9).toString()) -
min_smstr_org)/(max_smstr_org - min_smstr_org))*(0.9 - 0.1)) + 0.1);

        System.out.println(i+1 + "," + df.format(sks[i]) + "," + df.format(ipk[i])+ "," + df.format(jml_org[i]) +
"," + df.format(ketum[i]) + "," + df.format(waketum[i]) + "," + df.format(kadin[i]) + "," + df.format(staff[i])
+ "," + df.format(lama_org[i])+"," + df.format(smstr_org[i]));
    }

     try{
       Statement statement = (Statement) Connection_mysql.getConnection().createStatement();
       statement.executeUpdate("Delete from normalisasi");
       for(int a = 0 ; a< tabel_alumni.getRowCount() ; a++){
       statement.executeUpdate("insert into normalisasi VALUES ('"+tabel_alumni.getValueAt(a,
0).toString()+"','"+sks[a]+"','"+ ipk[a]
+"','"+jml_org[a]+"','"+ketum[a]+"','"+waketum[a]+"','"+kadin[a]+"','"+staff[a]+"','"+lama_org[a]+"','"+smstr
_org[a]+"','"+ Integer.parseInt(tabel_alumni.getValueAt(a, 10).toString())+"')");
       }
```

```
        statement.close();
        JOptionPane.showMessageDialog(null, "Data Normalisasi Berhasil disimpan");
      } catch (SQLException ex) {
        JOptionPane.showMessageDialog(null, "Data Normalisasi Gagal disimpan"+ex);
      }

      Object[]baris = {"ID Alumni","SKS", "IPK", "Jumlah Organisasi", "Ketua Umum", "Wakil Ketua Umum", "Kepala Dinas", "Staff", "Lama Organisasi", "Semester Ikut Organisasi", "Target", "Output_BPNN", "Output_Hybrid"};
      tabmode = new DefaultTableModel(null, baris);

      tabel_alumni.setModel(tabmode);

      try{
        Statement statement = (Statement) Connection_mysql.getConnection().createStatement();
        res = statement.executeQuery("select *from normalisasi");
        while (res.next()){

         String[] row = {res.getString(1),   df.format(Double.parseDouble(res.getString(2))),
                                    df.format(Double.parseDouble(res.getString(3))),
                                    df.format(Double.parseDouble(res.getString(4))),
                                    df.format(Double.parseDouble(res.getString(5))),
                                    df.format(Double.parseDouble(res.getString(6))),
                                    df.format(Double.parseDouble(res.getString(7))),
                                    df.format(Double.parseDouble(res.getString(8))),
                                    df.format(Double.parseDouble(res.getString(9))),
                                    df.format(Double.parseDouble(res.getString(10))), res.getString(11)};
          tabmode.addRow(row);
        }
        tabel_alumni.setModel(tabmode);
      }
      catch(SQLException ex){
        System.out.print(ex.getMessage());
      }
    }
```

Connection_mysql.java

```
package prediksimasastudi;

import java.sql.Connection;
import java.sql.DriverManager;
import javax.swing.JOptionPane;
/**
 *
 * @author M. Syukron Azim
 */
public class Connection_mysql {
    private static Connection koneksi;
    public static Connection getConnection(){
        try{
            Class.forName("com.mysql.jdbc.Driver");
        } catch(ClassNotFoundException ex){
        }

        try{
            koneksi = DriverManager.getConnection("jdbc:mysql://localhost/mahasiswa_alumni","root","");
        } catch (Exception e){
        } return koneksi;
    }
}
```

EditData.java

```
package prediksimasastudi;

import java.sql.Statement;
import java.sql.SQLException;
import java.sql.ResultSet;
import javax.swing.JOptionPane;
import javax.swing.JTable;

/**
 *
 * @author M. Syukron Azim
 */
public class EditData extends javax.swing.JFrame {

    /**
     * Creates new form EditData
     */

    public Statement statement;
    public ResultSet res;
    public JTable tabel_alumni;
    public String query;

    public EditData(JTable tabel_alumni) {
        initComponents();

        this.tabel_alumni = tabel_alumni;
    }

    public EditData(){
        throw new UnsupportedOperationException("Not supported yet.");
    }

    private void jBtn_KembaliActionPerformed(java.awt.event.ActionEvent evt) {
       Pelatihan n = new Pelatihan("Select *from tabel_alumni");
       n.setVisible(true);
       this.setVisible(false);
    }

    private void jBtn_NewActionPerformed(java.awt.event.ActionEvent evt) {
       jTxt_IdAlumni.setText("");
       jTxt_SKS.setText("");
       jTxt_IPK.setText("");
       jTxt_jumlahOrganisasi.setText("");
       jCmb_Ketum.setSelectedItem("Tidak Pernah");
       jCmb_Waketum.setSelectedItem("Tidak Pernah");
       jCmb_Kadin.setSelectedItem("Tidak Pernah");
       jCmb_Staff.setSelectedItem("Tidak Pernah");
       jCmb_lamaOrganisasi.setSelectedItem("1 Tahun");
       jCmb_semesterBergabungOrganisasi.setSelectedItem("Semester 1");
       jCmb_Durasi.setSelectedItem("< 4 Tahun");
       jTxt_IdAlumni.requestFocus();
    }

    private void jBtn_SimpanActionPerformed(java.awt.event.ActionEvent evt) {
       int nilai1, nilai2, nilai3, nilai_Ketum, nilai_Waketum, nilai_Kadin, nilai_Staff;

       String id_alumni = jTxt_IdAlumni.getText();
       String SKS = jTxt_SKS.getText();
       String IPK = jTxt_IPK.getText();
       String jumlahOrganisasi = jTxt_jumlahOrganisasi.getText();
```

```
String jabatanAsKetum = ((String)jCmb_Ketum.getSelectedItem());
if(jabatanAsKetum == "Tidak Pernah"){
    nilai_Ketum = 0;
}  else if(jabatanAsKetum == "1 Kali"){
    nilai_Ketum = 1;
} else if(jabatanAsKetum == "2 Kali"){
    nilai_Ketum = 2;
} else if(jabatanAsKetum == "3 Kali"){
    nilai_Ketum = 3;
} else if(jabatanAsKetum == "4 Kali"){
    nilai_Ketum = 4;
} else if(jabatanAsKetum == "5 Kali"){
    nilai_Ketum = 5;
} else if(jabatanAsKetum == "6 Kali"){
    nilai_Ketum = 6;
} else if(jabatanAsKetum == "7 Kali"){
    nilai_Ketum = 7;
} else if(jabatanAsKetum == "8 Kali"){
    nilai_Ketum = 8;
} else if(jabatanAsKetum == "9 Kali"){
    nilai_Ketum = 9;
} else if(jabatanAsKetum == "10 Kali"){
    nilai_Ketum = 10;
} else {
    nilai_Ketum = 12;
}
String jabatanAsWaketum = ((String)jCmb_Waketum.getSelectedItem());
 if(jabatanAsWaketum == "Tidak Pernah"){
    nilai_Waketum = 0;
}  else if(jabatanAsWaketum == "1 Kali"){
    nilai_Waketum = 1;
} else if(jabatanAsWaketum == "2 Kali"){
    nilai_Waketum = 2;
} else if(jabatanAsWaketum == "3 Kali"){
    nilai_Waketum = 3;
} else if(jabatanAsWaketum == "4 Kali"){
    nilai_Waketum = 4;
} else if(jabatanAsWaketum == "5 Kali"){
    nilai_Waketum = 5;
} else if(jabatanAsWaketum == "6 Kali"){
    nilai_Waketum = 6;
} else if(jabatanAsWaketum == "7 Kali"){
    nilai_Waketum = 7;
} else if(jabatanAsWaketum == "8 Kali"){
    nilai_Waketum = 8;
} else if(jabatanAsWaketum == "9 Kali"){
    nilai_Waketum = 9;
} else if(jabatanAsWaketum == "10 Kali"){
    nilai_Waketum = 10;
} else {
    nilai_Waketum = 12;
}
String jabatanAsKadin = ((String)jCmb_Kadin.getSelectedItem());
 if(jabatanAsKadin == "Tidak Pernah"){
    nilai_Kadin = 0;
}  else if(jabatanAsKadin == "1 Kali"){
    nilai_Kadin = 1;
} else if(jabatanAsKadin == "2 Kali"){
    nilai_Kadin = 2;
} else if(jabatanAsKadin == "3 Kali"){
    nilai_Kadin = 3;
} else if(jabatanAsKadin == "4 Kali"){
    nilai_Kadin = 4;
} else if(jabatanAsKadin == "5 Kali"){
```

```
    nilai_Kadin = 5;
} else if(jabatanAsKadin == "6 Kali"){
    nilai_Kadin = 6;
} else if(jabatanAsKadin == "7 Kali"){
    nilai_Kadin = 7;
} else if(jabatanAsKadin == "8 Kali"){
    nilai_Kadin = 8;
} else if(jabatanAsKadin == "9 Kali"){
    nilai_Kadin = 9;
} else if(jabatanAsKadin == "10 Kali"){
    nilai_Kadin = 10;
} else {
    nilai_Kadin = 12;
}
String jabatanAsStaff = ((String)jCmb_Staff.getSelectedItem());
 if(jabatanAsStaff == "Tidak Pernah"){
    nilai_Staff = 0;
}  else if(jabatanAsStaff == "1 Kali"){
    nilai_Staff = 1;
} else if(jabatanAsStaff == "2 Kali"){
    nilai_Staff = 2;
} else if(jabatanAsStaff == "3 Kali"){
    nilai_Staff = 3;
} else if(jabatanAsStaff == "4 Kali"){
    nilai_Staff = 4;
} else if(jabatanAsStaff == "5 Kali"){
    nilai_Staff = 5;
} else if(jabatanAsStaff == "6 Kali"){
    nilai_Staff = 6;
} else if(jabatanAsStaff == "7 Kali"){
    nilai_Staff = 7;
} else if(jabatanAsStaff == "8 Kali"){
    nilai_Staff = 8;
} else if(jabatanAsStaff == "9 Kali"){
    nilai_Staff = 9;
} else if(jabatanAsStaff == "10 Kali"){
    nilai_Staff = 10;
} else {
    nilai_Staff = 12;
}

String lamaOrganisasi = ((String) jCmb_lamaOrganisasi.getSelectedItem());
if(lamaOrganisasi == "1 Tahun"){
    nilai3 = 1;
} else if(lamaOrganisasi == "2 Tahun"){
    nilai3 = 2;
} else if(lamaOrganisasi == "3 Tahun"){
    nilai3 = 3;
} else if(lamaOrganisasi == "4 Tahun"){
    nilai3 = 4;
} else {
    nilai3 = 5;
}
String semesterBergabungOrganisasi = ((String)jCmb_semesterBergabungOrganisasi.getSelectedItem());
if(semesterBergabungOrganisasi == "Semester 1"){
    nilai1 = 1;
} else if(semesterBergabungOrganisasi == "Semester 2"){
    nilai1 = 2;
} else if(semesterBergabungOrganisasi == "Semester 3"){
    nilai1 = 3;
} else if(semesterBergabungOrganisasi == "Semester 4"){
    nilai1 = 4;
}else if(semesterBergabungOrganisasi == "Semester 5"){
    nilai1 = 5;
```

```
        }else if(semesterBergabungOrganisasi == "Semester 6"){
            nilai1 = 6;
        }else if(semesterBergabungOrganisasi == "Semester 7"){
            nilai1 = 7;
        }else if(semesterBergabungOrganisasi == "Semester 8"){
            nilai1 = 8;
        } else {
            nilai1 = 10;
        }
        String Target = ((String) jCmb_Durasi.getSelectedItem());
        if(Target == "< 4 Tahun"){
            nilai2 = 1;
        } else {
           nilai2 = 0;
        }

        Pelatihan n = new Pelatihan("Select *from tabel_alumni");
        n.setVisible(true);
        this.setVisible(false);
        try{
            Statement statement = (Statement) Connection_mysql.getConnection().createStatement();
            statement.executeUpdate("insert into tabel_alumni VALUES ('"+id_alumni+"','"+SKS+"','"+ IPK
+"','"+jumlahOrganisasi+"','"+nilai_Ketum+"','"+nilai_Waketum+"','"+nilai_Kadin+"','"+nilai_Staff+"','"+nila
i3+"','"+nilai1+"','"+nilai2+"')");
            statement.close();
            JOptionPane.showMessageDialog(null, "Data Berhasil disimpan");
        } catch (SQLException ex) {
            JOptionPane.showMessageDialog(null, "Data Gagal disimpan");
         }
       n.tampil();
    }

    private void jBtn_EditActionPerformed(java.awt.event.ActionEvent evt) {
        int nilai1, nilai2, nilai3, nilai_Ketum, nilai_Waketum, nilai_Kadin, nilai_Staff, nilai_Jabatan;
         if (jBtn_Edit.getText() == "jBtn_Edit"){
            jTxt_IdAlumni.requestFocus();
        } else{
        String id_alumni = jTxt_IdAlumni.getText();
        String SKS = jTxt_SKS.getText();
        String IPK = jTxt_IPK.getText();
        String jumlahOrganisasi = jTxt_jumlahOrganisasi.getText();

        String jabatanAsKetum = ((String)jCmb_Ketum.getSelectedItem());
        if(jabatanAsKetum == "Tidak Pernah"){
            nilai_Ketum = 0;
        }  else if(jabatanAsKetum == "1 Kali"){
            nilai_Ketum = 1;
        } else if(jabatanAsKetum == "2 Kali"){
            nilai_Ketum = 2;
        } else if(jabatanAsKetum == "3 Kali"){
            nilai_Ketum = 3;
        } else if(jabatanAsKetum == "4 Kali"){
            nilai_Ketum = 4;
        } else if(jabatanAsKetum == "5 Kali"){
            nilai_Ketum = 5;
        } else if(jabatanAsKetum == "6 Kali"){
            nilai_Ketum = 6;
        } else if(jabatanAsKetum == "7 Kali"){
            nilai_Ketum = 7;
        } else if(jabatanAsKetum == "8 Kali"){
            nilai_Ketum = 8;
        } else if(jabatanAsKetum == "9 Kali"){
            nilai_Ketum = 9;
        } else if(jabatanAsKetum == "10 Kali"){
```

```
    nilai_Ketum = 10;
} else {
    nilai_Ketum = 12;
}
String jabatanAsWaketum = ((String)jCmb_Waketum.getSelectedItem());
 if(jabatanAsWaketum == "Tidak Pernah"){
    nilai_Waketum = 0;
}  else if(jabatanAsWaketum == "1 Kali"){
    nilai_Waketum = 1;
} else if(jabatanAsWaketum == "2 Kali"){
    nilai_Waketum = 2;
} else if(jabatanAsWaketum == "3 Kali"){
    nilai_Waketum = 3;
} else if(jabatanAsWaketum == "4 Kali"){
    nilai_Waketum = 4;
} else if(jabatanAsWaketum == "5 Kali"){
    nilai_Waketum = 5;
} else if(jabatanAsWaketum == "6 Kali"){
    nilai_Waketum = 6;
} else if(jabatanAsWaketum == "7 Kali"){
    nilai_Waketum = 7;
} else if(jabatanAsWaketum == "8 Kali"){
    nilai_Waketum = 8;
} else if(jabatanAsWaketum == "9 Kali"){
    nilai_Waketum = 9;
} else if(jabatanAsWaketum == "10 Kali"){
    nilai_Waketum = 10;
} else {
    nilai_Waketum = 12;
}
String jabatanAsKadin = ((String)jCmb_Kadin.getSelectedItem());
 if(jabatanAsKadin == "Tidak Pernah"){
    nilai_Kadin = 0;
}  else if(jabatanAsKadin == "1 Kali"){
    nilai_Kadin = 1;
} else if(jabatanAsKadin == "2 Kali"){
    nilai_Kadin = 2;
} else if(jabatanAsKadin == "3 Kali"){
    nilai_Kadin = 3;
} else if(jabatanAsKadin == "4 Kali"){
    nilai_Kadin = 4;
} else if(jabatanAsKadin == "5 Kali"){
    nilai_Kadin = 5;
} else if(jabatanAsKadin == "6 Kali"){
    nilai_Kadin = 6;
} else if(jabatanAsKadin == "7 Kali"){
    nilai_Kadin = 7;
} else if(jabatanAsKadin == "8 Kali"){
    nilai_Kadin = 8;
} else if(jabatanAsKadin == "9 Kali"){
    nilai_Kadin = 9;
} else if(jabatanAsKadin == "10 Kali"){
    nilai_Kadin = 10;
} else {
    nilai_Kadin = 12;
}
String jabatanAsStaff = ((String)jCmb_Staff.getSelectedItem());
 if(jabatanAsStaff == "Tidak Pernah"){
    nilai_Staff = 0;
}  else if(jabatanAsStaff == "1 Kali"){
    nilai_Staff = 1;
} else if(jabatanAsStaff == "2 Kali"){
    nilai_Staff = 2;
} else if(jabatanAsStaff == "3 Kali"){
```

```
        nilai_Staff = 3;
    } else if(jabatanAsStaff == "4 Kali"){
        nilai_Staff = 4;
    } else if(jabatanAsStaff == "5 Kali"){
        nilai_Staff = 5;
    } else if(jabatanAsStaff == "6 Kali"){
        nilai_Staff = 6;
    } else if(jabatanAsStaff == "7 Kali"){
        nilai_Staff = 7;
    } else if(jabatanAsStaff == "8 Kali"){
        nilai_Staff = 8;
    } else if(jabatanAsStaff == "9 Kali"){
        nilai_Staff = 9;
    } else if(jabatanAsStaff == "10 Kali"){
        nilai_Staff = 10;
    } else {
        nilai_Staff = 12;
    }

    nilai_Jabatan = (nilai_Ketum + nilai_Waketum + nilai_Kadin + nilai_Staff);

    String lamaOrganisasi = ((String) jCmb_lamaOrganisasi.getSelectedItem());
    if(lamaOrganisasi == "1 Tahun"){
        nilai3 = 1;
    } else if(lamaOrganisasi == "2 Tahun"){
        nilai3 = 2;
    } else if(lamaOrganisasi == "3 Tahun"){
        nilai3 = 3;
    } else if(lamaOrganisasi == "4 Tahun"){
        nilai3 = 4;
    } else {
        nilai3 = 5;
    }
    String semesterBergabungOrganisasi = ((String)
jCmb_semesterBergabungOrganisasi.getSelectedItem());
    if(semesterBergabungOrganisasi == "Semester 1"){
        nilai1 = 1;
    } else if(semesterBergabungOrganisasi == "Semester 2"){
        nilai1 = 2;
    } else if(semesterBergabungOrganisasi == "Semester 3"){
        nilai1 = 3;
    } else if(semesterBergabungOrganisasi == "Semester 4"){
        nilai1 = 4;
    }else if(semesterBergabungOrganisasi == "Semester 5"){
        nilai1 = 5;
    }else if(semesterBergabungOrganisasi == "Semester 6"){
        nilai1 = 6;
    }else if(semesterBergabungOrganisasi == "Semester 7"){
        nilai1 = 7;
    }else if(semesterBergabungOrganisasi == "Semester 8"){
        nilai1 = 8;
    } else {
        nilai1 = 10;
    }
    String Target = ((String) jCmb_Durasi.getSelectedItem());
    if(Target == "< 4 Tahun"){
        nilai2 = 1;
    } else{
        nilai2 =0;
    }
    Pelatihan n = new Pelatihan("Select *from tabel_alumni");

    try{
        Statement statement = (Statement) Connection_mysql.getConnection().createStatement();
```

```
            statement.executeUpdate("update tabel_alumni SET id_alumni ='"+id_alumni+"',"+"SKS='"+SKS+"',"+"IPK='"+IPK+"',"+"jumlah_organisasi='"+jumlahOrganisasi+"',"+"Ketum='"+nilai_Ketum+"',"+"Waketum='"+nilai_Waketum+"',"+"Kadin='"+nilai_Kadin+"',"+"Staff='"+nilai_Staff+"',"+"lama_organisasi='"+nilai3+"',"+"semester_bergabungOrganisasi='"+nilai1+"',"+"Target='"+nilai2+"'" + "WHERE Id_alumni='"+id_alumni+"'");
            statement.close();
            JOptionPane.showMessageDialog(null, "Data Berhasil Diubah");
        } catch (Exception ex){
           JOptionPane.showMessageDialog(null, "Data Gagal Diubah");
        }
        n.tampil();
        }
    }

    private void jBtn_HapusActionPerformed(java.awt.event.ActionEvent evt) {
      String id_alumni = jTxt_IdAlumni.getText();

      try{
          Statement statement = (Statement) Connection_mysql.getConnection().createStatement();
          statement.executeUpdate("delete from tabel_alumni where id_alumni= '"+id_alumni+"';");
          JOptionPane.showMessageDialog(null, "Data Berhasil Dihapus");
          jTxt_IdAlumni.setText("");
          jTxt_SKS.setText("");
          jTxt_IPK.setText("");
          jTxt_jumlahOrganisasi.setText("");
          jCmb_Ketum.setSelectedItem("Tidak Pernah");
          jCmb_Waketum.setSelectedItem("Tidak Pernah");
          jCmb_Kadin.setSelectedItem("Tidak Pernah");
          jCmb_Staff.setSelectedItem("Tidak Pernah");
          jCmb_lamaOrganisasi.setSelectedItem("1 Tahun");
          jCmb_semesterBergabungOrganisasi.setSelectedItem("Semester 1");
          jCmb_Durasi.setSelectedItem("< 4 Tahun");
      } catch(Exception ex){
          JOptionPane.showMessageDialog(null, "Data Gagal Dihapus");
      }
    }

    private void jBtn_CariActionPerformed(java.awt.event.ActionEvent evt) {
        try{
        Statement statement = (Statement) Connection_mysql.getConnection().createStatement();
        ResultSet res = statement.executeQuery("select *from tabel_alumni where id_alumni = '"+ jTxt_Cari.getText()+"'");

        res.next();
        jTxt_IdAlumni.setText(res.getString(1));
        jTxt_SKS.setText(res.getString(2));
        jTxt_IPK.setText(res.getString(3));
        jTxt_jumlahOrganisasi.setText(res.getString(4));
        jCmb_Ketum.setSelectedItem(res.getString(5));
        jCmb_Waketum.setSelectedItem(res.getString(6));
        jCmb_Kadin.setSelectedItem(res.getString(7));
        jCmb_Staff.setSelectedItem(res.getString(8));
        jCmb_lamaOrganisasi.setSelectedItem(res.getString("Staff"));
        jCmb_semesterBergabungOrganisasi.setSelectedItem(res.getString(10));
        jCmb_Durasi.setSelectedItem(res.getString(11));
        } catch(Exception ex){
            System.out.println("pesan error= "+ex);
            JOptionPane.showMessageDialog(rootPane, "Error");
        }

    }
```

Prediksi.java

```
package prediksimasastudi;

import java.sql.SQLException;
import java.sql.Connection;
import java.sql.Statement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.text.DecimalFormat;
import javax.swing.JOptionPane;
import javax.swing.table.DefaultTableModel;
import prediksimasastudi.NeuralNetwork;
import javax.swing.JTable;

/**
 *
 * @author M. Syukron Azim
 */
public class Prediksi extends javax.swing.JFrame {

    public Connection con;
    public Statement stat;
    public ResultSet res;

    public JTable tabel_alumni;

    //double [][] data2 = new double[tabel_alumni.getRowCount()][9];

    String a;
    double b;
    int c,d,e,f,g;
    /**
     * Creates new form Prediksi
     */
    public Prediksi() {
        initComponents();
    }

    public JTable getTabel_alumni() {
        return tabel_alumni;
    }

    public void setTabel_alumni(JTable tabel_alumni) {
        this.tabel_alumni = tabel_alumni;
    }

private void jBtn_PrediksiActionPerformed(java.awt.event.ActionEvent evt) {
      int i=0, nilai_Ketum, nilai_Waketum, nilai_Kadin, nilai_Staff, nilai_lamaOrganisasi, nilai_Semester;
      Object [][] data = new Object[1][9];
      double target[][] = null;

       String a = jTxt_Nama.getText();
        System.out.println("Nama = "+a);
       String b = jTxt_SKS.getText();
        System.out.println("SKS = "+b);
       String c = jTxt_IPK.getText();
         System.out.println("IPK = "+c);
       String d = jTxt_jumlahOrganisasi.getText();
         System.out.println("Jumlah Organisasi ="+d);
       String e = ((String) jCmb_Ketum.getSelectedItem());
       if(e == "Tidak Pernah"){
           nilai_Ketum = 0;
       } else if(e == "1 Kali"){
```

```
    nilai_Ketum = 1;
} else if(e == "2 Kali"){
    nilai_Ketum = 2;
} else if(e == "3 Kali"){
    nilai_Ketum = 3;
} else if(e == "4 Kali"){
    nilai_Ketum = 4;
} else if(e == "5 Kali"){
    nilai_Ketum = 5;
} else if(e == "6 Kali"){
    nilai_Ketum = 6;
} else if(e == "7 Kali"){
    nilai_Ketum = 7;
} else if(e == "8 Kali"){
    nilai_Ketum = 8;
} else if(e == "9 Kali"){
    nilai_Ketum = 9;
} else if(e == "10 Kali"){
    nilai_Ketum = 10;
} else {
    nilai_Ketum = 12;
}
 System.out.println("Nilai Ketua Umum ="+nilai_Ketum);
String f = ((String) jCmb_Waketum.getSelectedItem());
 if(f == "Tidak Pernah"){
    nilai_Waketum = 0;
}   else if(f == "1 Kali"){
    nilai_Waketum = 1;
} else if(f == "2 Kali"){
    nilai_Waketum = 2;
} else if(f == "3 Kali"){
    nilai_Waketum = 3;
} else if(f == "4 Kali"){
    nilai_Waketum = 4;
} else if(f == "5 Kali"){
    nilai_Waketum = 5;
} else if(f == "6 Kali"){
    nilai_Waketum = 6;
} else if(f == "7 Kali"){
    nilai_Waketum = 7;
} else if(f == "8 Kali"){
    nilai_Waketum = 8;
} else if(f == "9 Kali"){
    nilai_Waketum = 9;
} else if(f == "10 Kali"){
    nilai_Waketum = 10;
} else {
    nilai_Waketum = 12;
}
 System.out.println("Nilai Wakil Ketua Umum ="+nilai_Waketum);
String g = ((String) jCmb_Kadin.getSelectedItem());
if(g == "Tidak Pernah"){
    nilai_Kadin = 0;
}   else if(g == "1 Kali"){
    nilai_Kadin = 1;
} else if(g == "2 Kali"){
    nilai_Kadin = 2;
} else if(g == "3 Kali"){
    nilai_Kadin = 3;
} else if(g == "4 Kali"){
    nilai_Kadin = 4;
} else if(g == "5 Kali"){
    nilai_Kadin = 5;
} else if(g == "6 Kali"){
```

```
        nilai_Kadin = 6;
    } else if(g == "7 Kali"){
        nilai_Kadin = 7;
    } else if(g == "8 Kali"){
        nilai_Kadin = 8;
    } else if(g == "9 Kali"){
        nilai_Kadin = 9;
    } else if(g == "10 Kali"){
        nilai_Kadin = 10;
    } else {
        nilai_Kadin = 12;
    }
    System.out.println("Nilai Kepala Dinas ="+nilai_Kadin);
    String h = ((String) jCmb_Staff.getSelectedItem());
    if(h == "Tidak Pernah"){
        nilai_Staff = 0;
    } else if(h == "1 Kali"){
        nilai_Staff = 1;
    } else if(h == "2 Kali"){
        nilai_Staff = 2;
    } else if(h == "3 Kali"){
        nilai_Staff = 3;
    } else if(h == "4 Kali"){
        nilai_Staff = 4;
    } else if(h == "5 Kali"){
        nilai_Staff = 5;
    } else if(h == "6 Kali"){
        nilai_Staff = 6;
    } else if(h == "7 Kali"){
        nilai_Staff = 7;
    } else if(h == "8 Kali"){
        nilai_Staff = 8;
    } else if(h == "9 Kali"){
        nilai_Staff = 9;
    } else if(h == "10 Kali"){
        nilai_Staff = 10;
    } else {
        nilai_Staff = 12;
    }
    System.out.println("Nilai Staff ="+nilai_Staff);
    String j = ((String) jCmb_lamaOrganisasi.getSelectedItem());
    if(j == "1 Tahun"){
        nilai_lamaOrganisasi = 1;
    } else if(j == "2 Tahun"){
        nilai_lamaOrganisasi = 2;
    } else if(j == "3 Tahun"){
        nilai_lamaOrganisasi = 3;
    } else if(j == "4 Tahun"){
        nilai_lamaOrganisasi = 4;
    } else {
        nilai_lamaOrganisasi = 5;
    }
    System.out.println("Lama Organisasi ="+nilai_lamaOrganisasi);
    String k = ((String) jCmb_semesterBergabungOrganisasi.getSelectedItem());
    if(k == "Semester 1"){
        nilai_Semester =1;
    } else if(k == "Semester 2"){
        nilai_Semester =2;
    } else if(k == "Semester 3"){
        nilai_Semester =3;
    } else if(k == "Semester 4"){
        nilai_Semester =4;
    }else if(k == "Semester 5"){
        nilai_Semester =5;
```

```
}else if(k == "Semester 6"){
   nilai_Semester =6;
}else if(k == "Semester 7"){
   nilai_Semester =7;
}else if(k == "Semester 8"){
   nilai_Semester =8;
} else{
   nilai_Semester = 10;
}
 System.out.println("Semester Bergabung = "+nilai_Semester);

 double tmpMin[] = new double [9];
 double tmpMax[] = new double [9];
 i = 0;
 Connection_mysql classKoneksi = new Connection_mysql();
  try{
     con = classKoneksi.getConnection();
     stat = con.createStatement();
     res = stat.executeQuery("select *from minmax");
     while (res.next()){
        tmpMin[i] = Double.parseDouble(res.getString(1));
        tmpMax[i] = Double.parseDouble(res.getString(2));
         i++;
     }
 }
 catch(SQLException ex){
     System.out.print(ex.getMessage());
 }
 double tmpSKS = ((((Double.parseDouble(b) - tmpMin[0])/(tmpMax[0]-tmpMin[0]))*(0.9-0.1))+0.1);
 double tmpIPK = ((((Double.parseDouble(c) - tmpMin[1])/(tmpMax[1] - tmpMin[1]))*(0.9 - 0.1)) + 
0.1);
 double tmp_jml_org = ((((Double.parseDouble(d) - tmpMin[2])/(tmpMax[2] - tmpMin[2]))*(0.9 - 0.1)) 
+ 0.1);
 double tmp_Ketum = (((((double)(nilai_Ketum) - tmpMin[3])/(tmpMax[3] - tmpMin[3]))*(0.9 - 0.1)) + 
0.1);
 double tmp_Waketum = (((((double)(nilai_Waketum) - tmpMin[4])/(tmpMax[4] - tmpMin[4]))*(0.9 - 
0.1)) + 0.1);
 double tmp_Kadin = (((((double)(nilai_Kadin) - tmpMin[5])/(tmpMax[5] - tmpMin[5]))*(0.9 - 0.1)) + 
0.1);
 double tmp_Staff = (((((double)(nilai_Staff) - tmpMin[6])/(tmpMax[6] - tmpMin[6]))*(0.9 - 0.1)) + 
0.1);
 double tmp_lama_org = (((((double)(nilai_lamaOrganisasi) - tmpMin[7])/(tmpMax[7] - 
tmpMin[7]))*(0.9 - 0.1)) + 0.1);
 double tmp_smstr_org = (((((double)(nilai_Semester) - tmpMin[8])/(tmpMax[8] - tmpMin[8]))*(0.9 - 
0.1)) + 0.1);

i=0;
try{
     con = classKoneksi.getConnection();
     stat = con.createStatement();
     res = stat.executeQuery("select *from outputpelatihan");
     System.out.println("ehe");
     target = new double [117][1];
     System.out.println("target = "+ target.length);
     while (res.next()){

         target[i][0] = (Double.parseDouble(res.getString(1))) ;
         i++;
     }
 }
 catch(SQLException ex){
     System.out.print(ex.getMessage());
 }
 Object [][] row = {{tmpSKS, tmpIPK, tmp_jml_org, tmp_Ketum, tmp_Waketum, tmp_Kadin,
```

```
tmp_Staff, tmp_lama_org, tmp_smstr_org}};

        double LR=0, ET=0;
        int epoch=0;

        try{
            res = stat.executeQuery("select *from parameter");
            while (res.next()){

             LR = Double.parseDouble(res.getString(1));
             ET = Double.parseDouble(res.getString(2));
             epoch = Integer.parseInt(res.getString(3));
            }
        }
        catch(SQLException ex){
            System.out.println("ahek");
            System.out.print(ex.getMessage());
        }
        System.out.println("LR = " +LR);
        System.out.println("ET = " +ET);
        System.out.println("epoch = " +epoch);
//      ============================== BPNN ONLY ==============================
       NeuralNetwork nn = new NeuralNetwork(9,12,1, row);
       nn.setLearningRate(LR);
       nn.setErrorTollerance(ET);
       nn.setEpoch(epoch);
       nn.run_Prediksi_BPNN(epoch,ET);

       double output[][];
       output = nn.getResultOutputs();

       double x = output[0][0];
       System.out.println("output = "+x);

       i=0;
       double[] tmp = new double [117];

        try{
           Statement statement = (Statement) Connection_mysql.getConnection().createStatement();
           res = statement.executeQuery("select *from outputpelatihan_bpnn");

           while(res.next()){
               tmp[i] = Double.parseDouble(res.getString(1));
               i++;
           }
           statement.close();
           JOptionPane.showMessageDialog(null, "Akses database output Berhasil");
       } catch (SQLException ex) {
           JOptionPane.showMessageDialog(null, "Akses database output Gagal"+ex);
        }

       double selisih;
       int index=0;
       selisih = x-tmp[0];
       for(int anjay=0; anjay<tmp.length; anjay++){
           if(x - tmp[anjay] <= selisih){// || x - tmp[anjay] >= selisih){ //tmp[anjay] - x <=selisih
               index = anjay;
               selisih = x - tmp[anjay];
           }
           System.out.println("data ke -  "+anjay + "  target = "+tmp[anjay] + "  selisih = "+selisih);
       }
        System.out.println("index prediksi mendekati = "+index);
        System.out.println("nilai database = "+tmp[index] + "  nilai prediksi = "+x);
```

```
Object[][] dataasli = new Object[117][11];

 i=0;
 try{
    Statement statement = (Statement) Connection_mysql.getConnection().createStatement();
    res = statement.executeQuery("select *from database_final");
    while(res.next()){
        String[] baris = {res.getString(1),res.getString(2), res.getString(3), res.getString(4), res.getString(5),
res.getString(6), res.getString(7), res.getString(8), res.getString(9), res.getString(10), res.getString(11),
res.getString(12), res.getString(13)};
        dataasli[i] = baris;
        i++;
    }
    statement.close();
    JOptionPane.showMessageDialog(null, "Database Normalisasi Berhasil Diakses");
} catch (SQLException ex) {
    JOptionPane.showMessageDialog(null, "Database Normalisasi Gagal Diakses"+ex);
 }

DecimalFormat df = new DecimalFormat ("#.##");

jTxt_Accuracy2.setText(String.valueOf(df.format((tmp[index]) * 100) + "%"));
 System.out.println("data asli klasifiikasi = "+dataasli[index][10]);
if(Double.parseDouble(dataasli[index][10].toString()) == 1){
    jTxt_Klasifikasi2.setText("Tepat Waktu");
}
else{
    jTxt_Klasifikasi2.setText("Tidak Tepat Waktu");
}

nn.setLearningRate(LR);
nn.setErrorTollerance(ET);
nn.setEpoch(epoch);
nn.run_Prediksi_ESABPNN(epoch,ET);

double output_ESABPNN[][];
output_ESABPNN = nn.getResultOutputs();

double y = output_ESABPNN[0][0];
System.out.println("output = "+y);


i=0;
double[] tmp_ESABPNN =  new double[117];
 try{
    Statement statement = (Statement) Connection_mysql.getConnection().createStatement();
     res = statement.executeQuery("select *from database_final");
    while(res.next()){
        tmp_ESABPNN[i] = Double.parseDouble(res.getString(12));
        i++;
    }
    statement.close();
    JOptionPane.showMessageDialog(null, "Akses database output Berhasil");
} catch (SQLException ex) {
    JOptionPane.showMessageDialog(null, "Akses database output Gagal"+ex);
 }

double selisih_ESABPNN;
int index_ESABPNN=0;
selisih_ESABPNN = y-tmp_ESABPNN[0];
for(int anjay=0; anjay<tmp_ESABPNN.length; anjay++){
    if(y - tmp_ESABPNN[anjay] <= selisih_ESABPNN){
        index_ESABPNN = anjay;
        selisih_ESABPNN = y - tmp_ESABPNN[anjay];
```

```
        }
        System.out.println("data ke -  "+anjay + "  target = "+tmp_ESABPNN[anjay] + "  selisih = "+selisih_ESABPNN);
      }
      System.out.println("index prediksi mendekati = "+index_ESABPNN);
      System.out.println("nilai database = "+tmp_ESABPNN[index_ESABPNN] + "  nilai prediksi = "+y);

      Object[][] dataasli_ESABPNN = new Object[117][13];

      i=0;
      try{
        Statement statement = (Statement) Connection_mysql.getConnection().createStatement();
        res = statement.executeQuery("select *from database_final");
        while(res.next()){
            String[] baris = {res.getString(1),res.getString(2), res.getString(3), res.getString(4), res.getString(5), res.getString(6), res.getString(7), res.getString(8), res.getString(9), res.getString(10), res.getString(11), res.getString(12), res.getString(13)};
            dataasli_ESABPNN[i] = baris;
            i++;
        }
        statement.close();
        JOptionPane.showMessageDialog(null, "Database Normalisasi Berhasil Diakses");
      } catch (SQLException ex) {
        JOptionPane.showMessageDialog(null, "Database Normalisasi Gagal Diakses"+ex);
      }

      jTxt_Accuracy1.setText(String.valueOf(df.format((tmp_ESABPNN[index_ESABPNN]) * 100) + "%"));
      System.out.println("data asli klasifiikasi = "+dataasli_ESABPNN[index_ESABPNN][10]);
      if(Double.parseDouble(dataasli_ESABPNN[index_ESABPNN][10].toString()) == 1){
        jTxt_Klasifikasi1.setText("Tepat Waktu");
      }
      else{
        jTxt_Klasifikasi1.setText("Tidak Tepat Waktu");
      }
    }

    private void jBtn_KembaliActionPerformed(java.awt.event.ActionEvent evt) {
      Main n = new Main();
      n.setVisible(true);
      this.setVisible(false);
    }
```

NeuralNetwork.java

```
package prediksimasastudi;

import java.sql.SQLException;
import java.sql.Statement;
import java.text.*;
import java.util.*;
import javax.swing.JTable;
import javax.swing.JTextField;
import java.util.Random;
import javax.swing.JOptionPane;
import prediksimasastudi.Pelatihan;

/**
 *
 * @author M. Syukron Azim
 */
public class NeuralNetwork {
    static {
```

```
        Locale.setDefault(Locale.ENGLISH);
    }

     public JTable tabel_alumni;
     public JTextField   jTxt_LifeLimit, jTxt_Dimension, jTxt_MaxGen,jMSE, jMSE2, jEpoch, jEpoch2,
jAccuracy, jAccuracy2;
     public static JTextField jTxt_ErrorTolerance, jTxt_Epoch, jTxt_LearningRate;

    final boolean isTrained = false;
    final DecimalFormat df;
    final Random rand = new Random();

    ArrayList<Neuron> inputLayer = new ArrayList<Neuron>();
    ArrayList<Neuron> hiddenLayer = new ArrayList<Neuron>();
    ArrayList<Neuron> outputLayer = new ArrayList<Neuron>();
     final Neuron bias = new Neuron ();
     int[] layers;
     final int randomWeightMultiplier = 1;

    Object[][] data;
    double [][] data2;
    double [][] data3;
    double[][] target;
    double resultOutputs[][] = { { -1 }, { -1 }, { -1 }, { -1 } };
    double output[];

    double W_error;
    int W_epoch;
    double W_accuracy;

    //FORM PELATIHAN
    final double epsilon = 0.00000000001;
    double errorTollerance;
    double learningRate;
    final double momentum = 0.7f;
    int epoch;

    public double getLearningRate() {
        return learningRate;
    }

    public void setLearningRate(double jTxt_LearningRate) {
        this.learningRate = jTxt_LearningRate;
    }

    public double getErrorTollerance() {
        return errorTollerance;
    }

    public void setErrorTollerance(double jTxt_ErrorTollerance) {
        this.errorTollerance = jTxt_ErrorTollerance;
    }

    public int getEpoch() {
        return epoch;
    }

    public void setEpoch(int jTxt_Epoch) {
        this.epoch = jTxt_Epoch;
    }

    public double getW_error() {
        return W_error;
    }
```

```
    public void setW_error(double W_error) {
        this.W_error = W_error;
    }

    public int getW_epoch() {
        return W_epoch;
    }

    public void setW_epoch(int W_epoch) {
        this.W_epoch = W_epoch;
    }

    public double getW_accuracy() {
        return W_accuracy;
    }

    public void setW_accuracy(double W_accuracy) {
        this.W_accuracy = W_accuracy;
    }

    public double[][] getResultOutputs() {
        return resultOutputs;
    }

    public void setResultOutputs(double[][] Outputs) {
        this.resultOutputs = Outputs;
    }

    Pelatihan pelatihan = new Pelatihan("Select *from tabel_alumni");

    final HashMap<String,Double> weightUpdate = new HashMap<String,Double>();

//============================== PELATIHAN ====================================

MAIN OF  NEURAL NETWORK ==============
    public NeuralNetwork(int input, int hidden, int output, Object [][] data, double[][] target){
        this.layers = new int[] {input, hidden, output};
        df = new DecimalFormat("#.0#");

        for (int i = 0 ; i < layers.length; i++){
            if (i==0) {
                for (int j = 0 ; j < layers[i]; j++){
                Neuron neuron = new Neuron();
                inputLayer.add(neuron);
                }
            }
        //Hidden Layer
            else if (i==1){
            for(int j = 0 ; j<layers[i];j++){
                Neuron neuron = new Neuron();
                neuron.addInConnectionsS(inputLayer);
                neuron.addBiasConnection(bias);
                outputLayer.add(neuron);
                }
            }
        //Output Layer
            else if (i==2){
            for(int j = 0 ; j<layers[i];j++){
                Neuron neuron = new Neuron();
                neuron.addInConnectionsS(hiddenLayer);
                neuron.addBiasConnection(bias);
                outputLayer.add(neuron);
                }
```

```
        }
        else {
            System.out.println("!Error NeuralNetwork Init");
        }
    }
    //initialize random weights
    for(Neuron neuron : hiddenLayer){
        ArrayList<Connection> connections = neuron.getAllInConnections();
        for(Connection conn : connections){
            double newWeight = getRandom();
            conn.setWeight(newWeight);
        }
    }
    for (Neuron neuron : outputLayer){
        ArrayList<Connection> connections = neuron.getAllInConnections();
        for(Connection conn : connections){
            double newWeight = getRandom();
            conn.setWeight(newWeight);
        }
    }

    //reset id Counters
    Neuron.counter = 0;
    Connection.counter = 0;

    if(isTrained){
        trainedWeights();
        updateAllWeights();
    }

    this.data = data;
    data2= new double [data.length][9];
    data3 = new double [data.length][9];
    this.target = target;

    for(int i=0 ; i<data.length; i++){

        for(int j=0; j<data[i].length;j++){
            data2[i][j] = Double.parseDouble(data[i][j].toString()) ;
        }
    }

    for(int i=0 ; i<data.length; i++){
        for(int j=0; j<data[i].length;j++){
            data3[i][j] = Double.parseDouble(data[i][j].toString()) ;
        }
    }

    resultOutputs = new double[target.length][1];
    for(int i=0;i<target.length;i++){
        for(int j=0; j< 1; j++){
            resultOutputs[i][j] = -1;
        }
    }
}

//random
double getRandom(){
return randomWeightMultiplier * (rand.nextDouble() * 2 - 1); // [1-1]
}

public void setInput(double inputs[]){
    for(int i = 0; i < inputLayer.size(); i++){
        inputLayer.get(i).setOutput(inputs[i]);
```

```
        }
    }

    public double[] getOutput(){
        double[] outputs = new double[outputLayer.size()];
        for (int i= 0; i< outputLayer.size(); i++)
        outputs[i] = outputLayer.get(i).getOutput();
        return outputs;
    }

    //calculate the output based on the input the forward, operation on below:
    public void activate(){
        for(Neuron n : hiddenLayer)
            n.calculateOutput();
        for(Neuron n : outputLayer)
            n.calculateOutput();
    }

     public void applyBackpropogation_Pelatihan(double expectedOutput[]){

             for(int i = 0 ; i < expectedOutput.length; i++){
            double d = expectedOutput[i];
            if(d<0 || d > 1){
                if(d<0)
                    expectedOutput[i] = 0 + epsilon;
                else
                    expectedOutput[i] = 1 - epsilon;
            }
        }
       //update bobot untuk output layer
        int i = 0;
        for (Neuron n : outputLayer){
            ArrayList<Connection> connections = n.getAllInConnections();
            for (Connection con : connections){
                double ak = n.getOutput();
                double ai = con.leftNeuron.getOutput();
                double desiredOutput = output[i];

                double partialDerivative = -ak * (1 - ak) * ai * (desiredOutput - ak);
                double deltaWeight = -learningRate * partialDerivative;
                double newWeight = con.getWeight() + deltaWeight;
                con.setDeltaWeight(deltaWeight);
                con.setWeight(newWeight + momentum * con.getPrevDeltaWeight());
            }
            i++;
        }

    //update bobot untuk hidden layer
    for(Neuron n : hiddenLayer){
        ArrayList<Connection> connections = n.getAllInConnections();
        for (Connection con : connections){
            double aj = n.getOutput();
            double ai = con.leftNeuron.getOutput();
            double sumKoutputs = 0;
            int j = 0;
            for (Neuron out_neu : outputLayer){
                double wjk = out_neu.getConnection(n.id).getWeight();
                double desiredOutput = (double) expectedOutput[j];
//                double desiredOutput = (double) target[j][0];
                double ak = out_neu.getOutput();
                j++;
                sumKoutputs = sumKoutputs + (-(desiredOutput - ak) *ak *(1-ak) *wjk);
        }
```

```
        double partialDerivative = aj * (1-aj) * ai * sumKoutputs;
        double deltaWeight = -learningRate * partialDerivative;
        double newWeight = con.getWeight() + deltaWeight;
        con.setDeltaWeight(deltaWeight);
        con.setWeight(newWeight + momentum * con.getPrevDeltaWeight());
        }
    }
}
//=============================== BPNN ONLY ================================
    void run_Pelatihan_BPNN (int maxSteps, double minError){
        int i;
        double error = 1;
        double accT, accF;
        double accuracy = 0;
        for (i= 0 ; i< maxSteps && error > minError; i++){
            error = 0;

            for(int p = 0; p < data2.length; p++){
                setInput(data2[p]);

                activate();

                output = getOutput();
               resultOutputs[p] = output;
                for(int j = 0; j< target[p].length;j++){

                    double err = Math.pow(output[j] - target[p][j],2);
                    error += err;
                }
                applyBackpropogation_Pelatihan(target[p]);
            }
        }

        printResult_Pelatihan_BPNN();

        if(i == maxSteps){
            System.out.println("!Error training try again");
        } else{
            printAllWeights();
            printWeightUpdate();
        }


        String w = df.format(error);
        setW_error(Double.parseDouble(w));
        System.out.println("Sum of Squared Error = "+ error);

          setW_epoch(i);
          System.out.println("##### EPOCH " + i+ "\n");

       int p=0;
       for(int x = 0; x < data.length ; x++){
           if(target[x][0] == Math.round(resultOutputs[x][0])){
               System.out.println("Index = "+x+ ", Target = "+target[x][0] + ", Output = "+resultOutputs[x][0]);
               p++;
           }
       }

       accT = p;
       accF = data.length - p;
       accuracy = (double) ((accT/data.length) *100);
        System.out.println(" "+ accT + "," +accF);

       String K = df.format(accuracy);
```

```
    System.out.println("Accuracy = "+K);
    setW_accuracy(Double.parseDouble(K));
  }

  void printResult_Pelatihan_BPNN(){
     System.out.println("Hasil Proses Neural Network: ");
     for(int p=0; p< data2.length; p++){
     System.out.print("Inputs : ");
     for(int x=0;x<layers[0];x++){
        System.out.print(data[p][x] + " ");
        }
     System.out.print("Expected : ");
     for(int x = 0; x < layers[2]; x++){
        System.out.print(target[p][x] + " ");

        }
     System.out.print("Actual : ");
     for(int x = 0; x < layers[2]; x++){
        System.out.print(resultOutputs[p][x] + " ");
        }
        System.out.println();
     }
     System.out.println();
  }

//================================= ESA - BPNN =================================
  void run_Pelatihan_ESABPNN (int maxSteps, double minError){
     int i;
     double error = 1;
     double accT, accF;
     double accuracy = 0;
     for (i= 0 ; i< maxSteps && error > minError; i++){
        error = 0;

        for(int p = 0; p < data3.length; p++){
           setInput(data3[p]);

           activate();

           output = getOutput();
          resultOutputs[p] = output;
           for(int j = 0; j< target[p].length;j++){

              double err = Math.pow(output[j] - target[p][j],2);
              error += err;
           }
           applyBackpropogation_Pelatihan(target[p]);
        }
     }
     printResult_Pelatihan_ESABPNN();

     if(i == maxSteps){
        System.out.println("!Error training try again");
     } else{
        printAllWeights();
        printWeightUpdate();
     }

     String w = df.format(error);
     setW_error(Double.parseDouble(w));
     System.out.println("Sum of Squared Error = "+ error);

      setW_epoch(i);
      System.out.println("##### EPOCH " + i+ "\n");
```

```
    int p=0;

    for(int x = 0; x < data.length ; x++){
       if(target[x][0] == Math.round(resultOutputs[x][0])){
          System.out.println("Target = "+target[x][0] + ", Output ="+resultOutputs[x][0]);
          p++;
       }
    }

    accT = p;
    accF = data.length - p;
    accuracy = (double) ((accT/data.length) * 100);
    System.out.println(" "+ accT + "," +accF);
    String K = df.format(accuracy);
    System.out.println("Accuracy = "+K);
    setW_accuracy(Double.parseDouble(K));
  }

  void printResult_Pelatihan_ESABPNN(){
     System.out.println("Hasil Proses Neural Network: ");
     for(int p=0; p< data3.length; p++){
     System.out.print("Inputs : ");
     for(int x=0;x<layers[0];x++){
        System.out.print(data[p][x] + " ");
        }
     System.out.print("Expected : ");
     for(int x = 0; x < layers[2]; x++){
        System.out.print(target[p][x] + " ");

        }
     System.out.print("Actual : ");
     for(int x = 0; x < layers[2]; x++){
        System.out.print(resultOutputs[p][x] + " ");
        }
        System.out.println();
     }
     System.out.println();
  }
//=============================== BATAS ESA - BPNN ==============================

//================================== PREDIKSI ==================================

  public NeuralNetwork(int input, int hidden, int output, Object [][] data){
     this.layers = new int[] {input, hidden, output};
     df = new DecimalFormat("#.0#");

     for (int i = 0 ; i < layers.length; i++){
        if (i==0) {
           for (int j = 0 ; j < layers[i]; j++){
           Neuron neuron = new Neuron();
           inputLayer.add(neuron);
           }
        }
     //Hidden Layer
        else if (i==1){
        for(int j = 0 ; j<layers[i];j++){
           Neuron neuron = new Neuron();
           neuron.addInConnectionsS(inputLayer);
           neuron.addBiasConnection(bias);
           outputLayer.add(neuron);
           }
        }
     //Output Layer
```

```
            else if (i==2){
            for(int j = 0 ; j<layers[i];j++){
                Neuron neuron = new Neuron();
                neuron.addInConnectionsS(hiddenLayer);
                neuron.addBiasConnection(bias);
                outputLayer.add(neuron);
                }
            }
            else {
                System.out.println("!Error NeuralNetwork Init");
            }
        }
        //initialize random weights
        for(Neuron neuron : hiddenLayer){
            ArrayList<Connection> connections = neuron.getAllInConnections();
            for(Connection conn : connections){
                double newWeight = getRandom();
                conn.setWeight(newWeight);
            }
        }
        for (Neuron neuron : outputLayer){
            ArrayList<Connection> connections = neuron.getAllInConnections();
            for(Connection conn : connections){
                double newWeight = getRandom();
                conn.setWeight(newWeight);
            }
        }

        //reset id Counters
        Neuron.counter = 0;
        Connection.counter = 0;

        if(isTrained){
            trainedWeights();
            updateAllWeights();
        }

        this.data = data;
        data2= new double [data.length][9];
        data3 = new double [(int)Math.round(data.length * 0.4)][9];

        for(int i=0 ; i<data.length; i++){
            for(int j=0; j<data[i].length;j++){
                data2[i][j] = Double.parseDouble(data[i][j].toString()) ;
            }
        }

        for(int i=0; i<(int)Math.round(data.length*0.4); i++){
            for(int j=0; j<(int)Math.round(data[i].length*0.4); j++){
                data3[i][j] = Double.parseDouble(data[i][j].toString());
            }
        }
}

//============================= BPNN ONLY ===================================
        void run_Prediksi_BPNN (int maxSteps, double minError){
        int i;
        //train neural network until minError reached or maxSteps exceeded
        double error = 1;
        for (i= 0 ; i< maxSteps && error > minError; i++){
            error = 0;

            for(int p = 0; p < data2.length; p++){
                setInput(data2[p]);
```

```
            activate();

            output = getOutput();
           resultOutputs[p] = output;

            applyBackpropogation_Prediksi_BPNN();
        }
    }

    printResult_Prediksi_BPNN();

    if(i == maxSteps){
        System.out.println("!Error training try again");
    } else{
        printAllWeights();
        printWeightUpdate();
    }

    System.out.println("Sum of Squared Error = "+ error);

    System.out.println("##### EPOCH " + i+ "\n");

}

public void applyBackpropogation_Prediksi_BPNN(){

     //update bobot untuk output layer
    int i = 0;
    for (Neuron n : outputLayer){
        ArrayList<Connection> connections = n.getAllInConnections();
        for (Connection con : connections){
            double ak = n.getOutput();
            double ai = con.leftNeuron.getOutput();
            double desiredOutput = output[i];

            double partialDerivative = -ak * (1 - ak) * ai * (desiredOutput - ak);
            double deltaWeight = -learningRate * partialDerivative;
            double newWeight = con.getWeight() + deltaWeight;
            con.setDeltaWeight(deltaWeight);
            con.setWeight(newWeight + momentum * con.getPrevDeltaWeight());
        }
        i++;
    }

//update bobot untuk hidden layer
for(Neuron n : hiddenLayer){
    ArrayList<Connection> connections = n.getAllInConnections();
    for (Connection con : connections){
        double aj = n.getOutput();
        double ai = con.leftNeuron.getOutput();
        double sumKoutputs = 0;
        int j = 0;
        for (Neuron out_neu : outputLayer){
            double wjk = out_neu.getConnection(n.id).getWeight();
            double desiredOutput = (double) output[j];
            double ak = out_neu.getOutput();
            j++;
            sumKoutputs = sumKoutputs + (-(desiredOutput - ak) *ak *(1-ak) *wjk);
    }

    double partialDerivative = aj * (1-aj) * ai * sumKoutputs;
    double deltaWeight = -learningRate * partialDerivative;
    double newWeight = con.getWeight() + deltaWeight;
```

```
        con.setDeltaWeight(deltaWeight);
        con.setWeight(newWeight + momentum * con.getPrevDeltaWeight());
        }
    }
}

    void printResult_Prediksi_BPNN(){
        System.out.println("Hasil Proses Neural Network: \n");
        for(int p=0; p< data2.length; p++){
        System.out.print("Inputs : ");
        for(int x=0;x<layers[0];x++){
            System.out.print(data[p][x] + " ");
            }
        System.out.print("Actual : ");
        for(int x = 0; x < layers[2]; x++){
            System.out.print(resultOutputs[p][x] + " ");
            }
            System.out.println();
        }
        System.out.println();
    }
//================================= ESA - BPNN ================================
    void run_Prediksi_ESABPNN (int maxSteps, double minError){
        int i;
        //train neural network until minError reached or maxSteps exceeded
        double error = 1;
        for (i= 0 ; i< maxSteps && error > minError; i++){
            error = 0;

            for(int p = 0; p < data3.length; p++){
                setInput(data3[p]);

                activate();

                output = getOutput();
               resultOutputs[p] = output;

                applyBackpropogation_Prediksi_ESABPNN();
            }
        }

        printResult_Prediksi_ESABPNN();

        if(i == maxSteps){
            System.out.println("!Error training try again");
        } else{
            printAllWeights();
            printWeightUpdate();
        }

        System.out.println("Sum of Squared Error = "+ error);
        System.out.println("##### EPOCH " + i+ "\n");
  }

    public void applyBackpropogation_Prediksi_ESABPNN(){
      //update bobot untuk output layer
        int i = 0;
        for (Neuron n : outputLayer){
            ArrayList<Connection> connections = n.getAllInConnections();
            for (Connection con : connections){
                double ak = n.getOutput();
                double ai = con.leftNeuron.getOutput();
                double desiredOutput = output[i];
```

```
            double partialDerivative = -ak * (1 - ak) * ai * (desiredOutput - ak);
            double deltaWeight = -learningRate * partialDerivative;
            double newWeight = con.getWeight() + deltaWeight;
            con.setDeltaWeight(deltaWeight);
            con.setWeight(newWeight + momentum * con.getPrevDeltaWeight());
         }
         i++;
      }

   //update bobot untuk hidden layer
   for(Neuron n : hiddenLayer){
      ArrayList<Connection> connections = n.getAllInConnections();
      for (Connection con : connections){
         double aj = n.getOutput();
         double ai = con.leftNeuron.getOutput();
         double sumKoutputs = 0;
         int j = 0;
         for (Neuron out_neu : outputLayer){
            double wjk = out_neu.getConnection(n.id).getWeight();
            double desiredOutput = (double) output[j];
            double ak = out_neu.getOutput();
            j++;
            sumKoutputs = sumKoutputs + (-(desiredOutput - ak) *ak *(1-ak) *wjk);
      }

      double partialDerivative = aj * (1-aj) * ai * sumKoutputs;
      double deltaWeight = -learningRate * partialDerivative;
      double newWeight = con.getWeight() + deltaWeight;
      con.setDeltaWeight(deltaWeight);
      con.setWeight(newWeight + momentum * con.getPrevDeltaWeight());
      }
   }
}

   void printResult_Prediksi_ESABPNN(){
      System.out.println("Hasil Proses Neural Network: ");
      for(int p=0; p< data3.length; p++){
      System.out.print("Inputs : ");
      for(int x=0;x<layers[0];x++){
         System.out.print(data[p][x] + " ");
         }
      System.out.print("Actual : ");
      for(int x = 0; x < layers[2]; x++){
         System.out.print(resultOutputs[p][x] + " ");
         }
         System.out.println();
      }
      System.out.println();
   }
//================================ BATAS PREDIKSI ============================

   String weightKey(int neuronId, int conId) {
      return "N" + neuronId + "_C" + conId;
   }

   public void updateAllWeights() {
      // update weights for the output layer
      for (Neuron n : outputLayer) {
         ArrayList<Connection> connections = n.getAllInConnections();
         for (Connection con : connections) {
            String key = weightKey(n.id, con.id);
            double newWeight = weightUpdate.get(key);
            con.setWeight(newWeight);
         }
```

```
        }
        // update weights for the hidden layer
        for (Neuron n : hiddenLayer) {
            ArrayList<Connection> connections = n.getAllInConnections();
            for (Connection con : connections) {
                String key = weightKey(n.id, con.id);
                double newWeight = weightUpdate.get(key);
                con.setWeight(newWeight);
            }
        }
    }

    // trained data
    void trainedWeights() {
        weightUpdate.clear();

        weightUpdate.put(weightKey(10, 0), 1.03);
        weightUpdate.put(weightKey(10, 1), 7.24);
        weightUpdate.put(weightKey(10, 2), -3.28);
        weightUpdate.put(weightKey(10, 3), 5.86);
        weightUpdate.put(weightKey(10, 4), 2.19);
        weightUpdate.put(weightKey(10, 5), 11.81);
        weightUpdate.put(weightKey(10, 6), 1.25);
        weightUpdate.put(weightKey(10, 7), 3.21);
        weightUpdate.put(weightKey(10, 8), -0.89);
        weightUpdate.put(weightKey(10, 9), 1.0);
        weightUpdate.put(weightKey(11, 10), 1.13);
        weightUpdate.put(weightKey(11, 11), -3.71);
        weightUpdate.put(weightKey(11, 12), 7.29);
        weightUpdate.put(weightKey(11, 13), 6.03);
        weightUpdate.put(weightKey(11, 14), -8.82);
        weightUpdate.put(weightKey(11, 15), 0.44);
        weightUpdate.put(weightKey(11, 16), 0.90);
        weightUpdate.put(weightKey(11, 17), 6.34);
        weightUpdate.put(weightKey(11, 18), 1.15);
        weightUpdate.put(weightKey(11, 19), 1.0);
        weightUpdate.put(weightKey(12, 20), -0.97);
        weightUpdate.put(weightKey(12, 21), -0.51);
        weightUpdate.put(weightKey(12, 22), -0.05);
        weightUpdate.put(weightKey(12, 23), 0.71);
        weightUpdate.put(weightKey(12, 24), -8.84);
        weightUpdate.put(weightKey(12, 25), 5.25);
        weightUpdate.put(weightKey(12, 26), -0.78);
        weightUpdate.put(weightKey(12, 27), 2.44);
        weightUpdate.put(weightKey(12, 28), 5.34);
        weightUpdate.put(weightKey(12, 29), 1.0);
        weightUpdate.put(weightKey(13, 30), 1.17);
        weightUpdate.put(weightKey(13, 31), 3.25);
        weightUpdate.put(weightKey(13, 32), -0.98);
        weightUpdate.put(weightKey(13, 33), 1.0);
    }

    public void printWeightUpdate() {
        System.out.println("printWeightUpdate, put this i trainedWeights() and set isTrained to true");
        // weights for the hidden layer
        for (Neuron n : hiddenLayer) {
            ArrayList<Connection> connections = n.getAllInConnections();
            for (Connection con : connections) {
                String w = df.format(con.getWeight());
                System.out.println("weightUpdate.put(weightKey(" + n.id + ", "
                        + con.id + "), " + w + ");");
            }
        }
        // weights for the output layer
```

```
        for (Neuron n : outputLayer) {
            ArrayList<Connection> connections = n.getAllInConnections();
            for (Connection con : connections) {
                String w = df.format(con.getWeight());
                System.out.println("weightUpdate.put(weightKey(" + n.id + ", "
                        + con.id + "), " + w + ");");
            }
        }
        System.out.println();
    }

    public void printAllWeights() {
        System.out.println("printAllWeights");
        // weights for the hidden layer
        for (Neuron n : hiddenLayer) {
            ArrayList<Connection> connections = n.getAllInConnections();
            for (Connection con : connections) {
                double w = con.getWeight();
                System.out.println("n=" + n.id + " c=" + con.id + " w=" + w);
            }
        }
        // weights for the output layer
        for (Neuron n : outputLayer) {
            ArrayList<Connection> connections = n.getAllInConnections();
            for (Connection con : connections) {
                double w = con.getWeight();
                System.out.println("n=" + n.id + " c=" + con.id + " w=" + w);
            }
        }
        System.out.println();
    }
}
```

Neuron.java

```
package prediksimasastudi;

import java.util.*;


/**
 *
 * @author M. Syukron Azim
 */
public class Neuron {
    static int counter = 0;
    final public int id;  // auto increment, starts at 0
    Connection biasConnection;
    final double bias = 1;
    double output;

    ArrayList<Connection> Inconnections = new ArrayList<Connection>();
    HashMap<Integer,Connection> connectionLookup = new HashMap<Integer,Connection>();

    public Neuron(){
        id = counter;
        counter++;
    }

    public void calculateOutput(){
        double s = 0;
        for(Connection con : Inconnections){
```

```
            Neuron leftNeuron = con.getFromNeuron();
            double weight = con.getWeight();
            double a = leftNeuron.getOutput(); //output from previous layer

            s = s + (weight*a);
        }
        s = s + (biasConnection.getWeight()*bias);
        output = g(s);
    }

    double g(double x) {
        return sigmoid(x);
    }

    double sigmoid(double x) {
        return 1.0 / (1.0 +  (Math.exp(-x)));
    }

    public void addInConnectionsS(ArrayList<Neuron> inNeurons){
        for(Neuron n: inNeurons){
            Connection con = new Connection(n,this);
            Inconnections.add(con);
            connectionLookup.put(n.id, con);
        }
    }

    public Connection getConnection(int neuronIndex){
        return connectionLookup.get(neuronIndex);
    }

    public void addInConnection(Connection con){
        Inconnections.add(con);
    }
    public void addBiasConnection(Neuron n){
        Connection con = new Connection(n,this);
        biasConnection = con;
        Inconnections.add(con);
    }
    public ArrayList<Connection> getAllInConnections(){
        return Inconnections;
    }

    public double getBias() {
        return bias;
    }
    public double getOutput() {
        return output;
    }
    public void setOutput(double o){
        output = o;
    }
}
```

Connection.java

```
package prediksimasastudi;

/**
 *
 * @author M. Syukron Azim
 */
public class Connection {
```

```
    double weight = 0;
    double prevDeltaWeight = 0;
    double deltaWeight = 0;
     final Neuron leftNeuron;
    final Neuron rightNeuron;
    static int counter = 0;
    final public int id; // auto increment, starts at 0

    public Connection(Neuron fromN, Neuron toN) {
        leftNeuron = fromN;
        rightNeuron = toN;
        id = counter;
        counter++;
    }

    public double getWeight() {
        return weight;
    }

    public void setWeight(double w) {
        weight = w;
    }

    public void setDeltaWeight(double w) {
        prevDeltaWeight = deltaWeight;
        deltaWeight = w;
    }

    public double getPrevDeltaWeight() {
        return prevDeltaWeight;
    }

    public Neuron getFromNeuron() {
        return leftNeuron;
    }

    public Neuron getToNeuron() {
        return rightNeuron;
    }
}
```

ElephantSearch_Aji.java

```
package prediksimasastudi;

import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.text.DecimalFormat;
import static javafx.scene.input.KeyCode.T;
import prediksimasastudi.Elephant_Aji;
import javax.swing.JTable;
import java.util.ArrayList;
import java.util.Vector;
import java.util.TreeSet;
import javax.swing.JOptionPane;
import javax.swing.table.DefaultTableModel;

/**
 *
 * @author M. Syukron Azim
 */
```

```
public class ElephantSearch_Aji {
    int m_random; //Variabel untuk jumlah generasi acak yg sekarang
    double m_eval; //Variabel untuk evaluasi fungsi yg digunakan sekarang
    double m_problem; //Variabel untuk masalah yg dihadapi sekarang
    int m_domain =1; //Variabel untuk dimensi yang terpilih
    double m_checkOptime; //Variabel untuk mengecheck apakah pos tersebut merupakan nilai yang paling
optimal
    double m_maxEval; //Variabel untuk hasil Evaluasi yang terbaik

    public ResultSet res;

     Vector<Elephant_Aji> Mpop = new Vector<Elephant_Aji>();
     Vector<Elephant_Aji> Fpop = new Vector<Elephant_Aji>();
     TreeSet<Integer> Position =  new TreeSet<Integer>();
     ArrayList<Double> ListBestLocal = new ArrayList<Double>();
    double BestGlobal; //Index Gajah yg memiliki Best Global
    double Value_ElephantBestGlobal; //Nilai Gajah yg memiliki Best Global
    public Elephant_Aji BestLocal; //Index Gajah yg memiliki Best Local
    public Elephant_Aji leader; //Pemimpin Kawanan gajah (Gajah Betina)
    int LimitLife; //Umur Gajah
    public Elephant_Aji MaleElephant; //Nilai Posisi Gajah Jantan
    int Iterasi =0; //Iterasi
    double MaleElephantBestGlobal;//Index Gajah Jantan yg terbaik
    double Value_MaleElephantBestGlobal; //Nilai Gajah Jantan yg terbaik
    public JTable tabel_alumni;

    Object[][] data;
    double [][] data2;

    ElephantSearch_Aji(double [][] data2){
        data2= new double[data.length][9];
    }

     ElephantSearch_Aji() {
    }

     public JTable getTabel_alumni() {
     return tabel_alumni;
     }

     public void setTabel_alumni(JTable tabel_alumni) {
     this.tabel_alumni = tabel_alumni;
     DecimalFormat df = new DecimalFormat ("#.##");

     try{
            Statement statement = (Statement) Connection_mysql.getConnection().createStatement();
            res = statement.executeQuery("select *from normalisasi");
            int i = 0;
            while (res.next()){


  double vector = Math.sqrt(Math.pow(Double.parseDouble(res.getString(2)),2) +
                         Math.pow(Double.parseDouble(res.getString(3)),2) +
                         Math.pow(Double.parseDouble(res.getString(4)),2) +
                         Math.pow(Double.parseDouble(res.getString(5)),2) +
                         Math.pow(Double.parseDouble(res.getString(6)),2) +
                         Math.pow(Double.parseDouble(res.getString(7)),2) +
                         Math.pow(Double.parseDouble(res.getString(8)),2) +
                         Math.pow(Double.parseDouble(res.getString(9)),2) +
                         Math.pow(Double.parseDouble(res.getString(10)),2));
          Elephant_Aji ele = new Elephant_Aji(vector);
          if(i < Math.round(tabel_alumni.getRowCount() * 0.6)){
             Mpop.add(ele);
              System.out.println("[Current Position] index -"+i + "nilai Male = "+ele.pos);
```

```
            }
          else{
              Fpop.add(ele);
              System.out.println("[Current Position] index -"+i + "nilai Female = "+ele.pos);
          }
           i++;
          }
      }
      catch(SQLException ex){
          System.out.print(ex.getMessage());
      }
 }
  public void setM_random(int m_random) {
      this.m_random = m_random;
  }

  public double getM_random() {
      return m_random;
  }

  public void setM_eval(double m_eval) {
      if(this.m_eval < m_eval){
          this.m_eval = m_eval; //buat kondisi m_eval itu menjadi penampung nilai max dari newsol
      }
  }

  public double getM_eval() {
      return m_eval;
  }

  public int getInitRandomPop(int PopSizeMale, int PopSizeFemale, int iterasi) {

      int res = iterasi;
      double newsol;//[] = null;// new double [data.length]; // posisi baru
          double newsol2;//[] = null;

        for(int i=0; i< PopSizeMale; i++)
        {
            getDimension();
            newsol = getInitRandom(Mpop.get(i).pos); //Check lagi //menampung posisi acak baru
            setM_eval(newsol); //ambil nilai max dari getinitrandom

            newsol2 = getM_eval();
            System.out.println("Gajah Jantan ke - "+i +"\n");
            System.out.println("newsol = "+newsol);
            System.out.println("newsol2 = "+newsol2);
            Elephant_Aji newE1 = new Elephant_Aji(0,newsol2,newsol,1); //Check lagi
            Elephant_Aji newE2 = new Elephant_Aji(0,newsol,newsol2,1); //Check lagi
            if(newsol > newsol2){
              Mpop.set(i,newE1);
            }
            else{
              Mpop.set(i, newE2);
            }
            //Mpop.set(i,newE1);
            System.out.println("Mpop pos = "+Mpop.get(i).Value_Elephant);
        }
        double BestMale = Mpop.get(0).pos;
        double value = Mpop.get(0).Value_Elephant;

        for(int i =0 ; i< Mpop.size() ; i++) //Male
      {
          if(Mpop.get(i).pos > BestMale)
          {
```

```
            BestMale = Mpop.get(i).pos;
        }
        if(Mpop.get(i).Value_Elephant > value){
            value = Mpop.get(i).Value_Elephant;
        }
    }
     System.out.println("BEST POS MALE FIRST = "+BestMale);
     System.out.println("BEST VALUE MALE FIRST = "+value);

     //Fpop
     for(int i=0; i <PopSizeFemale ; i++)
     {
         Elephant_Aji newE1 = new Elephant_Aji(1,BestMale,value,1);
         Fpop.set(i,newE1);
         //System.out.println("Female Pos = "+Fpop.get(i).pos);
     }
    return res-1;
}

public double getInitRandom(double crom)
{
    double min=0, max=0; //MPOP
    int i;

    for(i =0; i< Mpop.size();i++){
        if(min > Mpop.get(i).pos){
            min = Mpop.get(i).pos;
        }
        if(max < Mpop.get(i).pos){
            max = Mpop.get(i).pos;
        }
    }
    crom = randreal(min,max);
    System.out.println("[get InitRandom] crom = "+crom);

    return crom;
}

public double Apply(int Male, int Female, int IterationReal)
{
    data = new Object[tabel_alumni.getRowCount()][9];

    int Itera = (int) (0.9 *IterationReal);
        double RadiusMale = getDimension() * 2;
    int total = (Male + Female);
    int IterBL = (int) ((int) Math.round(total * 0.4));
    LimitLife = Itera / IterBL / 10; //90/47/10 =0

    System.out.println("\n PARAMETER & INFORMASI DATA: \n");
    System.out.println("Male Apply = "+Male);
    System.out.println("Female Apply = "+Female);
    System.out.println("Total Apply = "+total);
    System.out.println("IterBL Apply ="+IterBL);
    System.out.println("Radius Male = "+RadiusMale);
    System.out.println("IterationReal = "+IterationReal);
    System.out.println("Itera = "+Itera);
    int Itera2 = (IterationReal - (IterationReal - Itera));
    System.out.println("Itera2 = "+Itera2);
    if(LimitLife == 0)
    {
        LimitLife = Itera2 / 10;
        System.out.println("if limit life");
        System.out.println("LimitLife = "+LimitLife);
    }
```

```
int BLmal =0;

System.out.println("sebelum MASUK WHILE");

while(Iterasi < IterationReal)
{
   System.out.println("AWAL MULA");
   for(int i=0; i<Mpop.size() ; i++){
      System.out.println("Index ke - "+i+ "Mpop ="+Mpop.get(i).pos);
   }
   for(int j=0; j<Fpop.size() ; j++){
      System.out.println("Index Fpop ke - "+j+ "Fpop ="+Fpop.get(j).pos);
   }

Searching(Mpop, Fpop);

for(int itM = 0; itM < Male ; itM++)
{
   boolean out = false;

for(int j=0 ; j<Male && !out; j++)
{
   if(j != itM && EuclideanDistance(Mpop,itM,j) < RadiusMale)
   {
      EscapeElephant(Mpop,j,itM,RadiusMale); //BELUM BENER
      //EscapeElephant(Mpop,itM,j,RadiusMale);
      out = true;
   }
}
   Mpop.set(itM, NewPosition(0, Mpop.get(itM), RadiusMale)); //BELUM BENER
   System.out.println("\nSetelah Escape dan Sebelum Update, Mpop Pos = "+Mpop.get(itM).pos);
}
   UpdateLocation(0,Mpop); //BELUM BENER
   System.out.println("\nSudah MASUK While dan sebelum Search Local Best\n");
   double ant = BestLocal.Value_Elephant; //leader.Value_Elephant
   Iterasi += Leader_LocalSearch(IterBL);
   double des = leader.Value_Elephant;

   SearchLocalBest(Fpop);

if(ant - des == 0)
{
   BLmal++;
}
else
{
   BLmal=0;
}

if(BLmal > 2 * getDimension())
{
   for(int i =0; i<Fpop.size(); i++)
   {
   Fpop.get(i).Value_Elephant = Value_MaleElephantBestGlobal;
   Fpop.get(i).pos = MaleElephantBestGlobal;
   }
   MaleElephantBestGlobal = MaleElephant.pos;
   Value_ElephantBestGlobal = MaleElephant.Value_Elephant;
   }

for(int itF = 0; itF < Female; itF++)
{
   if(leader != Fpop.get(itF))
```

```
        {
        Fpop.set(itF, NewPosition(1, Fpop.get(itF), MaleElephant.pos));
        }
    }
    UpdateLocation(1,Fpop);

    System.out.println("\nSudah melakukan Search Local Best\n");
    for(int j=0; j<Female ; j++)
    {
        if(ChangeElephant(Fpop.get(j)))
        {
            BabyElephant(Mpop, Fpop, j, 1);
        }
    }

    if(BestLocal.Value_Elephant < Value_ElephantBestGlobal)
    {
        BestGlobal = BestLocal.pos;
        Value_ElephantBestGlobal = BestLocal.Value_Elephant;
    }

    if(MaleElephant.Value_Elephant < Value_MaleElephantBestGlobal && MaleElephant.Value_Elephant
> leader.Value_Elephant)
    {
        MaleElephantBestGlobal = MaleElephant.pos;
        Value_MaleElephantBestGlobal = BestLocal.Value_Elephant;
    }
    ListBestLocal.add(BestLocal.Value_Elephant);
    Iterasi++;
    }

    System.out.println("\nOverall\n");
    for(int i =0; i<Mpop.size(); i++){
        System.out.println("[Apply] Index Mpop ke-"+i+ ", Mpop = "+Mpop.get(i).pos);
    }
    for(int i=0; i<Fpop.size(); i++){
        System.out.println("[Apply] Index Fpop ke-"+i+ ", Fpop = "+Fpop.get(i).pos);
    }

    Vector<Elephant_Aji> Bubble = new Vector<Elephant_Aji>();

    int Sort = Mpop.size() + Fpop.size();
    Elephant_Aji tmp = new Elephant_Aji();
    int no=0;
    for(int i = 0; i<Mpop.size() ; i++){
        Mpop.get(i).id = no;
        Bubble.add(Mpop.get(i));
        no++;
    }
    for(int j= 0; j<Fpop.size() ; j++){
        Fpop.get(j).id = no;
        Bubble.add(Fpop.get(j));
        no++;
    }

    for(int i = 0; i<Bubble.size(); i++){
        for(int j=i+1; j<(Bubble.size()); j++){ //for(int j=1; j<(Bubble.size()-1); j++){
            if(Bubble.get(i).pos < Bubble.get(j).pos){ //j-1 & j
                tmp.pos = Bubble.get(i).pos; //j-1
                Bubble.get(i).pos = Bubble.get(j).pos; //j-1 & j
                Bubble.get(j).pos = tmp.pos;

                tmp.id = Bubble.get(i).id; //j-1
                Bubble.get(i).id = Bubble.get(j).id; //j-1 & j
```

```
                Bubble.get(j).id = tmp.id;
            }
        }
    }

    System.out.println("\nSorting Bubble Sort ESA\n");
    for(int i=0; i<Bubble.size(); i++){
        System.out.println("ID Alumni ke- "+Bubble.get(i).id+ ", Pos = "+Bubble.get(i).pos);
    }

Object[][]data = new Object[tabel_alumni.getRowCount()][11];
    int i=0;
    try{
    Statement statement = (Statement) Connection_mysql.getConnection().createStatement();
        statement.executeUpdate("Delete from database_esa");
        res = statement.executeQuery("select *from normalisasi");
        while(res.next()){
String[] row = {res.getString(1),res.getString(2), res.getString(3), res.getString(4), res.getString(5), res.getString(6), res.getString(7), res.getString(8), res.getString(9), res.getString(10), res.getString(11)};
            data[i] = row;
            i++;
        }
            for(int a = 0 ; a< Bubble.size() * 0.4 ; a++){
             statement.executeUpdate("insert into database_esa VALUES ('"+data[Bubble.get(a).id][0]+"','"+data[Bubble.get(a).id][1]+"','"+data[Bubble.get(a).id][2]+"','"+data[Bubble.get(a).id][3]+"','"+data[Bubble.get(a).id][4]+"','"+data[Bubble.get(a).id][5]+"','"+data[Bubble.get(a).id][6]+"','"+data[Bubble.get(a).id][7]+"','"+data[Bubble.get(a).id][8]+"','"+data[Bubble.get(a).id][9]+"','"+data[Bubble.get(a).id][10]+"')");
            }
        statement.close();
        JOptionPane.showMessageDialog(null, "Database ESA Berhasil disimpan");
    } catch (SQLException ex) {
        JOptionPane.showMessageDialog(null, "Database ESA Gagal disimpan"+ex);
    }
    System.out.println("Best Global = "+BestGlobal); //mengambil nilai crom index ke-1
    return BestGlobal;
}

  public void Searching(Vector<Elephant_Aji> Mpop, Vector<Elephant_Aji> Fpop)
  {
    MaleElephant = Mpop.get(0);
    BestLocal = Mpop.get(0);
    leader = Fpop.get(0);
    Iterasi = 0;
    System.out.println("(1) MaleElephant = "+Mpop.get(0).pos);
    System.out.println("(2) Best Local pos = "+BestLocal.pos);
    System.out.println("(3) Leader = "+leader.pos);
    for(int i =0 ; i< Mpop.size() ; i++) //Male
    {
        if(Mpop.get(i).pos < MaleElephant.pos) //pake value_elephant
        {
            MaleElephant.pos = Mpop.get(i).pos; //dk pake pos atau value_elephant
        }
        System.out.println("[searching - Male] Index ke-"+i +"Male Elephant = "+MaleElephant.pos);
    }
    BestLocal = MaleElephant;
    MaleElephantBestGlobal = MaleElephant.pos;
    Value_MaleElephantBestGlobal = MaleElephant.pos;

    for(int i=0; i< Fpop.size(); i++) //Female
    {
        if(Fpop.get(i).pos < leader.pos) //pake value_ele
        {
            leader.pos = Fpop.get(i).pos; //tanpa value_ele
```

```
            }
            //System.out.println("[searching - Female] Index ke-"+i+ "Leader Elephant = "+leader.pos);
            if(Fpop.get(i).pos < BestLocal.pos)
            {
                BestLocal.pos = Fpop.get(i).pos;
            }
        }

        BestGlobal = BestLocal.pos;
        System.out.println("[searching] BestGlobal = "+BestGlobal);
    }

    public void EscapeElephant(Vector<Elephant_Aji> Mpop, int i, int j, double radiusMaleElephant)
//Mengambil nilai min maybe
    {
        int better, worst;
        int min = 0, max = 0;
        System.out.println("Mpop EE i = "+Mpop.get(i).pos);
        System.out.println("Mpop EE j = "+Mpop.get(j).pos);

        if(Mpop.get(i).pos > Mpop.get(j).pos){ //pake value
            better = i; // besak
            worst = j; // kecik
        }
        else{
            better = j;
            worst = i;
        }
            //PAKE INI!
            Mpop.get(worst).pos = Mpop.get(worst).pos + LevyValue(1.7);

    }

    public void SearchLocalBest(Vector<Elephant_Aji> Fpop) //Atau fungsi menentukan gbest
    {
        if(MaleElephant.Value_Elephant < leader.Value_Elephant)
        {
            for(int i=0; i<Fpop.size(); i++)
            {
                Fpop.get(i).Value_Elephant = MaleElephant.Value_Elephant;
                Fpop.get(i).pos = MaleElephant.pos;
            }
        }
    }

    public Elephant_Aji NewPosition(int sex, Elephant_Aji Elephant, double RadiusMale)
    {
        double NewElephant[]= new double [Mpop.size() + Fpop.size()];
        double Ant_Elephant = Elephant.pos;//[] = new double [Mpop.size()];
        double Leader_Elephant = leader.pos;
        int min = 0, max = 0;
        Elephant_Aji res = new Elephant_Aji(sex,Elephant.pos,Elephant.pos, 1);
        Iterasi ++;
        return res;
    }

    public double EuclideanDistance(Vector<Elephant_Aji> pop, int First, int Second)
    {
        double sum = 0;
        for(int i=0; i<pop.get(First).pos; i++) //tidak menggunakan size dari pos
        {
            sum += (Math.pow((pop.get(First).pos - pop.get(Second).pos),2) * Math.pow((pop.get(First).pos -
pop.get(Second).pos),2)); //tidak menggunakan array[i]
```

```
        }
        System.out.println("Euclidean Distance = "+Math.sqrt(sum));
          return Math.sqrt(sum);
    }

        public void UpdateLocation(int sex, Vector<Elephant_Aji> pop){ //ME = Male Elephant, FE = Female
Elephant

        //Male Elephant
        if(sex == 0){
            BestLocal = pop.get(0);

            for(int j=0; j < pop.size(); j++) //mpop.size
            {
                if(pop.get(j).Value_Elephant < BestLocal.Value_Elephant)
                {
                    BestLocal = pop.get(j);
                    //System.out.println("[update location] BestLocal = "+BestLocal.pos);
                }
            }
            //MaleElephant = BestLocal;
            MaleElephant.pos = BestLocal.pos;
            System.out.println("[update location] Male Elephant = "+MaleElephant.pos);
        }
        //Female Elephant
        else{
            leader = pop.get(0);

            for(int j =0 ; j<pop.size(); j ++)
            {
                if(pop.get(j).Value_Elephant < BestLocal.Value_Elephant)
                {
                    BestLocal = pop.get(j);
                }
                if(pop.get(j).Value_Elephant < leader.Value_Elephant)
                {
                    leader = pop.get(j);
                }
            }
        }
    }

    public double Leader_LocalSearch (int Iter){
        double min=0, max=0;
        double evals=0;

        for(int i =0; i< Mpop.size();i++){
            if(min > Mpop.get(i).pos){
                min = Mpop.get(i).pos;
            }
            if(max < Mpop.get(i).pos){
                max = Mpop.get(i).pos;
            }
       }

        for(int i = 0; i < Fpop.size(); i++){
            if(leader.pos > max)
                {
                    leader.pos = max;
                }
                else if(leader.pos < min)
                {
                    leader.pos = min;
                }
```

```
        }

        evals = leader.pos;
        System.out.println("leader pos = "+evals);
        return evals;
    }

    public final int randint(int low, int high)
    {
            double random;

            random = rand1();
            return ((int)(low + (high - low + 1) * random));
    }
    public final double rand1() //Mengembalikan Fungsi Random
    {
            //return rand1();
        return Math.random();
    }

    public final double randreal(double low, double high) //untuk male elephant (jangkauan pencarian gajah
jantan)
    {
            return (low + (high - low) * rand1());
    }

    public boolean ChangeElephant(Elephant_Aji life) //fungsi keputusan untuk melakukan pergantian gajah
            {
                    if (randreal(0,1) > Math.pow(life.Life_Elephant / LimitLife,3))
                    {
                            return false;
                    }
                    return true;
            }

    public int getDimension()    {
        return m_domain; //menampung nilai dimensi
    }

    public void BabyElephant(Vector<Elephant_Aji> Mpop, Vector<Elephant_Aji> Fpop, int pos, int sex)
    {
      double Baby[] = null;
      double Father[] = null;
      double Mother[] = null;
      double Leader[] = null;

      if(sex == 0)
      { //Male
         m_random = (randint(0,Fpop.size()-1));
         Mother[0] =  Fpop.get(m_random).pos;
         Father[0] =  Mpop.get(pos).pos;
         Leader[0] =  leader.pos;

         for(int i=0 ; i< Father.length ; i++){ //tidak memanggil size dari si father
            Baby[i] = (0.6 * Mother[i] + 0.3 * Father[i] + 0.1 * Leader[i]) ;

            Mpop.get(pos).pos = Baby[i]; //Check lagi ini //Baby[pos]
            Mpop.get(pos).Life_Elephant = 1;
            Mpop.get(pos).Gender_Elephant = sex;
            Mpop.get(pos).Value_Elephant = Baby[i];           }

         if(Mpop.get(pos).Value_Elephant < MaleElephant.Value_Elephant){
            MaleElephant = Mpop.get(pos);
         }
```

```
        if(Mpop.get(pos).Value_Elephant < BestLocal.Value_Elephant){
            BestLocal = Mpop.get(pos);
        }
    }
    else
    {//Female
        Mother[0] = Fpop.get(pos).pos;
        int m = randint(0,Mpop.size()-1);
        Father[0] = Mpop.get(m).pos;
        Leader[0] = leader.pos;

        for(int i=0; i<Mother.length; i++)
        {
            double Value = 0.6 * Mother[i] + 0.3 * Father[i] + 0.1 * Leader[i];
            Baby[i] = (Value);

            Fpop.get(pos).pos = Baby[i]; // Baby[pos]
            Fpop.get(pos).Value_Elephant = 1;
            Fpop.get(pos).Gender_Elephant = sex;
            Fpop.get(pos).Value_Elephant = Baby[i]; //Baby[pos]
        }

        if(Fpop.get(pos).Value_Elephant < leader.Value_Elephant)
        {
            leader = Fpop.get(pos);
        }

        if(Fpop.get(pos).Value_Elephant < BestLocal.Value_Elephant)
        {
            BestLocal = Fpop.get(pos);
        }
    }
    Iterasi++;
  }

  public double LevyValue(double L){
      System.out.println("\nLEVY VALUE\n");
      double X = randreal(0,Math.PI / 2.0);//((-Math.PI / 2.0), (Math.PI / 2.0));
      System.out.println("X = "+X);
      double value;

      do{
          value = randreal(0,1);
          System.out.println("value = "+value);
      } while (value == 0);

      double Y = (-Math.log(value));
      System.out.println("Y = "+Y);
      double alpha = L - 1.0;
      System.out.println("alpha = "+alpha);

      System.out.println("1. sin = "+Math.sin(alpha * X));
      System.out.println("2. cos_1 = "+Math.pow(Math.cos(X), 1.0/alpha));
      System.out.println("3. cos_2 = "+Math.pow((Math.cos(1.0 - alpha) * X)/Y , (1.0-alpha)/alpha));
      double Z = (((Math.sin(alpha * X) / Math.pow(Math.cos(X), 1.0/alpha))) * (Math.pow((Math.cos(1.0 -
alpha) * X) / Y, (1.0-alpha)/alpha)));
      System.out.println("nilai levy value = "+Z);
      return Z;
  }
}
```

Elephant_Aji.java

```
package prediksimasastudi;

import java.util.*;
import java.text.*;
import javax.swing.JTable;
/**
  * @author M. Syukron Azim
 */
public class Elephant_Aji {
    public JTable tabel_alumni;
    Object[][] data;
    public int Gender_Elephant; // Male 0, Female 1
    public int Life_Elephant;
    public double Value_Elephant;
    public double pos; //Coba buat satu buah pos array -> public double pos[];
    public int id;
    public Elephant_Aji(int s, double[] v, double[] fitness, int vid) {
        Gender_Elephant = s;
        Life_Elephant = vid;
        pos = v[0];
        Value_Elephant = fitness[0];
        copyFrom(pos);
    }

    public Elephant_Aji(int s, double v, double fitness, int vid) {
        Gender_Elephant = s;
        Life_Elephant = vid;
        pos = v;
        Value_Elephant = fitness;
        copyFrom(pos);
    }
     public Elephant_Aji (){
        Gender_Elephant =0;
        Life_Elephant =0;
        Value_Elephant = 9999999;
    }
     public Elephant_Aji(double vector) {
           copyFrom(vector);
     }
    private void copyFrom(double v){
        pos = v;
    }
}
```

Lampiran 2. Hasil Pengujian Validasi Data Latih

Tabel L-1. Hasil Pengujian Menggunakan *Backpropogation Algorithm*

K1	K2	K3	K4	K5	K6	K7	K8	K9	K10	Target	Output	Output Klasifikasi	Klasifikasi
almn_0001	0.30	0.55	0.10	0.10	0.10	0.10	0.10	0.10	0.76	1	0.7376335137448238	1	Sesuai
almn_0002	0.43	0.55	0.21	0.10	0.10	0.10	0.18	0.10	0.23	1	0.5667050850229279	1	Sesuai
almn_0003	0.43	0.42	0.21	0.10	0.10	0.10	0.18	0.10	0.90	1	0.598237176136539	1	Sesuai
almn_0004	0.36	0.61	0.32	0.36	0.10	0.30	0.10	0.30	0.36	1	0.7016025301401747	1	Sesuai
almn_0005	0.43	0.55	0.21	0.10	0.10	0.30	0.18	0.30	0.36	1	0.7449402776556112	1	Sesuai
almn_0006	0.43	0.49	0.10	0.10	0.10	0.10	0.10	0.50	0.10	1	0.5905800276997738	1	Sesuai
almn_0007	0.63	0.49	0.10	0.10	0.10	0.10	0.10	0.10	0.10	1	0.6581796974263048	1	Sesuai
almn_0008	0.23	0.60	0.32	0.10	0.36	0.50	0.10	0.50	0.10	1	0.5578556920658804	1	Sesuai

almn_0009	0.50	0.49	0.21	0.10	0.10	0.30	0.10	0.30	0.36	1	0.70460511117401	1	Sesuai
almn_0010	0.16	0.49	0.21	0.10	0.10	0.50	0.10	0.30	0.10	1	0.7741363336790326	1	Sesuai
almn_0011	0.43	0.67	0.10	0.10	0.10	0.30	0.18	0.50	0.10	1	0.6632805188293611	1	Sesuai
almn_0012	0.50	0.49	0.32	0.10	0.36	0.30	0.18	0.50	0.36	1	0.41997471957521787	0	Tidak Sesuai
almn_0013	0.30	0.55	0.21	0.10	0.10	0.10	0.18	0.10	0.23	1	0.5804668312772125	1	Sesuai
almn_0014	0.30	0.53	0.10	0.10	0.10	0.10	0.10	0.10	0.63	1	0.6085628107986017	1	Sesuai
almn_0015	0.30	0.49	0.21	0.10	0.10	0.10	0.18	0.10	0.23	1	0.7581564354654056	1	Sesuai
almn_0016	0.30	0.50	0.32	0.10	0.10	0.50	0.10	0.30	0.36	1	0.4975636339747742	0	Tidak Sesuai
almn_0017	0.36	0.48	0.21	0.10	0.10	0.30	0.10	0.10	0.36	1	0.7571463987549691	1	Sesuai
almn_0018	0.30	0.55	0.10	0.10	0.10	0.10	0.10	0.10	0.50	1	0.6079595753191527	1	Sesuai
almn_0019	0.30	0.49	0.21	0.10	0.36	0.30	0.18	0.30	0.10	1	0.5226239841824509	1	Sesuai
almn_0020	0.36	0.63	0.10	0.10	0.10	0.10	0.10	0.10	0.36	1	0.6142256684908928	1	Sesuai
almn_0021	0.30	0.45	0.32	0.10	0.36	0.10	0.27	0.30	0.63	1	0.3237594316658106	0	Tidak Sesuai

almn_0022	0.36	0.67	0.32	0.10	0.10	0.30	0.27	0.30	0.36	1	0.2528012402329995	0	Tidak Sesuai
almn_0023	0.50	0.61	0.21	0.10	0.36	0.30	0.18	0.30	0.10	1	0.5138789701194301	1	Sesuai
almn_0024	0.36	0.51	0.67	0.63	0.36	0.50	0.54	0.50	0.23	0	0.32310277712462926	0	Sesuai
almn_0025	0.43	0.42	0.55	0.1	0.36	0.5	0.45	0.50	0.36	0	0.4776530697296644	0	Sesuai
almn_0026	0.23	0.51	0.44	0.10	0.10	0.50	0.36	0.30	0.10	0	0.18900893828949344	0	Sesuai
almn_0027	0.56	0.61	0.21	0.10	0.10	0.30	0.10	0.10	0.23	0	0.28084155950533557	0	Sesuai
almn_0028	0.16	0.45	0.32	0.10	0.10	0.50	0.27	0.50	0.10	0	0.28874876647622383	0	Sesuai
almn_0029	0.63	0.48	0.44	0.10	0.36	0.30	0.18	0.30	0.36	0	0.43149157958147344	0	Sesuai
almn_0030	0.30	0.73	0.21	0.10	0.36	0.50	0.27	0.50	0.36	0	0.2479889935998034	0	Sesuai

Pada Tabel L-1 adalah hasil pengujian model pelatihan menggunakan algoritma *Backpropogation*. Pada proses pengujian ini dilakukan untuk mengukur tingkat akurasi dengan menggunakan sampel data alumni sebagai *input* datanya. Didapatkan jumlah klasifikasi yang sesuai sebanyak 26 data dari 30 data. Sehingga, mendapatkan tingkat akurasi sebesar 86.67% seperti perhitungan tingkat akurasi dibawah ini.

$$\text{Akurasi} = \frac{Jumlah\ Data\ Klasifikasi\ Sesuai}{\sum Data}\ X\ 100$$

$$\text{Akurasi} = \frac{26}{30}\ X\ 100 = \ 86{,}67\%$$

Tabel L-2. Hasil Pengujian Menggunakan *Elephant Search Algorithm – Backpropogation Algorithm*

K1	K2	K3	K4	K5	K6	K7	K8	K9	K10	Target	Output	Output Klasifikasi	Klasifikasi
almn_0001	0.30	0.55	0.10	0.10	0.10	0.10	0.10	0.10	0.76	1	0.7376335137448238	1	Sesuai
almn_0002	0.43	0.55	0.21	0.10	0.10	0.10	0.18	0.10	0.23	1	0.5667050850229279	1	Sesuai
almn_0004	0.36	0.61	0.32	0.36	0.10	0.30	0.10	0.30	0.36	1	0.7016025301401747	1	Sesuai
almn_0005	0.43	0.55	0.21	0.10	0.10	0.30	0.18	0.30	0.36	1	0.7449402776556112	1	Sesuai
almn_0007	0.63	0.49	0.10	0.10	0.10	0.10	0.10	0.10	0.10	1	0.6581796974263048	1	Sesuai
almn_0008	0.23	0.60	0.32	0.10	0.36	0.50	0.10	0.50	0.10	1	0.5578556920658804	1	Sesuai
almn_0009	0.50	0.49	0.21	0.10	0.10	0.30	0.10	0.30	0.36	1	0.70460511117401	1	Sesuai
almn_0011	0.43	0.67	0.10	0.10	0.10	0.30	0.18	0.50	0.10	1	0.6632805188293611	1	Sesuai
almn_0012	0.50	0.49	0.32	0.10	0.36	0.30	0.18	0.50	0.36	1	0.5603904138934098	1	Sesuai
almn_0015	0.30	0.49	0.21	0.10	0.10	0.10	0.18	0.10	0.23	1	0.7581564354654056	1	Sesuai

almn_0016	0.30	0.50	0.32	0.10	0.10	0.50	0.10	0.30	0.36	1	0.4975636339747742	0	Tidak Sesuai
almn_0017	0.36	0.48	0.21	0.10	0.10	0.30	0.10	0.10	0.36	1	0.7571463987549691	1	Sesuai
almn_0020	0.36	0.63	0.10	0.10	0.10	0.10	0.10	0.10	0.36	1	0.6455534072612304	1	Sesuai
almn_0022	0.36	0.67	0.32	0.10	0.10	0.30	0.27	0.30	0.36	1	0.2528012402329995	0	Tidak Sesuai
almn_0023	0.50	0.61	0.21	0.10	0.36	0.30	0.18	0.30	0.10	1	0.5138789701194301	1	Sesuai
almn_0024	0.36	0.51	0.67	0.63	0.36	0.50	0.54	0.50	0.23	0	0.32310277712462926	0	Sesuai
almn_0025	0.43	0.42	0.55	0.1	0.36	0.5	0.45	0.50	0.36	0	0.4776530697296644	0	Sesuai
almn_0027	0.56	0.61	0.21	0.10	0.10	0.30	0.10	0.10	0.23	0	0.28084155950533557	0	Sesuai
almn_0029	0.63	0.48	0.44	0.10	0.36	0.30	0.18	0.30	0.36	0	0.43149157958147344	0	Sesuai
almn_0030	0.30	0.73	0.21	0.10	0.36	0.50	0.27	0.50	0.36	0	0.2479889935998034	0	Sesuai
almn_0033	0.23	0.59	0.33	0.10	0.37	0.50	0.28	0.50	0.23	1	0.7361783256884055	1	Sesuai
Almn_0035	0.23	0.70	0.10	0.10	0.10	0.30	0.10	0.30	0.37	1	0.5167360337167525	1	Sesuai
Almn_0038	0.23	0.56	0.21	0.10	0.10	0.10	0.10	0.10	0.23	1	0.725886076186253	1	Sesuai

Almn_0040	0.23	0.51	0.10	0.10	0.10	0.30	0.10	0.30	0.23	1	0.5918799669754017	1	Sesuai
Almn_0041	0.23	0.56	0.10	0.10	0.10	0.10	0.10	0.10	0.10	1	0.6461126311476407	1	Sesuai
Almn_0045	0.23	0.54	0.56	0.10	0.37	0.50	0.10	0.50	0.10	1	0.5328971047649419	1	Sesuai
Almn_0047	0.43	0.54	0.21	0.10	0.10	0.50	0.10	0.50	0.37	1	0.7141463475845767	1	Sesuai
Almn_0048	0.63	0.56	0.44	0.37	0.90	0.90	0.10	0.50	0.10	1	0.7365913625978139	1	Sesuai
Almn_0049	0.30	0.16	0.56	0.63	0.63	0.70	0.46	0.70	0.37	0	0.3926680916126861	0	Sesuai
Almn_0052	0.43	0.84	0.79	0.37	0.63	0.30	0.28	0.30	0.23	1	0.6358007505975923	1	Sesuai

Pada tabel L-2 adalah hasil pengujian model pelatihan menggunakan *Hybrid Elephant Search Algorithm* dan algoritma *Backpropogation*. Pada proses pengujian ini dilakukan untuk mengukur tingkat akurasi dengan menggunakan sampel data alumni sebagai *input* datanya. Didapatkan jumlah klasifikasi yang sesuai sebanyak 28 data dari 30 data. Sehingga, mendapatkan tingkat akurasi sebesar 93.33% seperti perhitungan tingkat akurasi dibawah ini.

$$\text{Akurasi} = \frac{Jumlah\ Data\ Klasifikasi\ Sesuai}{\sum Data}\ X\ 100$$

$$\text{Akurasi} = \frac{28}{30} x100 = 93.33\ \%$$

Lampiran 3. Hasil Pengujian Data Uji

Tabel L-3. Hasil Pengujian Prediksi Menggunakan Data Uji dengan *Backpropogation*

ID	K2	K3	K4	K5	K6	K7	K8	K9	K10	Output Prediksi	Output Pelatihan	Target Pelatihan	Klasifikasi
DU_001	0.15	0.49	0.10	0.10	0.10	0.10	0.10	0.50	0.10	0.9769982440584148	0.7741363336790326	1	Sesuai
DU_002	0.18	0.46	0.10	0.10	0.10	0.10	0.10	0.50	0.10	0.8932957730798111	0.7741363336790326	1	Sesuai
DU_003	0.66	0.69	0.37	0.10	0.10	0.10	0.26	0.50	0.10	0.852878057096135	0.7741363336790326	1	Sesuai
DU_004	0.84	0.45	0.90	0.90	0.50	0.26	0.26	0.90	0.90	0.37323064632897185	0.22583433717489185	0	Sesuai
DU_005	0.53	0.66	0.63	0.10	0.10	0.26	0.42	0.10	0.10	0.13041220095525097	0.2479889935998034	0	Sesuai
DU_006	0.15	0.80	0.37	0.10	0.10	0.10	0.26	0.10	0.10	0.15302828135267293	0.2479889935998034	0	Sesuai
DU_007	0.90	0.45	0.37	0.10	0.50	0.42	0.58	0.10	0.10	0.8164774391154481	0.7741363336790326	1	Sesuai
DU_008	0.27	0.45	0.63	0.10	0.10	0.26	0.26	0.10	0.10	0.9789568259189131	0.7741363336790326	1	Sesuai
DU_009	0.31	0.56	0.10	0.10	0.10	0.26	0.10	0.50	0.10	0.042104428170195794	0.2479889935998034	0	Sesuai
DU_010	0.52	0.80	0.63	0.10	0.10	0.26	0.26	0.10	0.10	0.2087592525858005	0.2479889935998034	0	Sesuai

DU_011	0.10	0.64	0.10	0.10	0.10	0.26	0.10	0.50	0.10	0.9907950247710812	0.7741363336790326	1	Sesuai
DU_012	0.19	0.90	0.10	0.10	0.10	0.10	0.58	0.10	0.10	0.016357296111291376	0.2479889935998034	0	Sesuai
DU_013	0.13	0.44	0.37	0.10	0.10	0.10	0.26	0.50	0.10	0.8809881233274804	0.7741363336790326	1	Sesuai
DU_014	0.81	0.66	0.37	0.90	0.10	0.42	0.10	0.10	0.10	0.7893107242655193	0.7741363336790326	1	Sesuai
DU_015	0.13	0.67	0.63	0.10	0.10	0.26	0.42	0.10	0.10	0.5563841722810314	0.5578556920658804	1	Sesuai
DU_016	0.17	0.38	0.10	0.90	0.50	0.10	0.10	0.50	0.10	0.5511163763917339	0.5578556920658804	1	Sesuai
DU_017	0.86	0.45	0.63	0.10	0.10	0.26	0.42	0.50	0.10	0.2673098051735971	0.2479889935998034	0	Sesuai
DU_018	0.13	0.52	0.63	0.10	0.10	0.10	0.42	0.10	0.10	0.17913208883838008	0.2479889935998034	0	Sesuai
DU_019	0.66	0.50	0.37	0.10	0.10	0.26	0.42	0.50	0.10	0.21820454614052234	0.2479889935998034	0	Sesuai
DU_020	0.61	0.69	0.37	0.10	0.10	0.42	0.26	0.10	0.10	0.507346849557253	0.5097044855889336	0	Tidak Sesuai
DU_021	0.62	0.61	0.37	0.10	0.10	0.10	0.26	0.10	0.10	0.37909386154679	0.32310277712462926	0	Sesuai
DU_022	0.11	0.44	0.37	0.90	0.10	0.26	0.10	0.50	0.10	0.949554214206018	0.7741363336790326	1	Sesuai
DU_023	0.13	0.46	0.90	0.10	0.10	0.10	0.26	0.10	0.10	0.9621677756747581	0.7741363336790326	1	Sesuai
DU_024	0.11	0.78	0.63	0.10	0.10	0.10	0.10	0.50	0.10	0.9443311242227027	0.7741363336790326	1	Sesuai

DU_025	0.26	0.10	0.37	0.10	0.10	0.26	0.42	0.50	0.10	0.7774298263523914	0.7741363336790326	1	Sesuai
DU_026	0.38	0.50	0.10	0.10	0.10	0.26	0.10	0.50	0.10	0.19866998314841472	0.2479889935998034	0	Sesuai
DU_027	0.30	0.38	0.63	0.90	0.10	0.10	0.10	0.10	0.10	0.8669875587062573	0.7741363336790326	1	Sesuai
DU_028	0.39	0.57	0.10	0.10	0.10	0.26	0.26	0.10	0.10	0.8765095852427776	0.7741363336790326	1	Sesuai
DU_029	0.68	0.63	0.63	0.90	0.10	0.26	0.10	0.10	0.10	0.48537289798920374	0.48009944978111563	0	Sesuai
DU_030	0.57	0.83	0.63	0.10	0.10	0.26	0.10	0.10	0.10	0.1490923746910949	0.2479889935998034	0	Sesuai

Tabel L-3 adalah hasil dari prediksi menggunakan data uji sebanyak 30 data mahasiswa yang masih aktif dalam menempuh program studi di perguruan tinggi dengan pengalaman minimum 1 kali organisasi, data tersebut dilakukan satu-persatu diinputkan kedalam fitur prediksi menggunakan perangkat lunak yang dikembangkan. Didapatkan hasil *confidence level* dibawah ini.

$$Confidence\ Level = \frac{Jumlah\ Data\ Klasifikasi\ Sesuai}{\sum Data}\ X\ 100$$

$$Confidence\ Level = \frac{29}{30} x100 = 96.67\ \%$$

Tabel L-4. Hasil Pengujian Prediksi Menggunakan Data Uji dengan *Elephant Search Algorithm* - *Backpropogation*

ID	K2	K3	K4	K5	K6	K7	K8	K9	K10	Output Prediksi	Output Pelatihan	Target Pelatihan	Klasifikasi
DU_001	0.15	0.49	0.10	0.10	0.10	0.10	0.10	0.50	0.10	0.9769982440584148	0.7581564354654056	1	Sesuai
DU_002	0.18	0.46	0.10	0.10	0.10	0.10	0.10	0.50	0.10	0.8932957730798111	0.7581564354654056	1	Sesuai
DU_003	0.66	0.69	0.37	0.10	0.10	0.10	0.26	0.50	0.10	0.852878057096135	0.7581564354654056	1	Sesuai
DU_004	0.84	0.45	0.90	0.90	0.50	0.26	0.26	0.90	0.90	0.57323064632897185	0.5918799669754017	1	Sesuai
DU_005	0.53	0.66	0.63	0.10	0.10	0.26	0.42	0.10	0.10	0.53041220095525097	0.22583433717489185	1	Sesuai
DU_006	0.15	0.80	0.37	0.10	0.10	0.10	0.26	0.10	0.10	0.15302828135267293	0.16703521160104748	0	Sesuai
DU_007	0.90	0.45	0.37	0.10	0.50	0.42	0.58	0.10	0.10	0.8164774391154481	0.7581564354654056	1	Sesuai
DU_008	0.27	0.45	0.63	0.10	0.10	0.26	0.26	0.10	0.10	0.9789568259189131	0.7581564354654056	1	Sesuai
DU_009	0.31	0.56	0.10	0.10	0.10	0.26	0.10	0.50	0.10	0.042104428170195794	0.2528012402329995	1	Tidak Sesuai

DU_010	0.52	0.80	0.63	0.10	0.10	0.26	0.26	0.10	0.10	0.5087592525858005	0.225834337174 89185	1	Sesuai
DU_011	0.10	0.64	0.10	0.10	0.10	0.26	0.10	0.50	0.10	0.9907950247710812	0.758156435465 4056	1	Sesuai
DU_012	0.19	0.90	0.10	0.10	0.10	0.10	0.58	0.10	0.10	0.016357296111291376	0.252801240232 9995	1	Tidak Sesuai
DU_013	0.13	0.44	0.37	0.10	0.10	0.10	0.26	0.50	0.10	0.8809881233274804	0.758156435465 4056	1	Sesuai
DU_014	0.81	0.66	0.37	0.90	0.10	0.42	0.10	0.10	0.10	0.7893107242655193	0.758156435465 4056	1	Sesuai
DU_015	0.13	0.67	0.63	0.10	0.10	0.26	0.42	0.10	0.10	0.5563841722810314	0.225834337174 89185	1	Sesuai
DU_016	0.17	0.38	0.10	0.90	0.50	0.10	0.10	0.50	0.10	0.5511163763917339	0.560390413893 4098	1	Sesuai
DU_017	0.86	0.45	0.63	0.10	0.10	0.26	0.42	0.50	0.10	0.5673098051735971	0.560390413893 4098	1	Sesuai
DU_018	0.13	0.52	0.63	0.10	0.10	0.10	0.42	0.10	0.10	0.17913208883838008	0.167035211601 04748	0	Sesuai
DU_019	0.66	0.50	0.37	0.10	0.10	0.26	0.42	0.50	0.10	0.51820454614052234	0.560390413893 4098	1	Sesuai
DU_020	0.61	0.69	0.37	0.10	0.10	0.42	0.26	0.10	0.10	0.507346849557253	0.560390413893 4098	1	Sesuai
DU_021	0.62	0.61	0.37	0.10	0.10	0.10	0.26	0.10	0.10	0.57909386154679	0.591879966975 4017	1	Sesuai
DU_022	0.11	0.44	0.37	0.90	0.10	0.26	0.10	0.50	0.10	0.949554214206018	0.758156435465 4056	1	Sesuai
DU_023	0.13	0.46	0.90	0.10	0.10	0.10	0.26	0.10	0.10	0.9621677756747581	0.758156435465 4056	1	Sesuai

DU_024	0.11	0.78	0.63	0.10	0.10	0.10	0.10	0.50	0.10	0.9443311242227027	0.7581564354654056	1	Sesuai
DU_025	0.26	0.10	0.37	0.10	0.10	0.26	0.42	0.50	0.10	0.7774298263523914	0.7581564354654056	1	Sesuai
DU_026	0.38	0.50	0.10	0.10	0.10	0.26	0.10	0.50	0.10	0.19866998314841472	0.16703521160104748	0	Sesuai
DU_027	0.30	0.38	0.63	0.90	0.10	0.10	0.10	0.10	0.10	0.8669875587062573	0.7581564354654056	1	Sesuai
DU_028	0.39	0.57	0.10	0.10	0.10	0.26	0.26	0.10	0.10	0.8765095852427776	0.7581564354654056	1	Sesuai
DU_029	0.68	0.63	0.63	0.90	0.10	0.26	0.10	0.10	0.10	0.58537289798920374	0.5918799669754017	1	Sesuai
DU_030	0.57	0.83	0.63	0.10	0.10	0.26	0.10	0.10	0.10	0.1490923746910949	0.2528012402329995	1	Tidak Sesuai

Tabel L-4 adalah hasil dari prediksi menggunakan data uji sebanyak 30 data mahasiswa yang masih aktif dalam menempuh program studi di perguruan tinggi dengan pengalaman minimum 1 kali organisasi, data tersebut dilakukan satu-persatu diinputkan kedalam fitur prediksi menggunakan perangkat lunak yang dikembangkan. Maka, didapatkan hasil *confidence level* seperti dibawah.

$$\textit{Confidence Level} = \frac{Jumlah\ Data\ Klasifikasi\ Sesuai}{\sum Data}\ X\ 100$$

$$\textit{Confidence Level} = \frac{27}{30}x100 = 90.00\ \%$$

Lampiran 4. Format Kuisioner

Kuisioner Mahasiswa Aktif Organisasi

Assalamualaikum Wr.Wb.
Perkenalkan saya Mukhammad Syukron Azim dari Program Studi Teknik Informatika, Fakultas Ilmu Komputer Universitas Sriwijaya angkatan 2015. Sehubungan dengan penyelesaian tugas akhir, saya meminta bantuan anda untuk mengisi kuisioner berikut ini. dimohon mengisi kuisioner berikut dengan sejujur-jujurnya, karena jawaban anda akan terjamin kerahasiaannya.
Atas kesediaan anda dalam mengisi kuisioner ini, saya ucapkan terima kasih sebesar-besarnya.
Semoga kebaikan Anda dibalas berlipat-berlipat ganda oleh Allah SWT.
Wassalamualaikum Wr.Wb.

Hormat saya,
peneliti

Mukhammad Syukron Azim

* Required

Email address *

Your email

NEXT

Never submit passwords through Google Forms.

Kuisioner Mahasiswa Aktif Organisasi

* Required

Untitled Section

Identitas Diri *
Nama Lengkap

Your answer

Tanggal Lahir
Date

mm/dd/yyyy

Tahun Berapa Anda Masuk Perguruan Tinggi? *

Your answer

Semester berapa Anda sekarang? *

Your answer

Tahun Berapa Anda Masuk Perguruan Tinggi? *

Your answer

Semester berapa Anda sekarang? *

Your answer

Jumlah SKS yang telah ditempuh selama menjadi mahasiswa aktif? *
Termasuk SKS yang sedang diambil sekarang

Your answer

IPK Terakhir Anda? *

Your answer

BACK NEXT

Never submit passwords through Google Forms.

Kuisioner Mahasiswa Aktif Organisasi

* Required

Kuisioner mengenai pengalaman organisasi selama menjadi mahasiswa aktif

Jumlah Organisasi yang Anda Ikuti selama menjadi mahasiswa aktif? *
apabila jumlah organisasi yang diambil lebih banyak dari pilihan yang ada, silahkan isi pada kolom other.

- ( ) 1
- ( ) 2
- ( ) 3
- (•) 4
- ( ) 5
- ( ) Other:

Pada semester berapa anda pertama kali mengikuti organisasi? *

	1	2	3	4	5	6	7	8
Semester	( )	( )	(•)	( )	( )	( )	( )	( )

Berapa lama anda mengikuti Organisasi? *

- ( ) 1 Tahun
- ( ) 2 Tahun
- (•) 3 Tahun
- ( ) 4 Tahun
- ( ) Other:

Jabatan didalam Organisasi yang Anda Ikuti?

	1 Kali	2 Kali	3 Kali	4 Kali	5 Kali
Ketua Umum	( )	( )	( )	( )	( )
Wakil Ketua Umum	( )	( )	( )	( )	( )
Kepala Dinas/Divisi	( )	(•)	( )	( )	( )
Staff	( )	( )	(•)	( )	( )
Anggota	( )	( )	( )	( )	( )

BACK SUBMIT

Never submit passwords through Google Forms.