Tugas Pra-Praktikum

Implementasi Classifier WEKA *myID3*

Iskandar Setiadi 13511073 | Raden Fajar Hadria Putra 13511076 | M Andri Eka Fauzy 13511088

IF4071 Pembelajaran Mesin - Kelompok WbTeladan

Program Studi Teknik Informatika - STEI ITB

Tahun 2014

A. Struktur Data

Nama Kelas : myID3
Parent Class : Classifier
Atribut :

Nama Atribut	Tipe Atribut	Fungsi Atribut
successors	array of myID3	Suksesor dari node
attribute	Attribute	atribut yang digunakan untuk *splitting*
classValue	double	nilai dari node jika node merupakan daun
distribution	array of double	distribusi dari node jika node merupakan daun
classAttribute	Attribute	atribut node dari dataset
infoGain	array of double	nilai *information gain* dari semua attribute di node suatu tree.
anotherCA	array of String	Nama-nama attribute dari data yang dibuat ID3

Metode :

Nama Metode	Output Metode	Parameter	Fungsi Metode
getCapabilities	Capabilities	-	mengembalikan Capabilities default dari *classifier*
buildClassifier	-	data (Instances)	Membangun *classifier* myID3
makeTree	-	data (Instances)	Membangun pohon (*tree*)

			myID3
classifyInstance	double	instance (Instance)	Mengembalikan nilai *classValue* hasil klasifikasi menggunakan myID3. Selain itu, menampilkan *path* dari proses klasifikasi
distributionForInstance	array of double	instance (Instance)	Menghitung distribusi node untuk instance yang menggunakan *decision tree*
toString	String	-	Menampilkan model *decision tree*
computeInfoGain	double	- data (Instances) - att (Attribute)	Menghitung *information gain* dari atribut pada parameter
computeEntropy	double	data (Instances)	Menghitung entropi dari dataset
splitData	array of Instances	- data (Instances) - att (Attribute)	Memisahkan dan mengelompokkan data berdasarkan atribut pada parameter
toString	String	level (Integer)	Menampilkan pohon pada level tertentu

B. Implementasi

Implementasi Pembuatan Pohon :

```
private void makeTree(Instances data) throws Exception {

        // Check if no instances have reached this node.
        if (data.numInstances() == 0) {
                attribute = null;
                classValue = Instance.missingValue();
                distribution = new double[data.numClasses()];
                return;
        }

        // Compute attribute with maximum information gain.
        // double[] infoGains = new double[data.numAttributes()];

        infoGains = new double[data.numAttributes()];
        anotherCA = new String[data.numAttributes()];
        int count = 0;

        Enumeration attEnum = data.enumerateAttributes();
        while (attEnum.hasMoreElements()) {
                Attribute att = (Attribute) attEnum.nextElement();
```

```
                anotherCA[count] = att.name();
                infoGains[att.index()] = computeInfoGain(data, att);
                count++;
        }
        attribute = data.attribute(Utils.maxIndex(infoGains));// attribute with Max IG
value

        // Make leaf if information gain is zero.
        // Otherwise create successors.
        if (Utils.eq(infoGains[attribute.index()], 0)) {
                attribute = null;
                distribution = new double[data.numClasses()];
                Enumeration instEnum = data.enumerateInstances();
                while (instEnum.hasMoreElements()) {
                        Instance inst = (Instance) instEnum.nextElement();
                        distribution[(int) inst.classValue()]++;
                }
                Utils.normalize(distribution); // convert it into 0.0 to 1.0 ratio
                classValue = Utils.maxIndex(distribution);
                classAttribute = data.classAttribute();
        } else {
                Instances[] splitData = splitData(data, attribute);
                successors = new myID3[attribute.numValues()];
                for (int j = 0; j < attribute.numValues(); j++) {
                        successors[j] = new myID3();
                        successors[j].makeTree(splitData[j]);
                }
        }
}
```

Implementasi Perhitungan Entropi :

```
private double computeEntropy(Instances data) throws Exception {

         double[] classCounts = new double[data.numClasses()];
         Enumeration instEnum = data.enumerateInstances();
         while (instEnum.hasMoreElements()) {
                 Instance inst = (Instance) instEnum.nextElement();
                 classCounts[(int) inst.classValue()]++;
         }
         double entropy = 0;
         for (int j = 0; j < data.numClasses(); j++) {
                 if (classCounts[j] > 0) {
                         entropy -= classCounts[j] * Utils.log2(classCounts[j]);
                 }
         }
         entropy /= (double) data.numInstances();
         return entropy + Utils.log2(data.numInstances());
    }
```

Implementasi Perhitungan Information Gain :

```
private double computeInfoGain(Instances data, Attribute att)
                 throws Exception {

         double infoGain = computeEntropy(data);
         Instances[] splitData = splitData(data, att);
         for (int j = 0; j < att.numValues(); j++) {
                 if (splitData[j].numInstances() > 0) {
                         infoGain -= ((double) splitData[j].numInstances() / (double)
data.numInstances()) * computeEntropy(splitData[j]);
```

```
                }
        }
        return infoGain;
    }
```

C. Pengujian

Data Set yang Digunakan :

```
@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
sunny,hot,high,FALSE,no
sunny,hot,high,TRUE,no
overcast,hot,high,FALSE,yes
rainy,mild,high,FALSE,yes
rainy,cool,normal,FALSE,yes
rainy,cool,normal,TRUE,no
overcast,cool,normal,TRUE,yes
sunny,mild,high,FALSE,no
sunny,cool,normal,FALSE,yes
rainy,mild,normal,FALSE,yes
sunny,mild,normal,TRUE,yes
overcast,mild,high,TRUE,yes
overcast,hot,normal,FALSE,yes
rainy,mild,high,TRUE,no
```

weather.nominal.arff

ID3 yang dihasilkan :

```
Id3
|
 outlook (IG = 0.24674981977443894)
 temperature (IG = 0.029222565658954536)
```

```
 humidity (IG = 0.15183550136234125)
 windy (IG = 0.04812703040826921)
 outlook = sunny

|
|
 outlook (IG = 0.0)
 temperature (IG = 0.5709505944546683)
 humidity (IG = 0.9709505944546684)
 windy (IG = 0.019973094021974558)
 humidity = high
 Kelas : no [LEAF]

|
|
 outlook (IG = 0.0)
 temperature (IG = 0.5709505944546683)
 humidity (IG = 0.9709505944546684)
 windy (IG = 0.019973094021974558)
 humidity = normal
 Kelas : yes [LEAF]


|
 outlook (IG = 0.24674981977443894)
 temperature (IG = 0.029222565658954536)
 humidity (IG = 0.15183550136234125)
 windy (IG = 0.04812703040826921)
 outlook = overcast
 Kelas : yes [LEAF]

|
 outlook (IG = 0.24674981977443894)
 temperature (IG = 0.029222565658954536)
 humidity (IG = 0.15183550136234125)
 windy (IG = 0.04812703040826921)
 outlook = rainy

|
|
 outlook (IG = 0.0)
 temperature (IG = 0.019973094021974558)
 humidity (IG = 0.019973094021974558)
 windy (IG = 0.9709505944546684)
 windy = TRUE
 Kelas : no [LEAF]

|
|
 outlook (IG = 0.0)
```

```
temperature (IG = 0.019973094021974558)
humidity (IG = 0.019973094021974558)
windy (IG = 0.9709505944546684)
windy = FALSE
Kelas : yes [LEAF]
```

Ilustrasi Pohon :