A Program demonstrating Gini Index Classification

Abstract

In this document, a small program demonstrating Gini Index Classification is introduced. Users can select specified training data set, build the decision tree, and then select independent testing data set to verify the accuracy of model. Furthermore, by providing the decision tree visualization (Using JTree of Java), the pattern recognition capacities of users can be greatly improved.When estimating classifier accuracy, the known class label is compared with the learned model's class prediction for that sample, the conflict records will be filtered to show user what the record's class label is and what the mined model tells you the result is supposed should be. The program is realized by Java under Windows XP.

1. Introduction

In classification, we are given a set of example records, called a training set, where each record consists of several fields or attributes. Attributes are continuous, coming from an ordered domain, or categorical, coming from an unordered domain. One of the attributes, called the classifying attribute, indicates the class to which each example belongs. The objective of classification is to build a model of the classifying attribute based upon the other attributes. Once a model is built, it can be used to determine the class of future unclassified records. Applications of classification arise in diverse fields, such as retail target marketing, customer retention, fraud detection and medical diagnosis

Classification is often seen as the most useful form of data mining. Although every pattern recognition technique has been used to do classification, decision trees are the most popular. This may be because they form rules which are easy to understand, or perhaps because they can be converted easily into SQL. While not as "robust" as neural nets and not as statistically "tidy" as discriminate analysis, decision trees often show very good generalization capability.

Two independent data set:

- *Training set*: a set of examples, where each example is a feature vector (i.e., a set of <attribute,value> pairs) with its associated class. The model is built on this set.
- *Test set*: a set of examples disjoint from the training set, used for testing the accuracy of a model.

The classification models range from easy to understand to incomprehensible are: Decision trees, Rule induction, Regression models, Genetic Algorithms, Bayesian Networks, Neural networks.

Gini index algorithm use Gini index to evaluate the goodness of the split. If a data set *T* contains examples from *n* classes, gini index, *gini*(*T*) is defined as

$$gini(T) = 1 - \sum_{j=1}^{n} p_j^2$$

where *pj* is the relative frequency of class *j* in *T.*

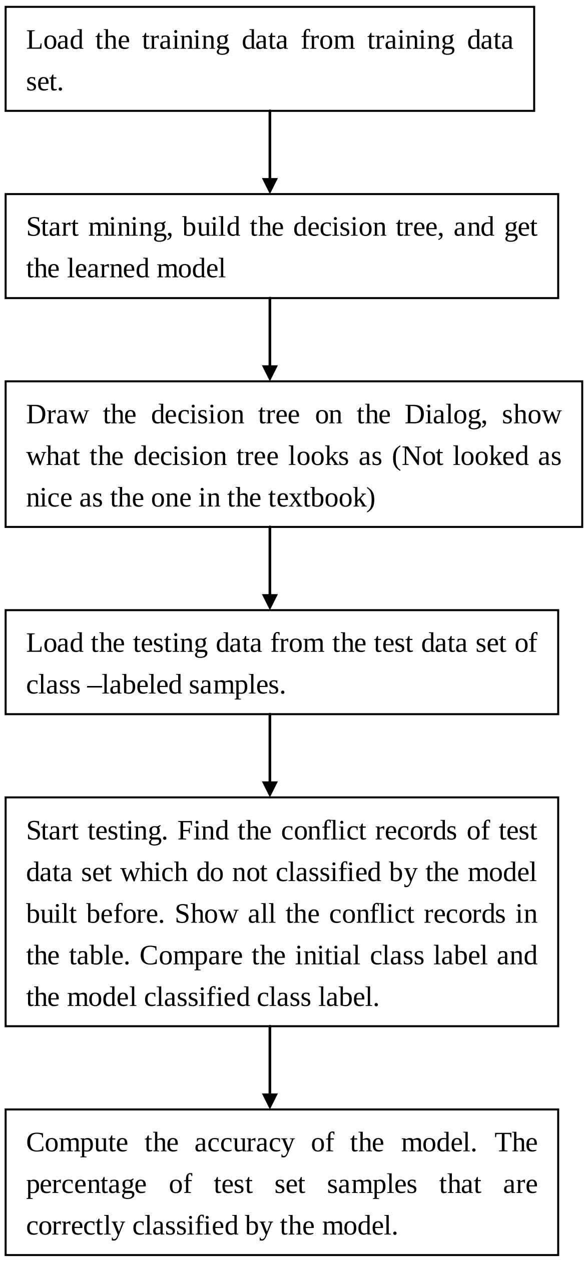
If a data set *T* is split into two subsets *T1* and *T2* with sizes *N1* and *N2* respectively, the *gini* index of the split data contains examples from *n* classes, the *gini* index *gini*(*T*) is defined as

$$gini_{split}(T) = \frac{N_1}{N} gini(T_1) + \frac{N_2}{N} gini(T_2)$$

The attribute provides the smallest *ginisplit*(*T*) is chosen to split the node (*need to enumerate all possible splitting points for each attribute*).

2. Realization

1) The main procedure



Load the training data from training data set.

Start mining, build the decision tree, and get the learned model

Draw the decision tree on the Dialog, show what the decision tree looks as (Not looked as nice as the one in the textbook)

Load the testing data from the test data set of class –labeled samples.

Start testing. Find the conflict records of test data set which do not classified by the model built before. Show all the conflict records in the table. Compare the initial class label and the model classified class label.

Compute the accuracy of the model. The percentage of test set samples that are correctly classified by the model.

2) The data class

- *Record*: The data to be mined, each sample of training data and testing data;
- *AttributeImp*: the attribute of records, encapsulate the categorical attribute and numerical attribute;
- *DesionTreeNode*: the node of decision tree. Include the decition attribute,the split point and the class value if it is a leaf, as well as the other attribute a tree node required such as parent node, children node.

3) Data format requirement

The training data set and the testing data set should share the same data format. We load the data from a text file. Require the first line should be the attribute name of records to be mined, and the class label should be the last attribute, the word or phrase representing attribute name split with space. The attribute numbers are not limited, in theory, you can mine the data their attribute can be as large as you will, usually the record with too much attributes are not recommended. And the types of attributes are not limited. it can be numeric attributes or categorical attributes. In the program, they will be encapsulated as ***AttributeImp*** and all the operation upon them are identical.

4) Build the decision Tree using training data, the key part of the work!

a) **Main idea**: For each attribute, try to use each value of this attribute as split point split the record into two partition, compute the Gini, the one with the lowest Gini will be choose as the split point of this attribute.

Do this work for each attribute, compute the Gini, the one with the lowest Gini will be choose as the test attribute. Create a node and labeled as this attribute, the records then partitioned into left records and right records accordingly, for the left records do the same work as before using the rest attributes, and then the right records.

The recursion will be end till the records are empty or the attributes all being used, and another case, all the samples are belong to the same class.

In the end of the branch, when there are no remaining attributes on which the samples may be further partitioned, convert the given node into a leaf and label it with the class in majority among the samples

b) Argorithm:BuildTreeNode

Input: ArrayList records //The collection of records load from the training data set)
ArrayList attributeNames // The collection attribute also load from the

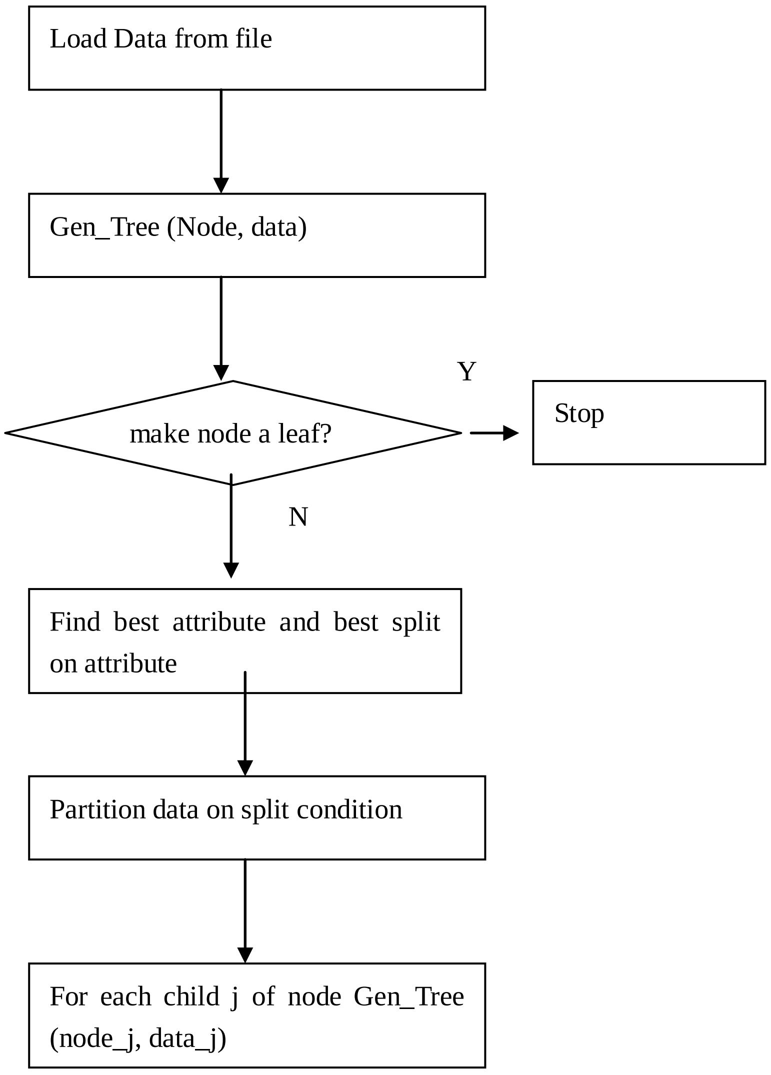
training data set

Output: DesionTreeNode root // The root of decision tree.

Method:

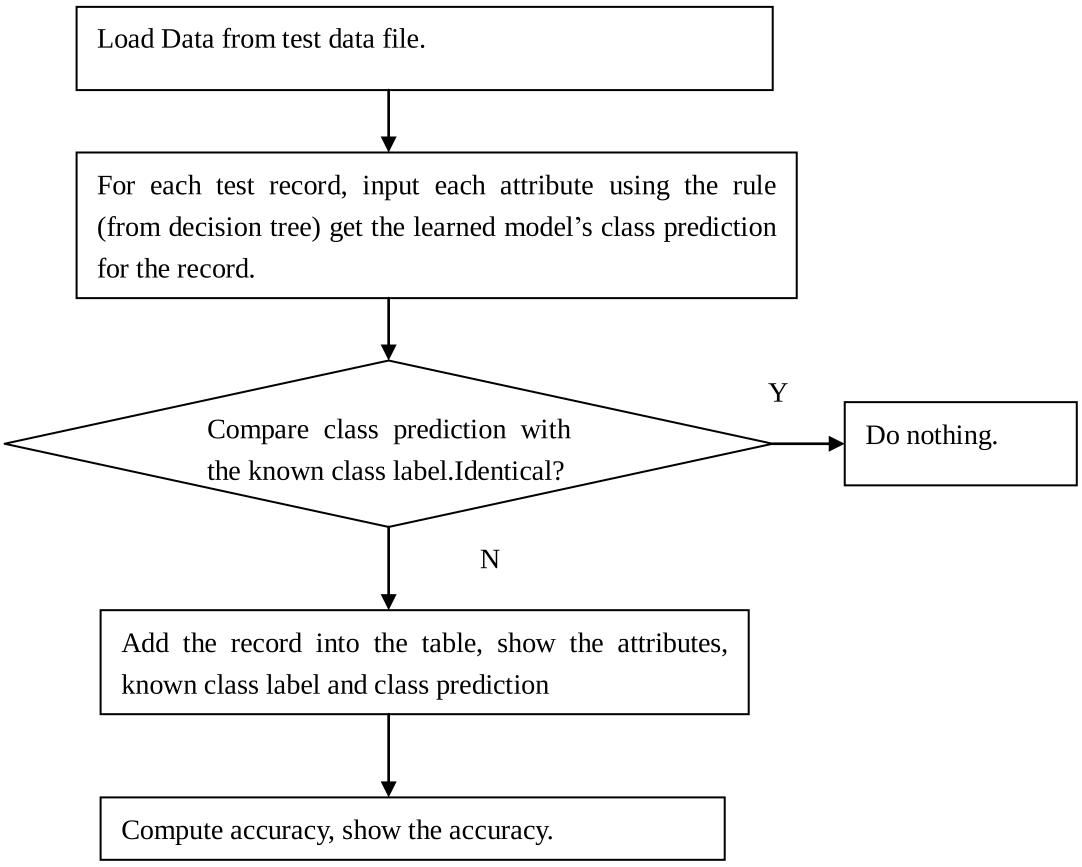
```
BuildTreeNode (ArrayList records,ArrayList attributeNames)
 splitRecords (records)
 if (all points in records belong to the same class) then
return;
 for each attribute A do
evaluate splits on attribute A using attribute list;
use best split found to partition records into leftRecord and rightRecord (Keep split
with lowest GINI index);
splitRecords (leftRecord);
splitRecords (rightRecord);
```

c) **Procedure:**


Load Data from file
Gen_Tree (Node, data)
make node a leaf?
Y
Stop
N
Find best attribute and best split on attribute
Partition data on split condition
For each child j of node Gen_Tree (node_j, data_j)

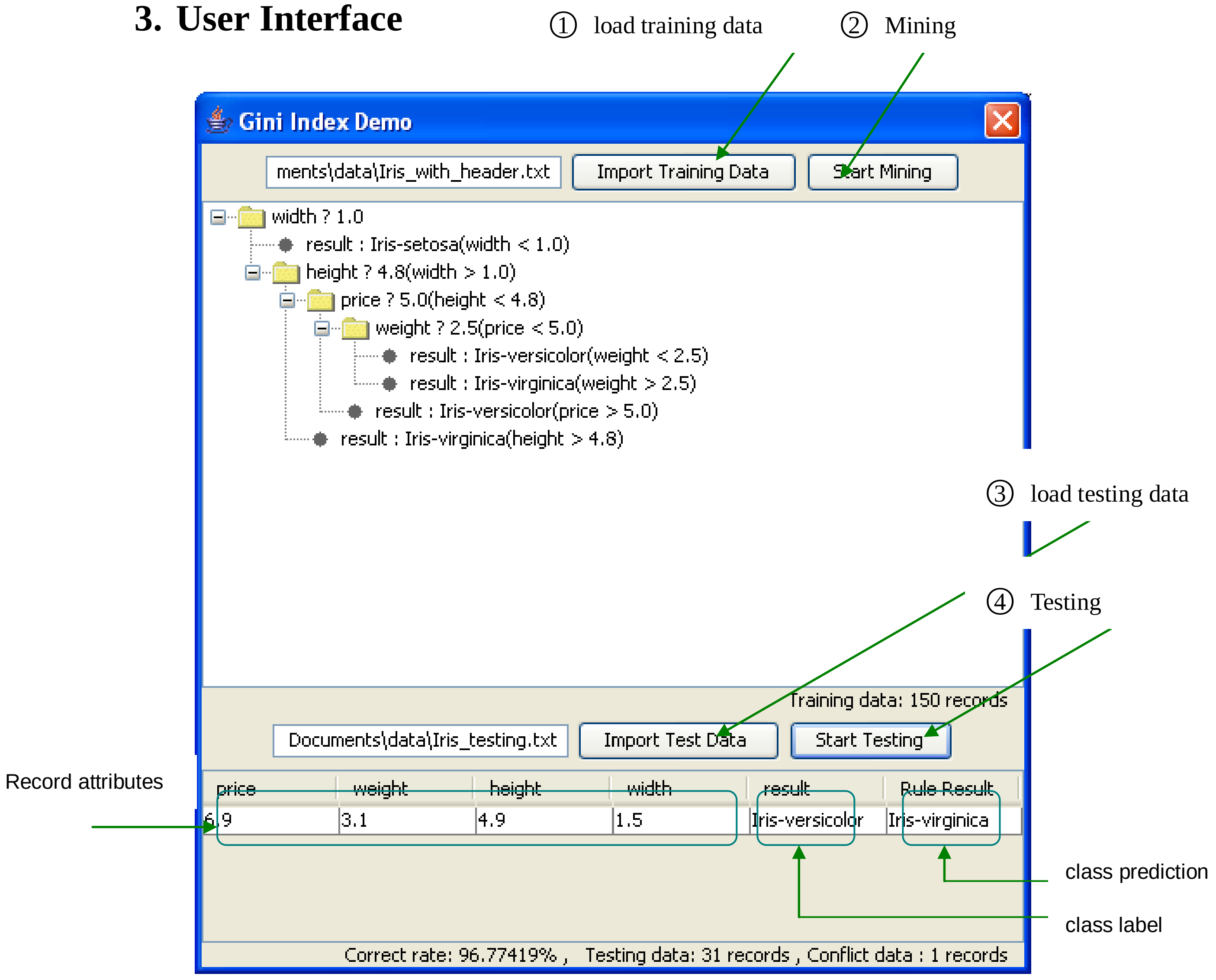
5) Test the model, compute the accuracy

Using the test data set which are randomly selected and are independent of the training data, Compute the percentage of test set samples that are correctly classified by the mode.



Load Data from test data file.
For each test record, input each attribute using the rule (from decision tree) get the learned model's class prediction for the record.
Compare class prediction with the known class label.Identical?
Y
Do nothing.
N
Add the record into the table, show the attributes, known class label and class prediction
Compute accuracy, show the accuracy.

3. User Interface



```
ments\data\Iris_with_header.txt
width ? 1.0
    result : Iris-setosa(width < 1.0)
    height ? 4.8(width > 1.0)
        price ? 5.0(height < 4.8)
            weight ? 2.5(price < 5.0)
                result : Iris-versicolor(weight < 2.5)
                result : Iris-virginica(weight > 2.5)
            result : Iris-versicolor(price > 5.0)
        result : Iris-virginica(height > 4.8)
Training data: 150 records
Documents\data\Iris_testing.txt
```

price	weight	height	width	result	Rule Result
6.9	3.1	4.9	1.5	Iris-versicolor	Iris-virginica

```
Correct rate: 96.77419% ,   Testing data: 31 records , Conflict data : 1 records
```

There is no need to say too much of how to use this small grogram besides pointing out the sequence of operation and the purpose of each part on the panel because it is really too simple.

4. Testing results

Training Data Set Name	Training Attribute Count	Training Records Count	Testing Data Set Name	Training Attribute Count	Training Records Count	Accuracy
Iris_training.txt	5	119	Iris_testing.txt	5	31	93.55%
heart train3-200.txt	14	200	heart test3-70.txt	14	70	81.43%
credit train2.txt	14	300	credit test2.txt	14	107	89.72%

We use three different data sets with separate training data set and test data set. The test results show:

- It will become harder for the records with more attribute to achieve high accuracy compare with the one with less attributes;
- The more sufficient records in the training data set can help construct a more accurate model.

5. Something can be improved

- The built decision tree is a binary branch tree, split values of attribute into two partitions, using Gini index, in some cases it may not work well, need consider more split point be extended to handle multi-way splits or using the cluster to split the values of each attribute.
- No preprocess.
- Construct the decision tree, draw the tree, save the model in the memory, then test the model, not save the rule on persitent media such as disk file for future use.

6. Environment requirement: Java Runtime environment 1.5

7. Conclusion

The motivation for this program is an exercise of Data Ming class 2006.With the processing of develop work, deepen the understanding of the algorithm, and found some inspiration in the work.

The program can interpret the dataset in a text file with unlimited attributes because of using extensible data class design.

The framework can easily be used to other decision tree mining. The only thing need to do is to modify the built tree algorithm using the specified mining algorithm such as ID3, C4.5. The data class (*Record*,*AttributeImp*,*DesionTreeNode*) can be used without change. And all the procedure such as loading the training data, drawing the decision tree, loading the testing data, testing model, computing the accuracy of the model can be used without any modification.

In the future work, I will try to add other decision tree classification algorithm into the frame work, add the function of store the learned model(decisiont tree here)into persitent file.

REFERENCES

1. Jiawei Han , Micheline Kamber, Data mining: concepts and techniques, Morgan Kaufmann Publishers Inc., San Francisco, CA, 2000

Author: zhangchaomeng
ID:066100593
Partner:Louyong