Weka & Java

Tugas Kelas Mandiri IF4071

Semester I Tahun 2014 / 2015

Pembelajaran Mesin

oleh :

Iskandar Setiadi / 13511073

Sekolah Teknik Elektro dan Informatika (STEI ITB)

Institut Teknologi Bandung

Jl. Ganesha No. 10, Bandung 40132

Tahun 2014

	Sekolah Teknik Elektro dan Informatika ITB	**-**	**Halaman**
		IF4071-TI1	*18*
			18 September 2014

I. Bagian Utama (Main)

Kode berikut adalah bagian utama yang digunakan untuk mensimulasikan program:

Main.java

```
package core;

import filter.SupervisedFilter;
import helper.Constants;
import helper.FileHelper;
import helper.TextWriter;

import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.util.Enumeration;

import classifier.ClassifyAlgorithm;
import classifier.CustomAlgorithm;
import loader.LoadARFF;
import loader.LoadCSV;
import weka.classifiers.Classifier;
import weka.classifiers.Evaluation;
import weka.core.Attribute;
import weka.core.Instance;
import weka.core.Instances;

/**
 * Main class for Weka
 *
 * @author Iskandar Setiadi
 * @version 0.1, by IS @since September 16, 2014
 *
 */

public class Main {

    @SuppressWarnings("unchecked")
    public static void main(String[] args) throws Exception {

        Instances data = null;
        Classifier cModel = null;
        String input, input2;
        boolean isNominal = true;

        BufferedReader reader = new BufferedReader(new
InputStreamReader(System.in));
        while (true) {
            TextWriter.printMainMenu();
            input = reader.readLine();

            switch (input) {
            case "1":
                /** Load from .arff */
                TextWriter.printLoadMenu();
                input2 = reader.readLine();
                if (input2.equals("1")) {
                    data =
LoadARFF.loadARFF(Constants.ARFF_NOMINAL_PATH);
                } else if (input2.equals("2")) {
                    data =
LoadARFF.loadARFF(Constants.ARFF_NUMERIC_PATH);
                    isNominal = false;
                }
                // Set play {yes, no}
                data.setClassIndex(data.numAttributes() - 1);
                break;
            case "2":
                /** Load from .csv */
                TextWriter.printLoadMenu();
                input2 = reader.readLine();
                if (input2.equals("1")) {
                    data = LoadCSV.loadCSV(Constants.CSV_NOMINAL_PATH);
                } else if (input2.equals("2")) {
                    data = LoadCSV.loadCSV(Constants.CSV_NUMERIC_PATH);
                    isNominal = false;
                }
                // Set play {yes, no}
                data.setClassIndex(data.numAttributes() - 1);
                break;
            case "3":
                /** Remove attribute (outlook) */
                if (data != null) {
                    Enumeration<Attribute> e;

                    e = data.enumerateAttributes();
                    TextWriter.printEnumerationAttribute(e);
                    // Delete first attribute - Outlook
                    data.deleteAttributeAt(0);

                    e = data.enumerateAttributes();
                    TextWriter.printEnumerationAttribute(e);
                } else {
                    System.out.println("You need to load your data
first!");
                }
                break;
            case "4":
                /** Filter (resample) */
                if (data != null) {
                    System.out.println("# Previous : " +
data.numInstances());
                    data = SupervisedFilter.resampleInstances(data);
                    System.out.println("# After : " +
data.numInstances());
                } else {
                    System.out.println("You need to load your data
first!");
                }
                break;
            case "5":
                /** Build classifier with Naive Bayes */
                if (data != null) {
                    TextWriter.printClassifierMenu();
                    input2 = reader.readLine();
                    if (input2.equals("1")) {
                        cModel =
ClassifyAlgorithm.naiveBayesAlgorithm(data, 1);
                    } else if (input2.equals("2")) {
                        cModel =
ClassifyAlgorithm.naiveBayesAlgorithm(data, 2);
```

```
                        }
                    } else {
                        System.out.println("You need to load your data
first!");
                    }
                    break;
                case "6":
                    /** Build classifier with DT */
                    if (data != null) {
                        TextWriter.printClassifierMenu();
                        input2 = reader.readLine();
                        if (input2.equals("1")) {
                            cModel = ClassifyAlgorithm.iD3Algorithm(data, 1);
                        } else if (input2.equals("2")) {
                            cModel = ClassifyAlgorithm.iD3Algorithm(data, 2);
                        }
                    } else {
                        System.out.println("You need to load your data
first!");
                    }
                    break;
                case "7":
                    /** Testing model given test set (Assume train = test) */
                    if (cModel != null) {
                        Evaluation eval = new Evaluation(data);
                        eval.evaluateModel(cModel, data);
                        System.out.println(eval.toSummaryString(
                                "\nResults\n======\n", false));
                    } else {
                        System.out.println("You need to build classifier
first!");
                    }
                    break;
                case "8":
                    /** Testing model to classify one unseen data */
                    if (cModel != null) {
                        Instance test = new Instance(5);
                        if (isNominal) {
                            test.setValue(data.attribute(0), "sunny");
                            test.setValue(data.attribute(1), "mild");
                            test.setValue(data.attribute(2), "high");
                            test.setValue(data.attribute(3), "FALSE");
                        } else {
                            test.setValue(data.attribute(0), "rainy");
                            test.setValue(data.attribute(1), 65);
                            test.setValue(data.attribute(2), 70);
                            test.setValue(data.attribute(3), "TRUE");
                        }
                        // Give access to dataset
                        test.setDataset(data);

                        System.out.print("Classifying result: ");

System.out.println(data.attribute(data.numAttributes() - 1).
                                value((int) cModel.classifyInstance(test)));
                    } else {
                        System.out.println("You need to build classifier
first!");
                    }
                    break;
                case "9":
```

```
                /** Save model */
                if (cModel != null) {
                    FileHelper.saveModel(cModel,
Constants.SAVE_MODEL_PATH);
                } else {
                    System.out.println("You need to build classifier
first!");
                }
                break;
            case "10":
                /** Load model */
                cModel = FileHelper.loadModel(Constants.SAVE_MODEL_PATH);
                break;
            case "11":
                /** Create an extended classifier */
                if (data != null) {
                    cModel = new CustomAlgorithm();
                    cModel.buildClassifier(data);

                    // Test to classify data1
                    Evaluation eval = new Evaluation(data);
                    eval.evaluateModel(cModel, data);
                    System.out.println(eval.toSummaryString(
                            "\nResults\n======\n", false));
                } else {
                    System.out.println("You need to load your data
first!");
                }
                break;
            case "999":
                System.out.println("Goodbye!");
                return;
            default:
                System.out.println("Unrecognized input value!");
            }
        }

    }

}
```

Constants.java

```
package helper;

/**
 * Constants file for Application
 *
 * @author Iskandar Setiadi
 * @version 0.1, by IS @since September 16, 2014
 *
 */

public class Constants {
    public static String ARFF_NOMINAL_PATH =
"./data/weather.nominal.arff";
    public static String ARFF_NUMERIC_PATH =
"./data/weather.numeric.arff";
    public static String CSV_NOMINAL_PATH = "./data/weather.nominal.csv";
    public static String CSV_NUMERIC_PATH = "./data/weather.numeric.csv";
    public static String SAVE_MODEL_PATH = "./data/weather.model";
}
```

TextWriter.java

```
package helper;

import java.util.Enumeration;

import weka.core.Attribute;

/**
 * Text Writer for Application
 *
 * @author Iskandar Setiadi
 * @version 0.1, by IS @since September 16, 2014
 *
 */

public class TextWriter {

    /**
     * Print Main Menu
     */
    public static void printMainMenu() {
        /** Menu settings */
        System.out.println("-- Menu -- ");
        System.out.println("1 - Test load from .arff");
        System.out.println("2 - Test load from .csv");
        System.out.println("3 - Test remove attribute (outlook)");
        System.out.println("4 - Filter (resample)");
        /**
         * In no 5 & 6, you can choose between 10-fold cross validation
         * or percentage split
         * */
        System.out.println("5 - Build classifier with Naive Bayes");
        System.out.println("6 - Build classifier with DT");
        System.out.println("7 - Testing model given test set");
        System.out.println("8 - Testing model to classify one unseen
data");
        System.out.println("9 - Save model");
        System.out.println("10 - Load model");
        System.out.println("11 - Create an extended Classifier");
        System.out.println("999 - Exit");
        System.out.print("Input: ");
    }

    /**
     * Print Load Menu
     */
    public static void printLoadMenu() {
        System.out.println("-- Load --");
        System.out.println("1 - Nominal data");
        System.out.println("2 - Numeric data");
        System.out.print("Input: ");
    }

    /**
```

```
     * Print Classifier Menu
     */
    public static void printClassifierMenu() {
        System.out.println("-- Classifier --");
        System.out.println("1 - 10-fold cross-validation");
        System.out.println("2 - Percentage Split (50%)");
        System.out.print("Input: ");
    }

    /**
     * Enumerate through Attributes
     * @param e
     */
    public static void printEnumerationAttribute(Enumeration<Attribute>
e) {
        System.out.println("-- List of Attributes --");
        while (e.hasMoreElements()) {
            Attribute element = e.nextElement();
            System.out.println(element);
        }
    }

}
```

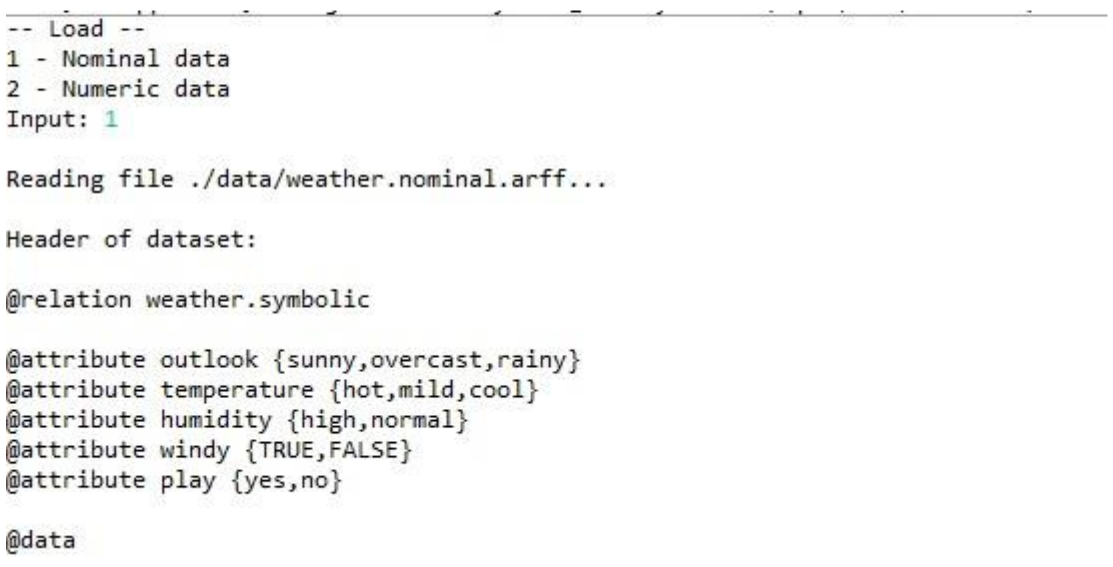
Detail dari masing-masing *method* akan dijabarkan pada bagian-bagian dibawah ini.

II. Load Data .arff pada Java

LoadArff.java

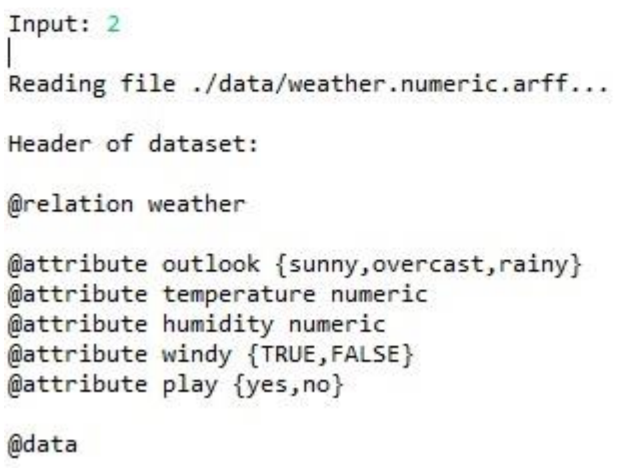
```
public class LoadARFF {

    public static Instances loadARFF(String path) throws Exception {
        Instances data = null;

        System.out.println("\nReading file " + path + "...");
        ArffLoader loader = new ArffLoader();
        if (path.startsWith("http:") || path.startsWith("ftp:"))
          loader.setURL(path);
        else
          loader.setSource(new File(path));
        data = loader.getDataSet();

        System.out.println("\nHeader of dataset:\n");
        System.out.println(new Instances(data, 0));

        return data;
    }

}
```

Hasil Eksekusi (Nominal):



```
-- Load --
1 - Nominal data
2 - Numeric data
Input: 1

Reading file ./data/weather.nominal.arff...

Header of dataset:

@relation weather.symbolic

@attribute outlook {sunny,overcast,rainy}
@attribute temperature {hot,mild,cool}
@attribute humidity {high,normal}
@attribute windy {TRUE,FALSE}
@attribute play {yes,no}

@data
```

Hasil Eksekusi (Numerik):



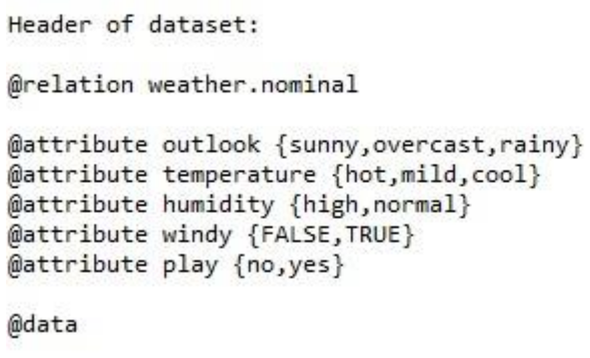
```
Input: 2

Reading file ./data/weather.numeric.arff...

Header of dataset:

@relation weather

@attribute outlook {sunny,overcast,rainy}
@attribute temperature numeric
@attribute humidity numeric
@attribute windy {TRUE,FALSE}
@attribute play {yes,no}

@data
```

III. Load Data .csv pada Java

LoadCSV.java

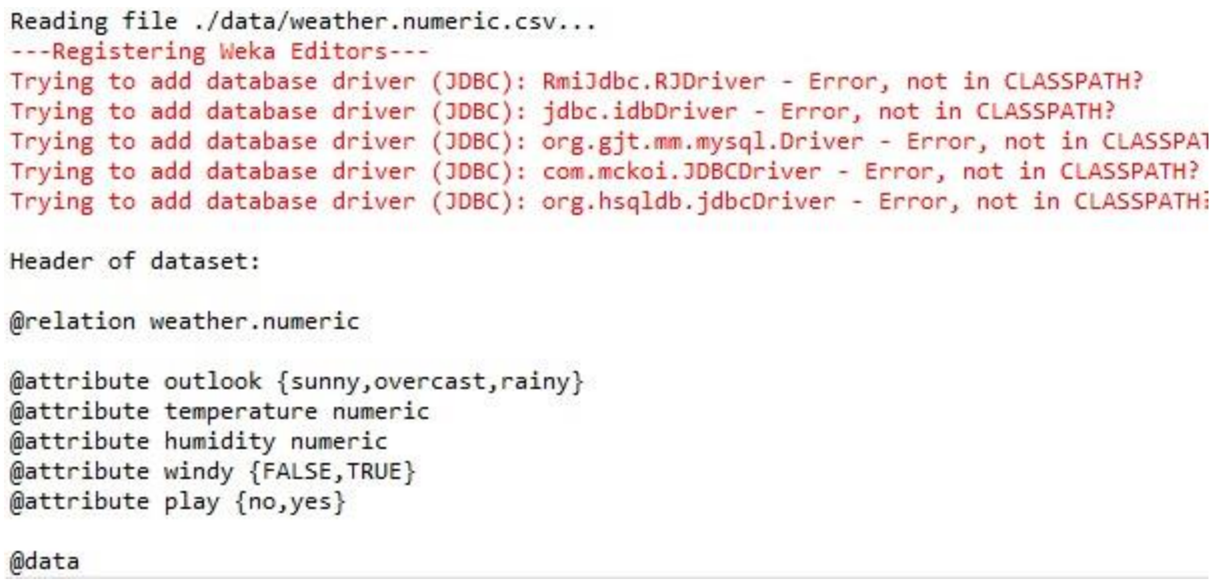
```java
public class LoadCSV {

    public static Instances loadCSV(String path) throws Exception {
        Instances data = null;

        System.out.println("\nReading file " + path + "...");
        CSVLoader loader = new CSVLoader();
        loader.setSource(new File(path));
        data = loader.getDataSet();

        System.out.println("\nHeader of dataset:\n");
        System.out.println(new Instances(data, 0));

        return data;
    }
}
```

Hasil Eksekusi (Nominal):



```
Header of dataset:

@relation weather.nominal

@attribute outlook {sunny,overcast,rainy}
@attribute temperature {hot,mild,cool}
@attribute humidity {high,normal}
@attribute windy {FALSE,TRUE}
@attribute play {no,yes}

@data
```

Hasil Eksekusi (Numerik):



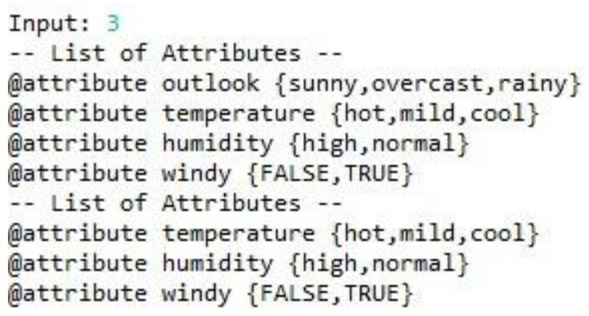
```
Reading file ./data/weather.numeric.csv...
---Registering Weka Editors---
Trying to add database driver (JDBC): RmiJdbc.RJDriver - Error, not in CLASSPATH?
Trying to add database driver (JDBC): jdbc.idbDriver - Error, not in CLASSPATH?
Trying to add database driver (JDBC): org.gjt.mm.mysql.Driver - Error, not in CLASSPAT
Trying to add database driver (JDBC): com.mckoi.JDBCDriver - Error, not in CLASSPATH?
Trying to add database driver (JDBC): org.hsqldb.jdbcDriver - Error, not in CLASSPATH

Header of dataset:

@relation weather.numeric

@attribute outlook {sunny,overcast,rainy}
@attribute temperature numeric
@attribute humidity numeric
@attribute windy {FALSE,TRUE}
@attribute play {no,yes}

@data
```

IV. Remove Attribute pada Java

Bagian dari **Main.java**

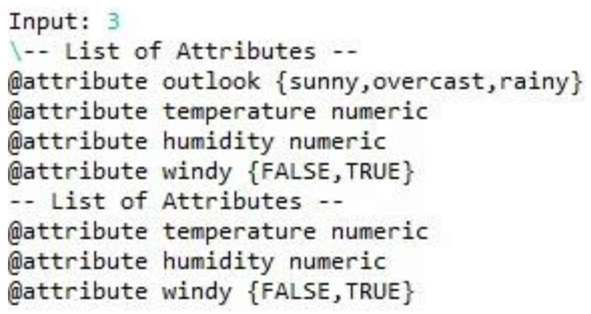
```
    // Delete first attribute - Outlook
    data.deleteAttributeAt(0);
```

Hasil Eksekusi (Nominal):



```
Input: 3
-- List of Attributes --
@attribute outlook {sunny,overcast,rainy}
@attribute temperature {hot,mild,cool}
@attribute humidity {high,normal}
@attribute windy {FALSE,TRUE}
-- List of Attributes --
@attribute temperature {hot,mild,cool}
@attribute humidity {high,normal}
@attribute windy {FALSE,TRUE}
```

Hasil Eksekusi (Numerik):



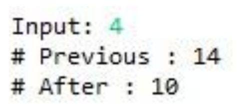
```
Input: 3
\-- List of Attributes --
@attribute outlook {sunny,overcast,rainy}
@attribute temperature numeric
@attribute humidity numeric
@attribute windy {FALSE,TRUE}
-- List of Attributes --
@attribute temperature numeric
@attribute humidity numeric
@attribute windy {FALSE,TRUE}
```

V. Filter pada Java

SupervisedFilter.java

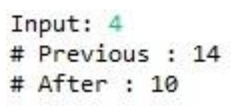
```java
public class SupervisedFilter {

    /**
     * Resample Instances (for test purposes, we take 75% size)
     * @param i
     * @return
     * @throws Exception
     */
    public static Instances resampleInstances(Instances i) throws
Exception {
        String Filteroptions="-B 1.0";
        Resample sampler = new Resample();
        /** Resample Options */
        sampler.setOptions(weka.core.Utils.splitOptions(Filteroptions));
        sampler.setRandomSeed((int)System.currentTimeMillis());
        sampler.setSampleSizePercent(75.0);
        sampler.setInputFormat(i);

        i = Resample.useFilter(i, sampler);

        return i;
    }

}
```

Hasil Eksekusi (Nominal):



```
Input: 4
# Previous : 14
# After : 10
```

Hasil Eksekusi (Numerik):



```
Input: 4
# Previous : 14
# After : 10
```

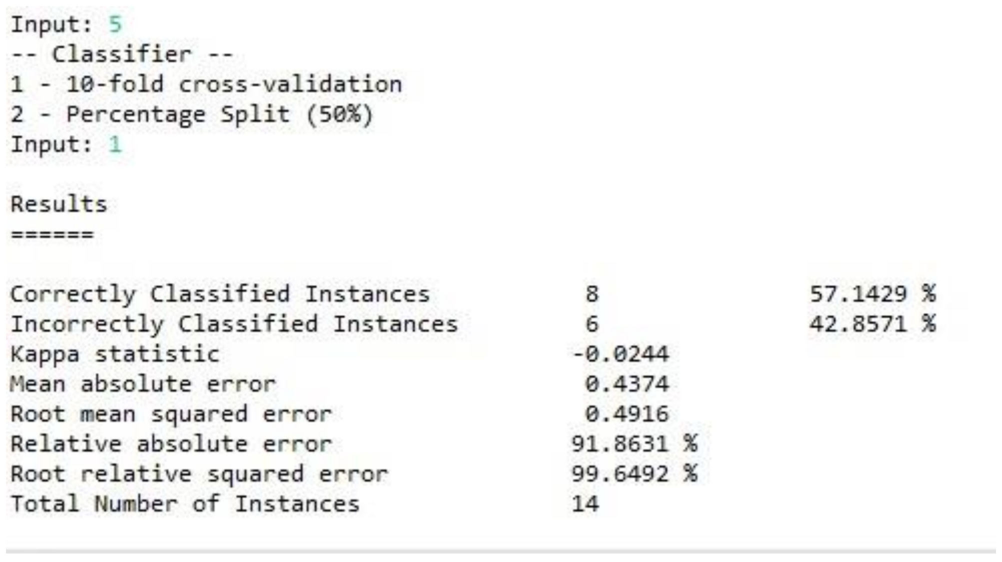
VI. Build Classifier (Naïve Bayes) pada Java

/** 10 fold cross validation & percentage split */

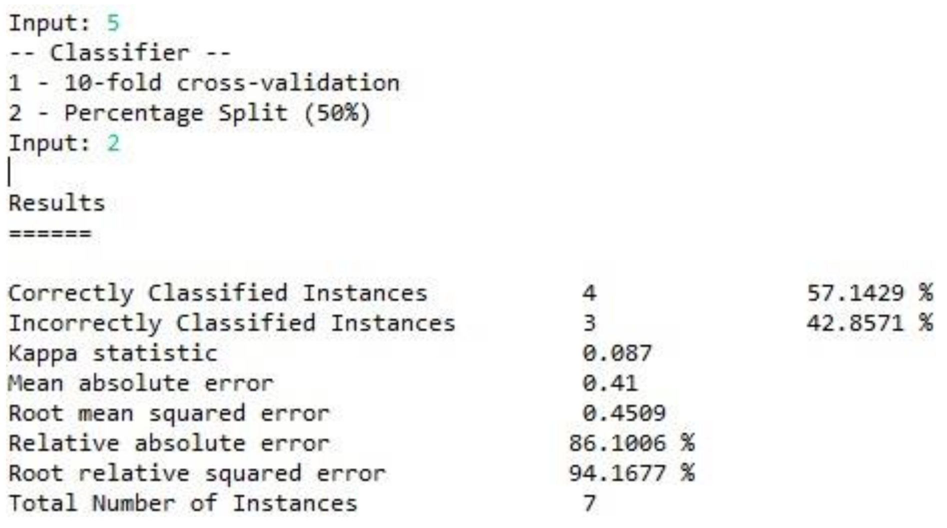
Bagian dari **ClassifyAlgorithm.java**

```
    /**
     * Classifier Model using Naive Bayes Algorithm
     *
     * @param trainingSet
     * @param id
     * @return
     * @throws Exception
     */
    public static Classifier naiveBayesAlgorithm(Instances trainingSet,
int id)
            throws Exception {
        // Create a naive bayes classifier
        Classifier cModel = (Classifier) new NaiveBayes();
        if (id == 1) {
            Evaluation eval = new Evaluation(trainingSet);
            eval.crossValidateModel(cModel, trainingSet, 10, new
Random(1));

System.out.println(eval.toSummaryString("\nResults\n======\n", false));
            cModel.buildClassifier(trainingSet);
        } else if (id == 2) {
            /** Split is not random (Preserve Order for debug) */
            int trainSize = (int) Math.round(trainingSet.numInstances() *
0.5);
            int testSize = trainingSet.numInstances() - trainSize;
            Instances train = new Instances(trainingSet, 0, trainSize);
            Instances test = new Instances(trainingSet, trainSize,
testSize);
            cModel.buildClassifier(train);

            /** Test section */
            Evaluation eval = new Evaluation(train);
            eval.evaluateModel(cModel, test);

System.out.println(eval.toSummaryString("\nResults\n======\n", false));
        }
        return cModel;
    }
```

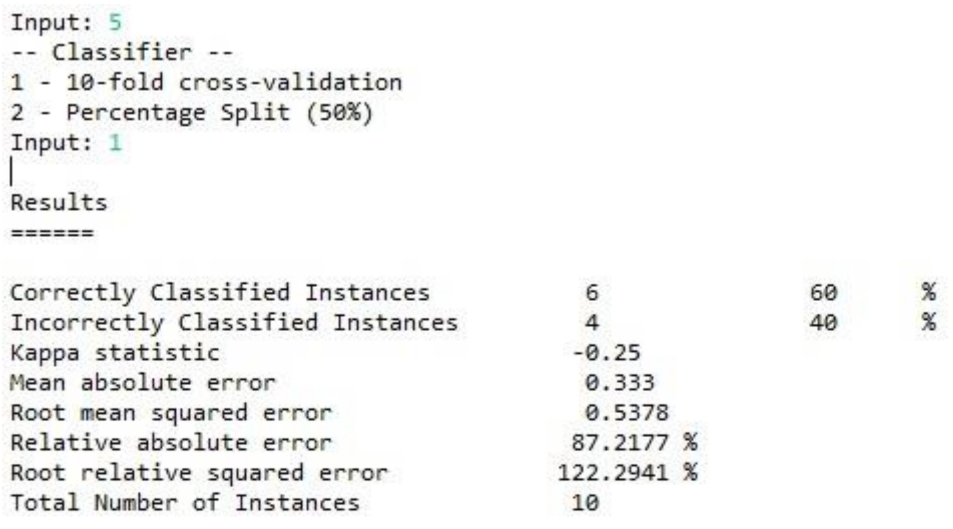
Hasil Eksekusi (10-fold cross-validation untuk Nominal):


```
Input: 5
-- Classifier --
1 - 10-fold cross-validation
2 - Percentage Split (50%)
Input: 1

Results
======

Correctly Classified Instances           8               57.1429 %
Incorrectly Classified Instances         6               42.8571 %
Kappa statistic                         -0.0244
Mean absolute error                      0.4374
Root mean squared error                  0.4916
Relative absolute error                 91.8631 %
Root relative squared error             99.6492 %
Total Number of Instances               14
```

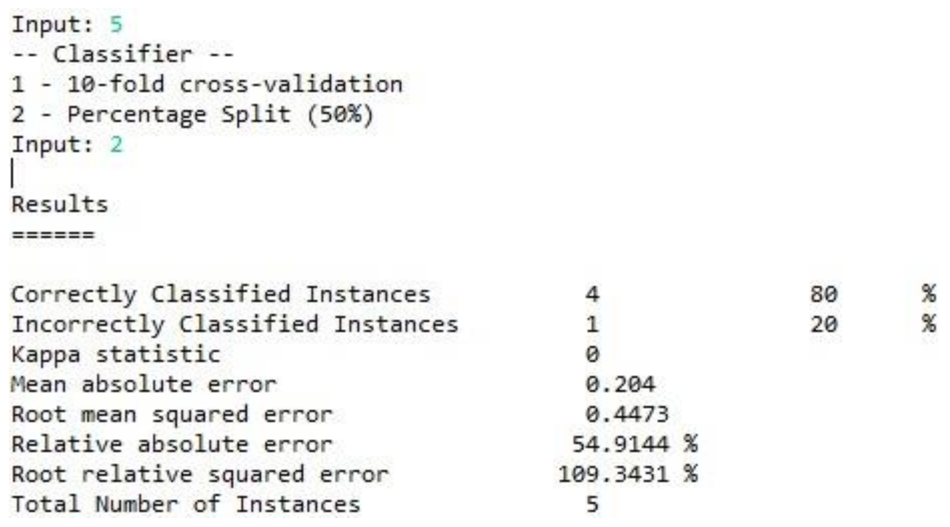
Hasil Eksekusi (Percentage split untuk Nominal):


```
Input: 5
-- Classifier --
1 - 10-fold cross-validation
2 - Percentage Split (50%)
Input: 2

Results
======

Correctly Classified Instances           4               57.1429 %
Incorrectly Classified Instances         3               42.8571 %
Kappa statistic                          0.087
Mean absolute error                      0.41
Root mean squared error                  0.4509
Relative absolute error                 86.1006 %
Root relative squared error             94.1677 %
Total Number of Instances                7
```

Hasil Eksekusi (10-fold cross-validation untuk Numerik):


```
Input: 5
-- Classifier --
1 - 10-fold cross-validation
2 - Percentage Split (50%)
Input: 1

Results
======

Correctly Classified Instances           6               60      %
Incorrectly Classified Instances         4               40      %
Kappa statistic                         -0.25
Mean absolute error                      0.333
Root mean squared error                  0.5378
Relative absolute error                 87.2177 %
Root relative squared error            122.2941 %
Total Number of Instances               10
```

Hasil Eksekusi (Percentage split untuk Numerik):



```
Input: 5
-- Classifier --
1 - 10-fold cross-validation
2 - Percentage Split (50%)
Input: 2
|
Results
======

Correctly Classified Instances           4               80      %
Incorrectly Classified Instances         1               20      %
Kappa statistic                          0
Mean absolute error                      0.204
Root mean squared error                  0.4473
Relative absolute error                 54.9144 %
Root relative squared error            109.3431 %
Total Number of Instances                5
```

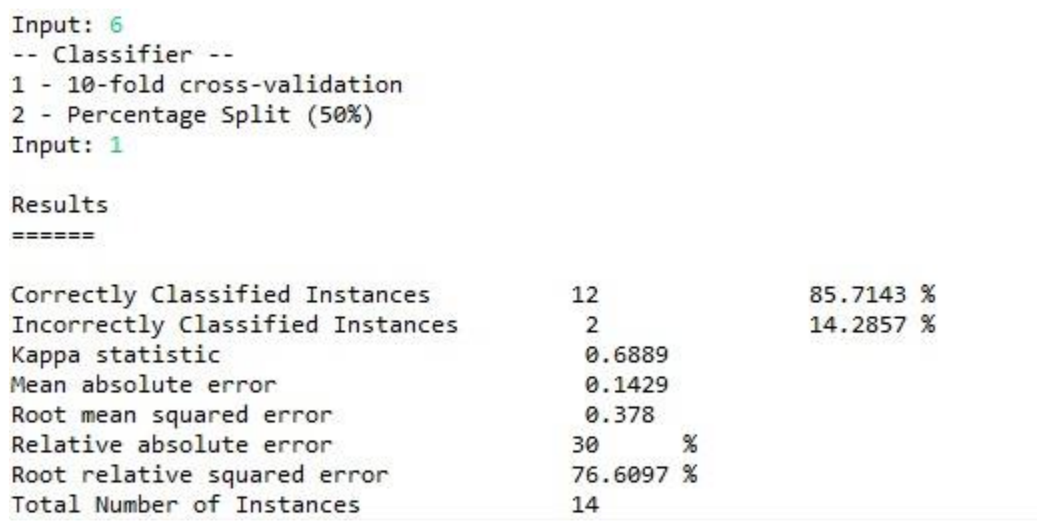
VII. Build Classifier (Decision Tree) pada Java

/** 10 fold cross validation & percentage split */

Bagian dari **ClassifyAlgorithm.java**

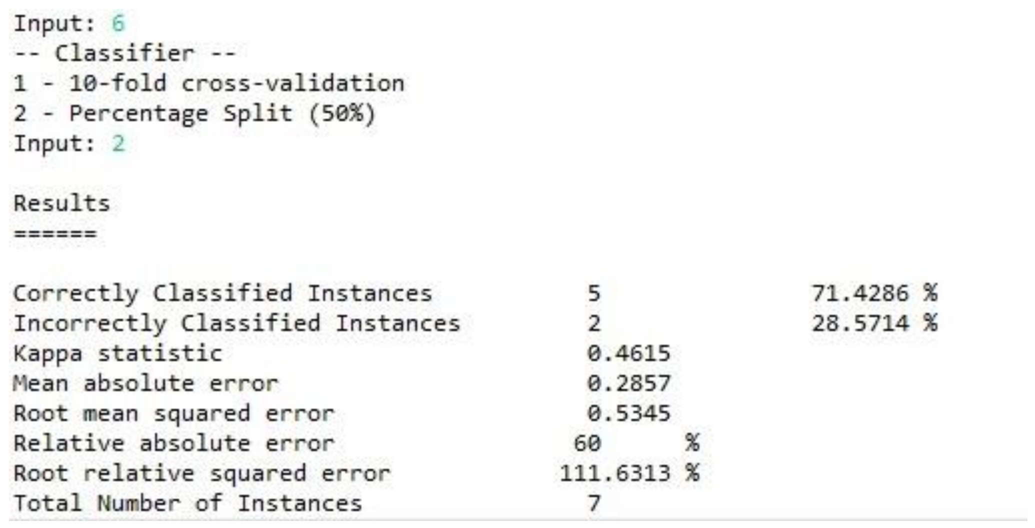
```
     /**
      * Classifier Model using ID Tree Algorithm
      * @param trainingSet
      * @param id
      * @return
      * @throws Exception
      */
     public static Classifier iD3Algorithm(Instances trainingSet, int id)
             throws Exception {
         // Create an ID3 classifier
         Classifier cModel = (Classifier) new Id3();
         if (id == 1) {
             Evaluation eval = new Evaluation(trainingSet);
             eval.crossValidateModel(cModel, trainingSet, 10, new
Random(1));

System.out.println(eval.toSummaryString("\nResults\n======\n", false));
             cModel.buildClassifier(trainingSet);
         } else if (id == 2) {
             /** Split is not random (Preserve Order for debug) */
             int trainSize = (int) Math.round(trainingSet.numInstances() *
0.5);
             int testSize = trainingSet.numInstances() - trainSize;
             Instances train = new Instances(trainingSet, 0, trainSize);
             Instances test = new Instances(trainingSet, trainSize,
testSize);
             cModel.buildClassifier(train);

             /** Test section */
             Evaluation eval = new Evaluation(train);
             eval.evaluateModel(cModel, test);
System.out.println(eval.toSummaryString("\nResults\n======\n", false));
         }
         return cModel;
     }
```

Hasil Eksekusi (10-fold cross-validation untuk Nominal):



```
Input: 6
-- Classifier --
1 - 10-fold cross-validation
2 - Percentage Split (50%)
Input: 1

Results
======

Correctly Classified Instances          12               85.7143 %
Incorrectly Classified Instances         2               14.2857 %
Kappa statistic                          0.6889
Mean absolute error                      0.1429
Root mean squared error                  0.378 
Relative absolute error                 30      %
Root relative squared error             76.6097 %
Total Number of Instances               14
```

Hasil Eksekusi (Percentage split untuk Nominal):



```
Input: 6
-- Classifier --
1 - 10-fold cross-validation
2 - Percentage Split (50%)
Input: 2

Results
======

Correctly Classified Instances           5               71.4286 %
Incorrectly Classified Instances         2               28.5714 %
Kappa statistic                          0.4615
Mean absolute error                      0.2857
Root mean squared error                  0.5345
Relative absolute error                 60      %
Root relative squared error            111.6313 %
Total Number of Instances                7
```

VIII. Save & Load Model pada Java

FileHelper.java

```
public class FileHelper {

    /**
     * Save cModel to path
     * @param cModel
     * @param path
     * @throws Exception
     */
    public static void saveModel(Classifier cModel, String path)
            throws Exception {
        ObjectOutputStream oos = new ObjectOutputStream(new
FileOutputStream(
                path));
        oos.writeObject(cModel);
        oos.flush();
        oos.close();
    }

    /**
     * Load cModel from path
     * @param path
     * @return
     * @throws Exception
     */
    public static Classifier loadModel(String path) throws Exception {
        ObjectInputStream ois = new ObjectInputStream(new
FileInputStream(path));
        Classifier cModel = (Classifier) ois.readObject();
        ois.close();
        return cModel;
    }

}
```

Hasil Eksekusi:

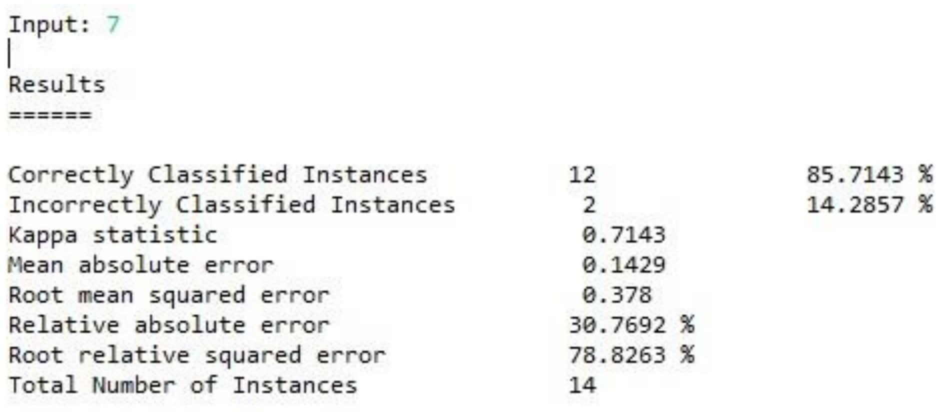
Sebuah file .model yang dapat digunakan untuk proses *load model*.

IX. Testing Model dengan Given Test Set pada Java

Bagian dari **Main.java**

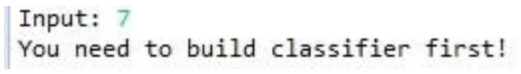
```
  Evaluation eval = new Evaluation(data);
  eval.evaluateModel(cModel, data);
  System.out.println(eval.toSummaryString(
            "\nResults\n======\n", false));
```

Hasil Eksekusi (Nominal):



```
Input: 7
|
Results
======

Correctly Classified Instances          12               85.7143 %
Incorrectly Classified Instances         2               14.2857 %
Kappa statistic                          0.7143
Mean absolute error                      0.1429
Root mean squared error                  0.378
Relative absolute error                 30.7692 %
Root relative squared error             78.8263 %
Total Number of Instances               14
```

Hasil Eksekusi (Numerik):



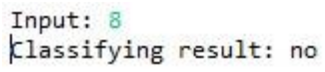
```
Input: 7
You need to build classifier first!
```

X. Classify Unseen Data dengan Existing Model pada Java

Bagian dari **Main.java**

```
    Instance test = new Instance(5);
    if (isNominal) {
        test.setValue(data.attribute(0), "sunny");
        test.setValue(data.attribute(1), "mild");
        test.setValue(data.attribute(2), "high");
        test.setValue(data.attribute(3), "FALSE");
    } else {
        test.setValue(data.attribute(0), "rainy");
        test.setValue(data.attribute(1), 65);
        test.setValue(data.attribute(2), 70);
        test.setValue(data.attribute(3), "TRUE");
    }
    // Give access to dataset
    test.setDataset(data);

    System.out.print("Classifying result: ");
    System.out.println(data.attribute(data.numAttributes() - 1).
            value((int) cModel.classifyInstance(test)));
```

Hasil Eksekusi (Nominal):



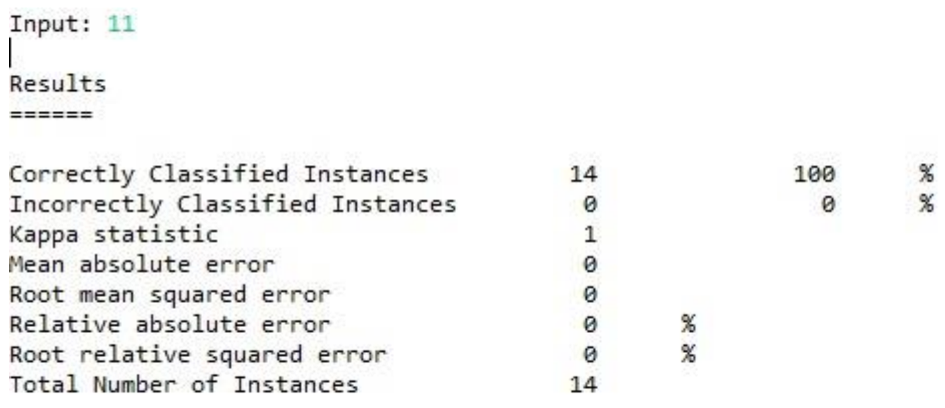
```
Input: 8
Classifying result: no
```

XI. Classifier Turunan pada Java

CustomAlgorithm.java

```
@SuppressWarnings("serial")
public class CustomAlgorithm extends Classifier {

    Instances m_Instances;
    int m_nAttributes;

    @Override
    public void buildClassifier(Instances data) throws Exception {
        // TODO Auto-generated method stub
        m_Instances = new Instances(data);
        m_nAttributes = data.numAttributes();
    }

    @Override
    public double classifyInstance(Instance instance) {
        @SuppressWarnings("rawtypes")
        Enumeration enu = m_Instances.enumerateInstances();
        double distance = 9999999;
        double classValue = -1;
        while (enu.hasMoreElements()) {
            Instance _instance = (Instance) enu.nextElement();
            double _distance = CalculateDistance(instance, _instance);
            if (_distance < distance) {
                distance = _distance;
                classValue = _instance.classValue();
            }
        }
        return classValue;
    }

    public double CalculateDistance(Instance i1, Instance i2) {
        double s = 0;
        for (int i = 0; i < m_nAttributes - 1; i++) {
            double p = (i1.value(i) - i2.value(i));
            s += p * p;
        }
        return s;
    }

    @Override
    public String[] getOptions() {
        String[] options = new String[2];
        int current = 0;
        while (current < options.length)
            options[current++] = "";
        return options;
    }

}
```

Hasil Eksekusi:


```
Input: 11
|
Results
======

Correctly Classified Instances          14              100      %
Incorrectly Classified Instances         0                0      %
Kappa statistic                          1
Mean absolute error                      0
Root mean squared error                  0
Relative absolute error                  0      %
Root relative squared error              0      %
Total Number of Instances               14
```

XII. Referensi

[1] *Weka 3: Data Mining Software in Java*. Diakses 16 September 2014 pukul 18.00.

< http://www.cs.waikato.ac.nz/ml/weka/ >