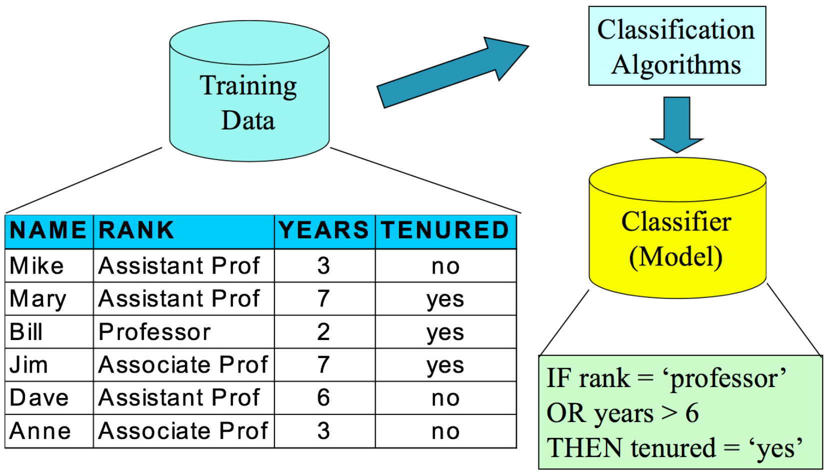
Data Classification

The Decision Tree Approach

Introduction

Decision Tree is an approach to achieve data classification. Data classification is the process of organizing data into categories for its most effective and efficient use. It predicts categorical class labels and classifies data base on the training set and the values which are class labels in a classifying attribute and uses it in classifying new data. It typically applies for the fields of credit/ loan approval, medical diagnosis, fraud detection, web page categorization and so on. Classification works during two steps of a learning step(FIG 1) and a classification step(FIG 2).

FIG 1: Step One- Learning Step

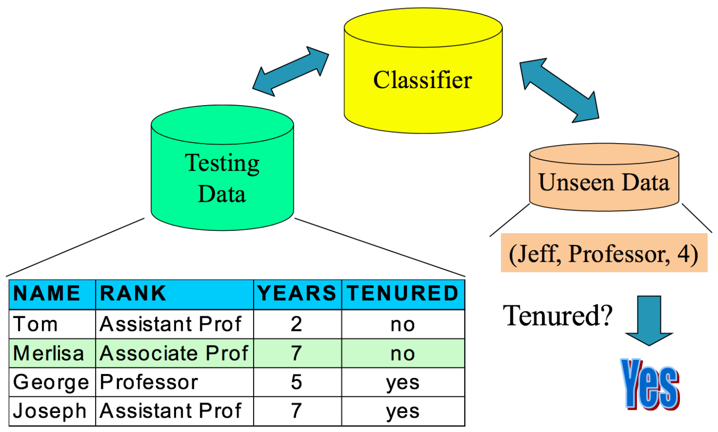


NAME	RANK	YEARS	TENURED
Mike	Assistant Prof	3	no
Mary	Assistant Prof	7	yes
Bill	Professor	2	yes
Jim	Associate Prof	7	yes
Dave	Assistant Prof	6	no
Anne	Associate Prof	3	no

IF rank = ‘professor’
OR years > 6
THEN tenured = ‘yes’

In the first step, it constructs model that describes a set of predetermined classes. This is the learning step where a classification algorithm builds the classifier by analyzing or learning from a training set made up of database tuples and their associated class labels. Each tuple is assumed to belong to a predefined class, and the training set is the set of tuples used for model construction. The model can be represented as a decision tree.

FIG 2: Step Two- Classification Step

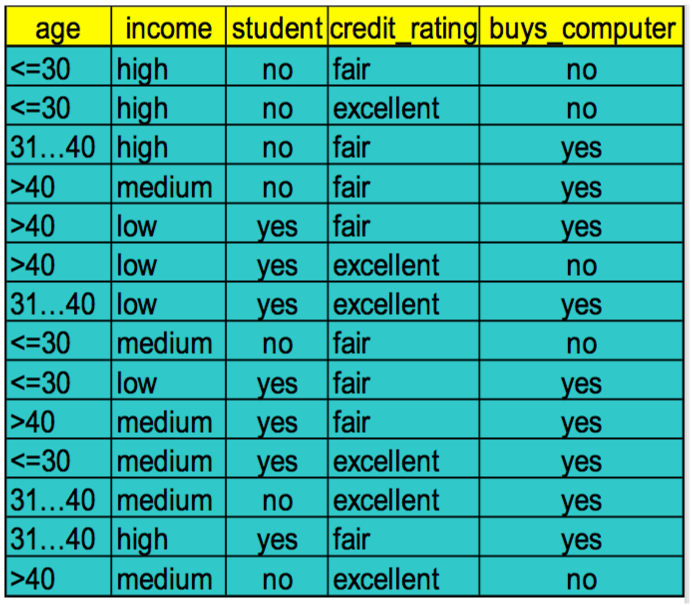


NAME	RANK	YEARS	TENURED
Tom	Assistant Prof	2	no
Merlisa	Associate Prof	7	no
George	Professor	5	yes
Joseph	Assistant Prof	7	yes

In the second step, the model which generated by the first step is used for classification. The ac classification accuracy on a given test set is the percentage of test set tuples that are correctly classified by the classifier. If the accuracy is acceptable, the classifier is used for classification of the future data tuples for which the class label is not know so that to achieve prediction by using the model.

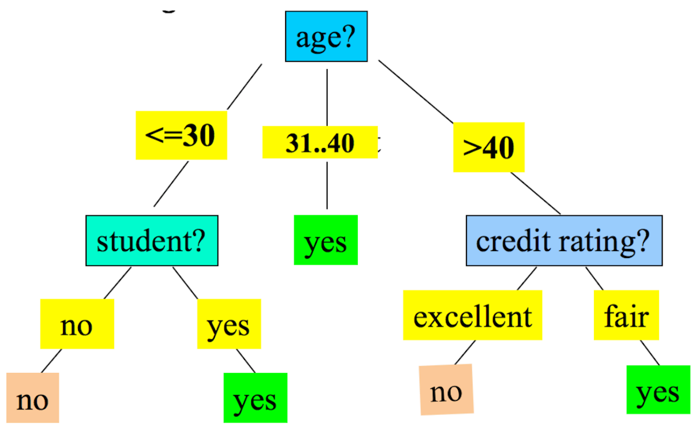
This report mentioned it before the model can be presented as an decision tree. A decision tree is a flowchat-like tree structure where each internal node represents a test on an attribute and each branch denotes an outcome of the test, and each leaf node holds a class label. Below is a typical decision tree(FIG 3) generated from the training data set (TABLE 1) that comes from the concept buys_computer, indicating whether an ALLElectronics customer is likely to purchase a computer. Each internal node denotes a test on an attribute. Each leaf node represents a class that either “buys_computer = yes” or “buys_computer = No”.

TABLE 1 : Training Data Set: buys_computer



age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

FIG 3: Decision Tree



To classify data set by using the decision tree is that given a tuple for an unknown class label and then test the attribute values of the tuple against the decision tree. Tracing a path from the root to a leaf node so that find the leaf node which holds the class prediction for the tuple.

Experimental Design

The construction of the decision tree is independent of domain knowledge. It uses attribute selection measures to select the attribute that best partition the tuples into distinct classes, thus, therefore, the construction of the decision tree is to determine the topological structure of each attribute by making the attribute selection measures.

The key step in construction a decision tree is to split the attributes. Splitting attribute is to construct a different branch according to the different attribute of a certain attribute at a certain node. The goal of this way is to make each split subset as pure as possible. As far as possible pure, is to try to make a spilt subset of the items to be classified belong to the same category. During the late 1970s and early 1980s, J.Ross Quinlan developed a decision tree algorithm called ID3 which is stand for Iterative Dichotomiser). Later, Quinlan declared C4.5 that is a successor of ID3, which became a benchmark to which newer supervised learning algorithms are often compared. Then, in 1984s, L. Breiman, J. Frideman, R. Olshen, and C. Stone presented CART which is stand for Classification and Regression Trees to described the generation of binary decision trees. The ID3, C4.5 and CART adopt a greedy approach in which decision trees are constructed in a top-down recursive divide- and conquer manner. The basic algorithm can be described as below.

Basic Algorithm (A Greedy Algorithm)

- Tree is constructed in a top-down recursive divide-and-conquer manner
- At start, all the training examples are at the root
- Attributes are categorical
- Examples are partitioned recursively based on selected attributes

- Test attributes are selected on the basis of a heuristic or statistical measure

Conditions for stopping partitioning

- All samples for a given node belong to the same class
- There are no remaining attributes for further partitioning- majority voting is employed for classifying the leaf
- There are no samples left

This project is implemented by ID3 which mentioned before. It called with three parameters that are D, attribute_list and Attribute_selection-method. The D is a data partition. The attribute_list is a list of attributes describing the tuples, and the attribute_selection_method specifies a heuristic procedure for selecting the attribute that "best" discriminates the given tuples according to class.

From the knowledge of information theory, we know that the smaller the expected information, the greater the information gain, the higher the purity. Therefore, the core idea of the ID3 algorithm is to choose the information gain measure attribute and then select the splitting information gain maximum split.

D is a data partition, the entropy of D is :

$$Info(D) = -\sum_{i=1}^{m} p_i \log_2(p_i)$$

Where pi is the probability of the i-th class occurring in the entire training tuple, and the number of elements belonging to this class can be divided by the total number of training tuple elements as an estimate. The actual meaning of the entropy is the average amount of information

Now suppose dividing the training tuple D according to attribute A, then the expected information of A to D is :

$$Info_A(D) = \sum_{j=1}^{v} \frac{|D_j|}{|D|} \times Info(D_j)$$

And the information gained by branching on attribute A is the difference between Info(D) and InfoA (D):

$$Gain(A) = Info(D) - Info_A(D)$$

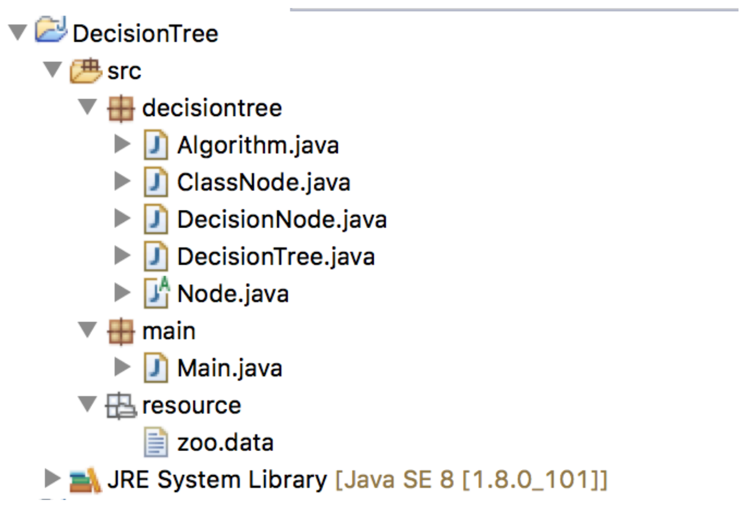
That is to say, ID3 algorithm is when each splitting, calculating the gain rate of each property, and then select the maximum gain rate of the attribute split.

Implementation and Results

1. Implementation

The implementation(FIG 4) of decision tree by using the ID3 algorithm is applied by java.

FIG 4 : Decision Tree Approach

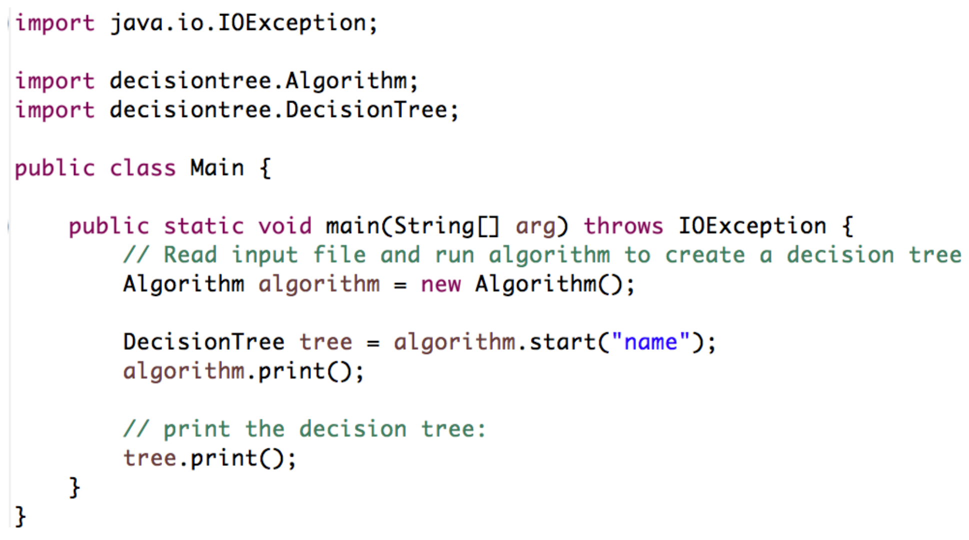


```
▼ DecisionTree
  ▼ src
    ▼ decisiontree
      ▶ Algorithm.java
      ▶ ClassNode.java
      ▶ DecisionNode.java
      ▶ DecisionTree.java
      ▶ Node.java
    ▼ main
      ▶ Main.java
    ▼ resource
        zoo.data
  ▶ JRE System Library [Java SE 8 [1.8.0_101]]
```

The Main(FIG 5) part reads input file and run algorithm to create a decision tree. It contains three parameters:

- A file path
- The target attribute that should be used to create the decision tree
- The separator that was used in the file to separate values (by default it is a space)

FIG 5 : Main



```java
import java.io.IOException;

import decisiontree.Algorithm;
import decisiontree.DecisionTree;

public class Main {

    public static void main(String[] arg) throws IOException {
        // Read input file and run algorithm to create a decision tree
        Algorithm algorithm = new Algorithm();

        DecisionTree tree = algorithm.start("name");
        algorithm.print();

        // print the decision tree:
        tree.print();
    }
}
```

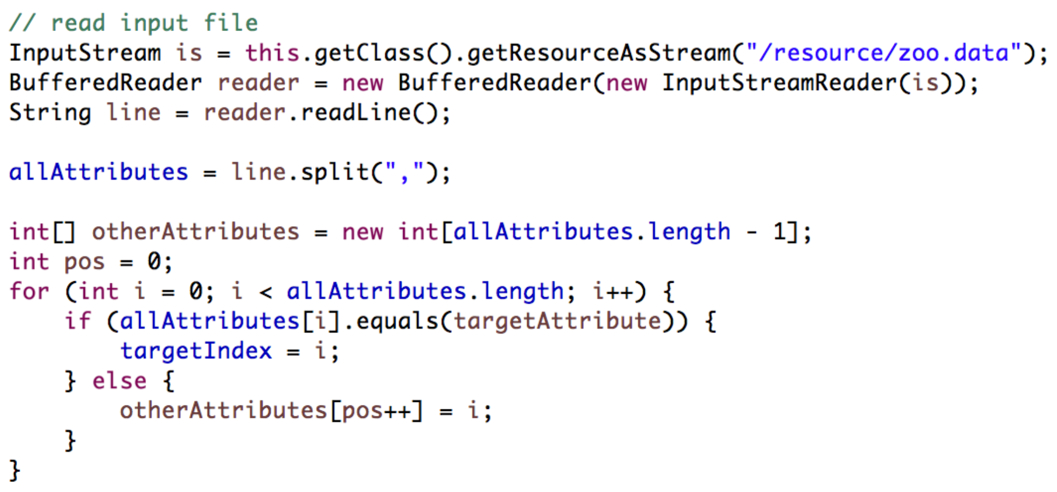
The Algorithm.java class contains parts:

Create a decision tree from a set of training instances, it has three parameters which are input (a path to an input file containing instances), target attribute that be used for classification, and separate. It will return a decision tree.

In the beginning, creating an empty decision tree.

The first step(FIG 6) is to read input file. Read the first line and note the name of the attributes, at the same time, identify the position of the target attribute and other attributes. There is something should noticed. When make an array to store the attributes, if it’s the target attribute, should save the position of the target attribute, otherwise add the attribute to the array of attributes.

FIG 6: First Step- Read input file



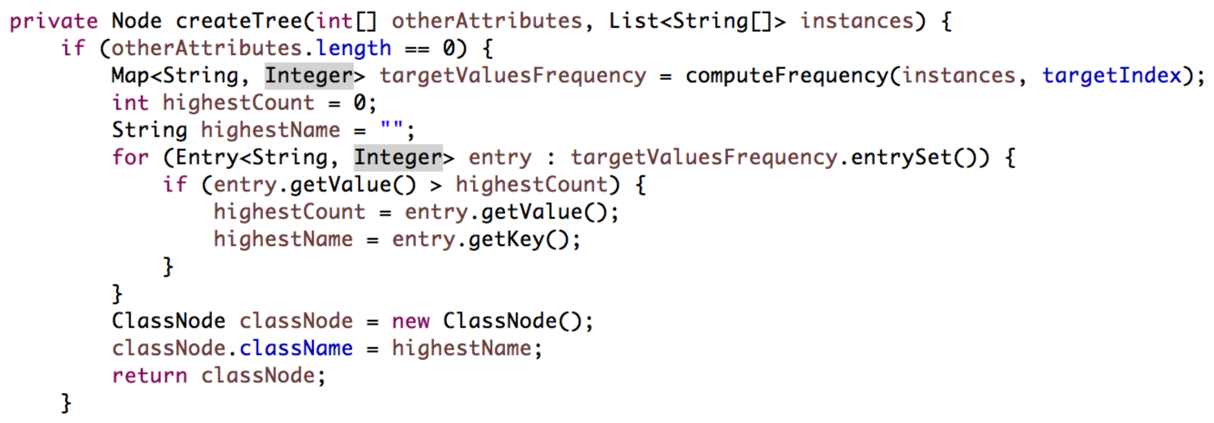
```
// read input file
InputStream is = this.getClass().getResourceAsStream("/resource/zoo.data");
BufferedReader reader = new BufferedReader(new InputStreamReader(is));
String line = reader.readLine();

allAttributes = line.split(",");

int[] otherAttributes = new int[allAttributes.length - 1];
int pos = 0;
for (int i = 0; i < allAttributes.length; i++) {
    if (allAttributes[i].equals(targetAttribute)) {
        targetIndex = i;
    } else {
        otherAttributes[pos++] = i;
    }
}
```

The second step is start the recursive process to create the tree. The method (FIG 7) to create a subtree according to a set of attributes and training instances. The parameters here are otherAttributes which remaining attributes to create the tree and instances that is a list of training instances, and it will return the node of the subtree.

FIG 7 Method to Create a Node of a Subtree

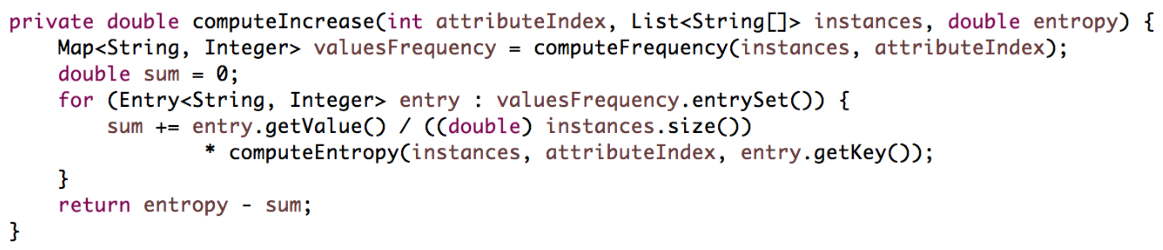


```
private Node createTree(int[] otherAttributes, List<String[]> instances) {
    if (otherAttributes.length == 0) {
        Map<String, Integer> targetValuesFrequency = computeFrequency(instances, targetIndex);
        int highestCount = 0;
        String highestName = "";
        for (Entry<String, Integer> entry : targetValuesFrequency.entrySet()) {
            if (entry.getValue() > highestCount) {
                highestCount = entry.getValue();
                highestName = entry.getKey();
            }
        }
        ClassNode classNode = new ClassNode();
        classNode.className = highestName;
        return classNode;
    }
```

Next, to create a decision node for the attribute.

Then, calculate the information gain of an attribute for a set of instance. It (FIG 8) takes three parameters that are attributeIndex which is the position of the attribute, instances, and the global entropy. It return the gain.

FIG 8 : Calculate The Information Gain

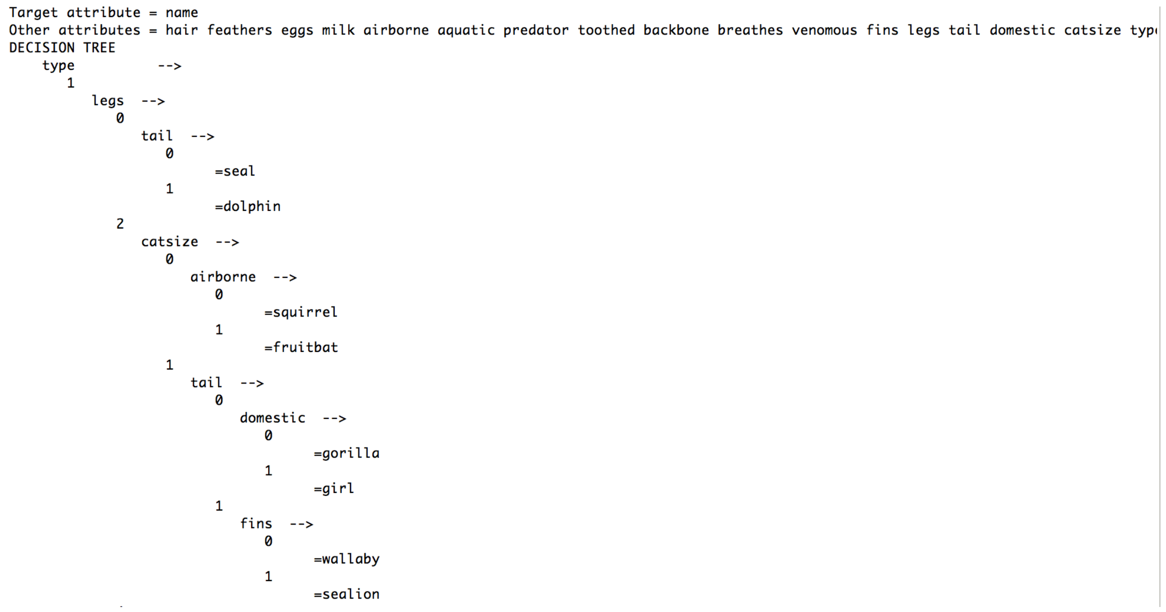


```java
private double computeIncrease(int attributeIndex, List<String[]> instances, double entropy) {
    Map<String, Integer> valuesFrequency = computeFrequency(instances, attributeIndex);
    double sum = 0;
    for (Entry<String, Integer> entry : valuesFrequency.entrySet()) {
        sum += entry.getValue() / ((double) instances.size())
                * computeEntropy(instances, attributeIndex, entry.getKey());
    }
    return entropy - sum;
}
```

2. Results

Below is a part of output of results(FIG 9):

FIG 9: A Part of Result



```
Target attribute = name
Other attributes = hair feathers eggs milk airborne aquatic predator toothed backbone breathes venomous fins legs tail domestic catsize typ
DECISION TREE
    type         -->
       1
          legs  -->
             0
                tail  -->
                   0
                         =seal
                   1
                         =dolphin
             2
                catsize  -->
                   0
                      airborne  -->
                         0
                               =squirrel
                         1
                               =fruitbat
                   1
                      tail  -->
                         0
                            domestic  -->
                               0
                                     =gorilla
                               1
                                     =girl
                         1
                            fins  -->
                               0
                                     =wallaby
                               1
                                     =sealion
```

Conclusion

The decision tree classifier construction doesn't require any domain knowledge or parameter setting, it is appropriate for exploratory knowledge discovery. Decision trees can handle multidimensional data. Their representation of acquired knowledge in tree form is intuitive and generally easy to assimilate. The learning and classifier have good accuracy depend on the data at hand. Decision tree induction algorithms have been used for classification in many application areas such as medicine, manufacturing and production, financial analysis, astronomy, and molecular biology. Decision trees are the basis of several commercial rule induction systems.