IMPLEMENTATION OF A CLASSIFICATION ALGORITHM FOR INSTITUTIONAL ANALYSIS

HONGLIANG SUN
Master of Science, University of Lethbridge, 2008

A Project
Submitted to the School of Graduate Studies
of the University of Lethbridge
in Fulfillment of the
Requirements for the Degree

MASTER OF SCIENCE

Department of Mathematics and Computer Science
University of Lethbridge
LETHBRIDGE, ALBERTA, CANADA

Abstract

The report presents an implemention of a classification algorithm for the Institutional Analysis Project. The algorithm used in this project is the decision tree classification algorithm which uses a gain ratio attribute selection method. The algorithm discovers the hidden rules from the student records, which are used to predict whether or not other students are at risk of dropping out. It is shown that special rules exist in different data sets, each with their natural hidden knowledge. In other words, the rules that are obtained depend on the data that is used for classification. In our preliminary experiments, we show that between 55-78 percent of data with unknown class lables can be correctly classified, using the rules obtained from data whose class labels are known. We feel this is acceptable, given the large number of records, attributes, and attribute values that are used in the experiments. The project results are useful for large data set analysis.

Acknowledgements

I thank all the people who helped me in my education at University of Lethbridge and gave me the opportunity to do research work in this project.

First, I thank Prof. Jackie Rice who guided me into this wonderful university and an interesting research area, I am grateful for her encouragment for me to learn new knowledge and research on power design.

I thank my supervisor Prof. Wendy Osborn who introduced me to this project and helped me to finish it. I appreciate her time patience, and valuable suggestions very much.

I thank my committee members, Prof. Jo-Anne Fiske, Prof. Marc Roussel, Prof. David Siminovitch, and Prof. Howard Cheng who gave me so much useful advice, spent their time to review my project report, and treated me nicely from the very beginning.

I also thank Ms. Barbara Williams who always supported me, helped me and guided me to solve a lot of the study and life problems I met.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Data Mining

The analysis of data to uncover the hidden rules or patterns is an interesting field. Data mining, a part of database system technology, is an advanced data analysis technique to find information and extract patterns (discover knowledge) from data sets [2].

The tasks of data mining can be classified into either descriptive or predictive [2]. The data mining functions specify the kinds of patterns that are mined. The patterns describe the properties of the data (descriptive mining) or make predictions based on the current data (predictive mining).

Classification is one type of predictive data mining. It is performed by building a model based on the training data whose objects' class labels are known. There are various models with several forms, such as classification rules, decision trees, mathematical formulae, or neural networks [2]. Classifiers use these models to predict the class labels for the data set with unknown class labels.

Data mining is part of the knowledge discovery process. The data can be processed by the following steps [2]:

1. Data Preprocessing: the preprocessing step can reduce data noise (such as outliers) and data size for better knowledge discovery results. Data preprocessing includes:

    - Cleaning: removing noise and inconsistent parts;
    - Integration: combining different data sources;
    - Selection: selecting the relevant data from the database;

- Transformation: changing the forms of the data into the ones appropriate for the data mining task by using different operations;

2. Mining: extracting the data rules from the data;

3. Evaluation: evaluating the knowledge the data carry;

4. Presentation: presenting the mined knowledge from the data to the user by using different representation techniques.

We assume that the data we work with is already cleaned, integrated, and selected. In the following sections, we will apply the remaining steps to perform data mining in the Institutional Analysis Project.

1.2 Goal of the Project and the Structure of the Report

The goal of the Institution Analysis Project is to get the patterns of attrition (i.e., patterns that specify students that drop out) from the student records from the University of Lethbridge by using decision tree classification, and then predict the final class labels of other student records based on the rules obtained from the knowledge discovery process. The structure of the report is as follows: in Chapter 2, we introduce some background knowledge for the project, and then use an example to show the calculation in detail of an attribute selection method; in Chapter 3, we explain solutions to some problems which we met during the algorithm implementation and data analysis process; in Chapter 4, we test the algorithm on several different data files and explain the reason for the different test results; and in Chapter 5, we review the whole project and point out the limitations and potential future work.

Chapter 2

Background

The goal of this project is to use decision tree classification to predict which students may be at risk of dropping out for Institutional Analysis at the University of Lethbridge. In this chapter, we introduce the background knowledge for the project.

2.1 Institutional Analysis Project

The Institutional Analysis project involves implementing the decision tree classification algorithm to analyze student records and obtaining the rules for students' drop-out prediction pattern. These rules can be applied to the records of new students to predict if any are at risk of dropping out of university.

Each student record has a class label of 'Y' if the student has dropped out, and 'N' if the student is still registered. Each rule obtained from the records consists of a class label of 'Y' or 'N', and a set of attribute values that lead to an outcome of 'Y' or 'N'. By following the rules obtained from the decision tree, the class labels for other student records (i.e. records that have no class label) can be predicted.

2.2 Entities, Attributes and Tuples

Here we will introduce some basic database concepts which are used in this report.

An *entity* is an object that exists in the world, such as a house or a student. An entity is described by a set of properties. An *entity set* is a set of entities that share the same group of properties. Each unique entity in an entity set is distinguished by the unique values of the

properties. For example, a student can be separated from other students by unique values of the set of the student's properties, such as different values of programs, GPAs, or countries of origin. We call each entity with its property values a *tuple*. In other words, a tuple is the set of property values for a particular entity. The *attributes* are the properties that describe similar information shared by all entities in an entity set. Program, GPA, Country and Residence are attributes. In this project we also have a classification attribute to set the tuple into different classes, e.g., yes (Y) or no (N). A *domain* is a set of all possible values of an attribute.

The training data consist of tuples obtained from a database, which are used to generate the decision tree for classification.

2.3 Decision Tree Classification

A decision tree is used as a classification method for the training tuples. It is a tree structure, where the internal nodes represent the attributes selected at each step; the branches are the values of the attributes; and the leaf nodes are labeled with a class label of the tuples tested under them [3]. The classification rules can be easily obtained by tracing the root-to-leaf paths of the decision tree. The general classification algorithm of the decision tree is introduced below. The inputs to the algorithm are as follows:

1. A set of tuples with their class labels. For this project, the class attribute is "dropped out?". For each tuple, the value of "dropped out?", can be "Y" (the class label is Yes) or "N" (the class label is No) ;

2. A list of candidate attributes that the algorithm can choose from to build the internal nodes of the decision tree;

3. An attribute selection method which selects an attribute for an internal node at each step. For this project it is the gain ratio method [4] which will be explained in Section 2.4.

The output is a decision tree. Each node of the tree represents a subset of tuples that belong to the same path from the root to the node.

The following describes the decision tree classification algorithm [4]:

1. First, generate a node to represent the remaining tuples. In the begining, the node represents the whole tuple set;

2. If all remaining training tuples belong to the same class, label this node as a leaf node with this class;

3. If the attributes list is empty (which means that no attribute can be selected in the next step), then determine the majority class of the training tuples represented by this node (the major/maximum class in this group, e.g., if 3 tuples are labeled ‘Y’ in a group which has 5 tuples, then ‘Y’ is the majority class of the group), and label the node as a leaf node with this class. If no majority exists, then either ‘Y’ or ‘N’ is chosen to handle this situation in all cases;

4. By using the chosen attribute selection method, select the best attribute from the remaining attributes. Label the node with the selected attribute, and then delete this attribute from the candidate attribute list;

5. Split the tuples(*D*) into groups. There will be one group per distinct value of the domain of the selected attribute;

6. If a group is empty, generate a leaf node for this group. Label this node by the majority class of the parent group from which this group is formed;

7. For non-empty groups, repeat the procedure from the first step.

The above steps are performed recursively to build a tree for the training set. The algorithm stops when the terminating conditions are met. Basically, there are three possible termination conditions [2]:

1. All the tuples in a group for a node are in the same class;

2. The attribute list for further selection is empty. Therefore, the node will be labeled as a leaf node and the class will be the majority class in the group of tuples;

3. The group of tuples is empty for the branch of a node. In this case, the newly generated node is a leaf node and labeled with the majority class in the parent tuple group.

The decision tree classifier is one of the simplest methods for classification [2]. Decision tree classification is a fast, generally accurate method for discovering knowledge in many areas. One limitation of decision tree classification is that modifying the tree to incorporate new domain values or attributes is not straightforward. The decision tree must be rebuilt.

2.4 Attribute Selection Measures

2.4.1 Overview of Measures

An attribute selection measure is used to find the attribute which is the most suitable one for partitioning the training tuples into groups. Attribute selection measures may affect the

result of the algorithm during the procedure of tree building. Attribute selection measures are used by the decision tree classifier to partition a group of training tuples into several smaller groups. Ideally, each group should be *pure*, which means all of the tuples in each group should belong to the same class. Based on this requirement, the best attribute for splitting the group of tuples should be chosen so that groups are as pure as possible.

An attribute selection measure will rank the attributes to describe the tuples. The attribute which gets the best score is chosen. There are many different ways to select attributes. Three of them are very popular: information gain [3], gain ratio [4], and gini index [1].

1. Information Gain: This method chooses the attribute which minimizes the information, reflecting best purity in the partitions. So, the number of tests will be the least, thereby building the simplest tree.

2. Gain Ratio: When an attribute (such as student number) has a large number of distinct values, the information gain will prefer to select this attribute over others because it will result in a lot of pure partitions. This is because the information gain calculated for these attributes is close to one, which means that these attributes will almost always be chosen. However, these attributes are useless for classification because each resulting rule will encompass very few tuples. Gain Ratio is a modification to overcome the above disadvantage. It extends the split information needed to describe potential information based on the number of tuples with each value of the attributes.

3. Gini index: It selects the attributes for a binary split method, which divides attribute values into 2 groups. It considers the weight of the impurity of each partition. The attribute which maximizes the purity is selected as the splitting attribute. Because

the tree generated is binary, it may be deeper than others that are generated using a different attribute selection measure.

2.4.2 Gain Ratio Selection Method

In the Institutional Analysis project, gain ratio is chosen as the attribute selection method, because it partitions on multiple attribute values at each step, and at the same time it can handle a large number of distinct attribute values better than information gain based on the reason explained above.

The information present in a tuple group D is as follows [2]:

$$Info(D) = -\sum_{i=1}^{m} p_i \log_2 p_i bits \tag{2.1}$$

where p_i is the probability that a tuple in D belongs to class C_i. Here, D is the whole tuple group, D_i is the subset of tuples from D that belongs to class C_i, and m is the number of classes D has.

For exact classification, the value of $Info_A(D)$ will be needed for each attribute. It is the information needed to classify a tuple of D based on partitioning by attribute A. A smaller value of $Info_A(D)$ stands for greater purity of the partitions [2]:

$$Info_A(D) = \sum_{j=1}^{v} \frac{|D_j|}{|D|} Info(D_j) bits \tag{2.2}$$

where $|D_j|/|D|$ is the weight of the jth partition and v is the number of partitions. The

information gain is the difference of $Info(D)$ and $Info_A(D)$:

$$Gain(A) = Info(D) - Info_A(D) bits \tag{2.3}$$

For v distinct values of attribute A, split information is needed to represent the potential information generated by splitting D:

$$SplitInfo_A(D) = -\sum_{j=1}^{v} \frac{|D_j|}{|D|} \log_2 \left(\frac{|D_j|}{|D|} \right) bits \tag{2.4}$$

Finally, the attribute with the maximum gain ratio will be selected as the splitting attribute [2]:

$$GainRatio(A) = \frac{Gain(A)}{SplitInfo(A)} \tag{2.5}$$

The above steps are repeated on the smaller groups of D until all the groups are processed.

2.5 Example of Training

In this section, by using an example, we show how to build a decision tree by using the gain ratio method to select an appropriate attribute at each step. There are 15 student records, with attributes such as the program, country, GPA, etc. By using the gain ratio to calculate the information among the attributes given, the attributes will be selected one by one for the partition of the student group to get the pure classes for the dropped out records (Table 2.1). As we can see from the Table 2.1, for the attribute "Dropped out?", 7 tuples belong to "N" and 8 tuples belong to "Y". So using Equation (2.1): $Info(D) = -\frac{7}{15}\log_2(\frac{7}{15}) - \frac{8}{15}\log_2(\frac{8}{15}) = 0.996792\ bits$

Table 2.1: Example of Tuples of Students' Records

Number	Program	Country	GPA	Residence	Dropped Out?
1	CPSC	Canada	B	Y	N
2	CPSC	Canada	A	Y	N
3	CHEM	China	C	Y	N
4	CHEM	China	C	N	N
5	ENGL	India	B	Y	Y
6	ENGL	Canada	B	N	Y
7	CHEM	China	B	N	Y
8	FREN	Bangladesh	A	N	Y
9	FREN	Canada	B	Y	Y
10	FREN	Canada	B	Y	N
11	MATH	Canada	B	Y	N
12	MATH	Iran	B	N	N
13	CHEM	India	B	N	Y
14	CPSC	Bangladesh	B	Y	Y
15	CPSC	Bangladesh	B	Y	Y

For each attribute of the set of tuples, the information for each attribute is needed to classify a tuple based on the partitioning by this attribute. As the next step, we will calculate information for each attribute. We should notice that column "Number" is not considered a factor, so no gain ratio is calculated for it. Some attributes such as "Number", only serve as identifiers (e.g., student numbers). They carry little or no information that is useful for classification.

First we calculate the gain ratio for the attribute Program. For the total 15 tuples of Program, 4 of them have "CPSC", and from these 4 tuples, 2 belong to "Y" and 2 belong to "N". We then continue to process the same count on the next Program "CHEM", and so on through all the Programs:

$$Info_{Program}(D) = \frac{4}{15}(-\frac{2}{4}\log_2\frac{2}{4}-\frac{2}{4}\log_2\frac{2}{4})+\frac{4}{15}(-\frac{2}{4}\log_2\frac{2}{4}-\frac{2}{4}\log_2\frac{2}{4})$$
$$+\frac{2}{15}(-\frac{2}{2}\log_2\frac{2}{2})+\frac{3}{15}(-\frac{2}{3}\log_2\frac{2}{3}-\frac{1}{3}\log_2\frac{1}{3})+\frac{2}{15}(-\frac{2}{2}\log_2\frac{2}{2})$$

$= 0.716993\ bits$

Then calculate the difference between them:

$$Gain(Program) = Info(D) - Info_{Program}(D) = 0.279799\,bits$$

For the split information of "Program", we have 4 "CPSC", 4 "CHEM", 2 "ENGL", 3 "FREN", and 2 "MATH", so based on the Equation 2.4, we get the following calculation:

$$SplitInfo_{Program}(D) = -\frac{4}{15}(\log_2\frac{4}{15}) - \frac{4}{15}(\log_2\frac{4}{15}) - \frac{2}{15}(\log_2\frac{2}{15}) - \frac{3}{15}(\log_2\frac{3}{15}) - \frac{2}{15}(\log_2\frac{2}{15}) = 2.25656\ bits$$

Then the last step for one attribute is to get the gain ratio by using the gain and split information:

$$GainRatio(Program) = \frac{Gain_{Program}(D)}{SplitInfo_{Program}(D)} = 0.123993$$

Similarly, the gain ratio is calculated for the attributes Country, GPA and Residence.

For Country:

$$Info_{Country}(D) = \frac{6}{15}(-\frac{4}{6}\log_2\frac{4}{6} - \frac{2}{6}\log_2\frac{2}{6}) + \frac{3}{15}(-\frac{2}{3}\log_2\frac{2}{3} - \frac{1}{3}\log_2\frac{1}{3}) + \frac{2}{15}(-\frac{2}{2}\log_2\frac{2}{2}) + \frac{3}{15}(-\frac{3}{3}\log_2\frac{3}{3}) + \frac{1}{15}(-\frac{1}{1}\log_2\frac{1}{1}) = 0.550978\ bits$$

$$Gain(Country) = Info(D) - Info_{Country}(D) = 0.445814\ bits$$

$$SplitInfo_{Country}(D) = -\frac{6}{15}(\log_2\frac{6}{15}) - \frac{3}{15}(\log_2\frac{3}{15}) - \frac{2}{15}(\log_2\frac{2}{15}) - \frac{3}{15}(\log_2\frac{3}{15}) - \frac{1}{15}(\log_2\frac{1}{15}) = 2.10559\ bits$$

$$GainRatio(Country) = \frac{Gain_{Country}(D)}{SplitInfo_{Country}(D)} = 0.211729$$

For GPA:

$$Info_{GPA}(D) = \frac{11}{15}(-\frac{4}{11}\log_2\frac{4}{11} - \frac{7}{11}\log_2\frac{7}{11}) + \frac{2}{15}(-\frac{1}{2}\log_2\frac{1}{2} - \frac{1}{2}\log_2\frac{1}{2})$$
$$+ \frac{2}{15}(-\frac{2}{2}\log_2\frac{2}{2}) = 0.826818\ bits$$

$$Gain(GPA) = Info(D) - Info_{GPA}(D) = 0.169974\ bits$$

$$SplitInfo_{GPA}(D) = -\frac{11}{15}(\log_2\frac{11}{15}) - \frac{2}{15}(\log_2\frac{2}{15}) - \frac{2}{15}(\log_2\frac{2}{15}) = 1.10331\ bits$$

$$GainRatio(GPA) = \frac{Gain_{GPA}(D)}{SplitInfo_{GPA}(D)} = 0.154059$$

For Residence:

$$Info_{Residence}(D) = \frac{9}{15}(-\frac{4}{9}\log_2\frac{4}{9} - \frac{5}{9}\log_2\frac{5}{9}) + \frac{6}{15}(-\frac{2}{6}\log_2\frac{2}{6} - \frac{4}{6}\log_2\frac{4}{6})$$
$$= 0.961964\ bits$$

$$Gain(Residence) = Info(D) - Info_{Residence}(D) = 0.0348277\ bits$$

$$SplitInfo_{Residence}(D) = -\frac{9}{15}(\log_2\frac{9}{15}) - \frac{6}{15}(\log_2\frac{6}{15}) = 0.970951\ bits$$

$$GainRatio(Residence) = \frac{Gain_{Residence}(D)}{SplitInfo_{Residence}(D)} = 0.0358596$$

So, after the gain ratio for each attribute is calculated, the first attribute we select is Country. Next, we separate the tuples based on the value of this attribute: “India” is pure (all the “Dropped out” values are “Y”), “Bangladesh” is pure, and “Iran” is pure. Therefore, we continue to partition the groups “Canada” and “China”.

Table 2.2: Example of Tuples of Students' Records from Canada

Number	Program	GPA	Residence	Dropped Out?
1	CPSC	B	Y	N
2	CPSC	A	Y	N
6	ENGL	B	N	Y
9	FREN	B	Y	Y
10	FREN	B	Y	N
11	MATH	B	Y	N

For country "Canada" in Table 2.2:

$$Info(D) = -\frac{4}{6}\log_2(\frac{4}{6}) - \frac{2}{6}\log_2(\frac{2}{6}) = 0.918296\ bits$$

$$Info_{Program}(D) = \frac{2}{6}(-\frac{2}{2}\log_2\frac{2}{2}) + \frac{2}{6}(-\frac{1}{2}\log_2\frac{1}{2} - \frac{1}{2}\log_2\frac{1}{2}) + \frac{1}{6}(-\frac{1}{1}\log_2\frac{1}{1})$$
$$+ \frac{1}{6}(-\frac{1}{1}\log_2\frac{1}{1}) = 0.333333\ bits$$

$$Gain(Program) = Info(D) - Info_{Program}(D) = 0.584963\ bits$$

$$SplitInfo_{Program}(D) = -\frac{2}{6}(\log_2\frac{2}{6}) - \frac{2}{6}(\log_2\frac{2}{6}) - \frac{1}{6}(\log_2\frac{1}{6}) - \frac{1}{6}(\log_2\frac{1}{6})$$
$$= 1.9183\ bits$$

$$GainRatio(Program) = \frac{Gain(Program)}{SplitInfo_{Program}(D)} = 0.304939$$

$$Info_{GPA}(D) = \frac{5}{6}(-\frac{2}{5}\log_2\frac{2}{5} - \frac{3}{5}\log_2\frac{3}{5}) + \frac{1}{6}(-\frac{1}{1}\log_2\frac{1}{1}) = 0.809126\ bits$$

$$Gain(GPA) = Info(D) - Info_{GPA}(D) = 0.10917\ bits$$

Table 2.3: Example of Tuples of Students' Records from Canada with "Residence"=Y

Number	Program	GPA	Dropped Out?
1	CPSC	B	N
2	CPSC	A	N
9	FREN	B	Y
10	FREN	B	N
11	MATH	B	N

$$SplitInfo_{GPA}(D) = -\frac{5}{6}(\log_2 \frac{5}{6}) - \frac{1}{6}(\log_2 \frac{1}{6}) = 0.650022\ bits$$

$$GainRatio(GPA) = \frac{Gain(GPA)}{SplitInfo_{GPA}(D)} = 0.167949$$

$$Info_{Residence}(D) = \frac{5}{6}(-\frac{4}{5}\log_2 \frac{4}{5} - \frac{1}{5}\log_2 \frac{1}{5}) + \frac{1}{6}(-\frac{1}{1}\log_2 \frac{1}{1})$$
= 0.601607 *bits*

$$Gain(Residence) = Info(D) - Info_{Residence}(D) = 0.316689\ bits$$

$$SplitInfo_{Residence}(D) = -\frac{5}{6}(\log_2 \frac{5}{6}) - \frac{1}{6}(\log_2 \frac{1}{6}) = 0.650022\ bits$$

$$GainRatio(Residence) = \frac{Gain(Residence)}{SplitInfo_{Residence}(D)} = 0.487197$$

The second attribute selected was "Residence". When "Residence" is N, the partition is pure. Continue partitioning in Table 2.3:

$$Info(D) = -\frac{4}{5}\log_2(\frac{4}{5}) - \frac{1}{5}\log_2(\frac{1}{5}) = 0.721928\ bits$$

$$Info_{Program}(D) = \frac{2}{5}(-\frac{2}{2}\log_2\frac{2}{2}) + \frac{2}{5}(-\frac{1}{2}\log_2\frac{1}{2} - \frac{1}{2}log_2\frac{1}{2}) + \frac{1}{5}(-\frac{1}{1}\log_2\frac{1}{1})$$
= 0.4 *bits*

$$Gain(Program) = Info(D) - Info_{Program}(D) = 0.321928\ bits$$

$$SplitInfo_{Program}(D) = -\frac{2}{5}(\log_2\frac{2}{5}) - \frac{2}{5}(\log_2\frac{2}{5}) - \frac{1}{5}(\log_2\frac{1}{5}) = 1.52193\ bits$$

$$GainRatio(Program) = \frac{Gain(Program)}{SplitInfo_{Program}(D)} = 0.211526$$

$$Info_{GPA}(D) = \frac{4}{5}(-\frac{3}{4}\log_2\frac{3}{4} - \frac{1}{4}\log_2\frac{1}{4}) + \frac{1}{5}(-\frac{1}{1}\log_2\frac{1}{1}) = 0.649022\ bits$$

$$Gain(GPA) = Info(D) - Info_{GPA}(D) = 0.0729056\ bits$$

$$SplitInfo_{GPA}(D) = -\frac{4}{5}(\log_2\frac{4}{5}) - \frac{1}{5}(\log_2\frac{1}{5}) = 0.721928\ bits$$

$$GainRatio(GPA) = \frac{Gain(GPA)}{SplitInfo_{GPA}(D)} = 0.100987$$

The attribute selected is program. “CPSC” is pure. “Math” is pure. Continuing with “French”, the only attribute left is GPA. No majority class exists (the number of tuples with “Y” is equal to the number of tuples with “N”). So, the last node can be labelled with either ’Y’ or ’N’. We indicate this with Y/N in Figure 2.1.

For attribute Country=“China” (in Table 2.4): We have three attributes to choose from: “Program”, “Residence” and “GPA”. This time we choose “GPA”. Then all subsets are pure, and the whole procedure is finished. The decision tree is built as in Figure 2.1.

Table 2.4: Example of Tuples of Students' Records from China

Number	Program	GPA	Residence	Dropped Out?
3	CHEM	C	Y	N
4	CHEM	C	N	N
7	CHEM	B	N	Y

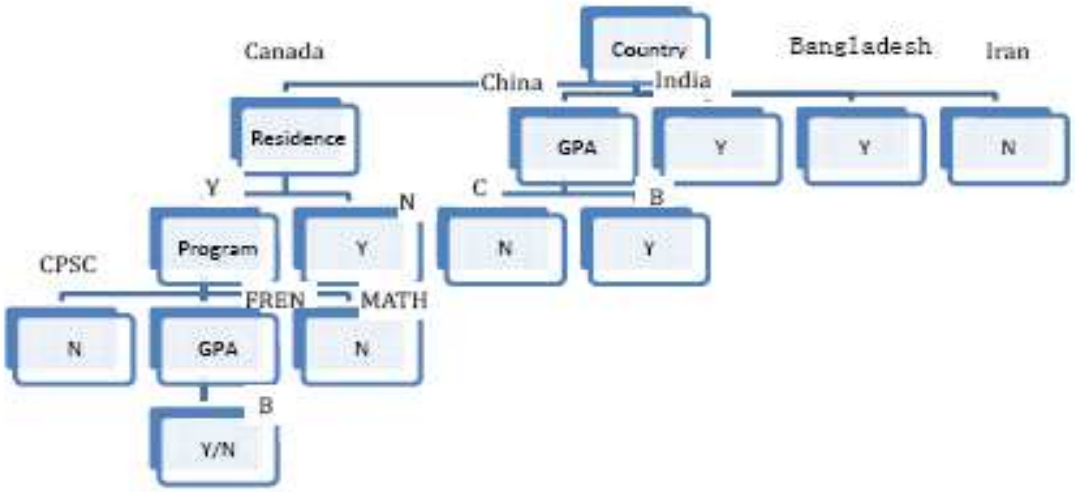

Figure 2.1: The Decision Tree of the Example

2.6 Example of Testing

The testing tuples will follow the tree from the root to the leaves step by step, based on the values of the attributes in the tuples. When the class attribute of the tuple is reached, the testing of this tuple is finished. We will now show examples of testing tuples against the decision tree by following the paths by using the example tuples in Table 2.5. The tuple 1 follows the steps as below:

1. compares the value of tree root node Country, which is Canada;

Table 2.5: Example of classifiable Test Tuples of Students' Records

Number	Program	Country	GPA	Residence	Dropped Out?
1	CPSC	Canada	B	Y	N
2	ENGL	India	B	Y	N

2. follows the branch Canada, reaches the node Residence; compares the value of node Residence, which is Y;

3. follows the branch Y, reaches the node Program; compares the value of node Program, which is CPSC; and

4. follows the branch CPSC, reaches the leaf node: N;

At this point, we can tell that the final class is reached, which is N. So the tuple 1 is classified as N.

The tuple 2 follows the steps as below:

1. compares the value of tree root node Country, which is India;

2. follows the branch India, reach the leaf node: Y;

At this point, we can tell that the final class is reached, which is Y. So the tuple 2 is classified as Y, which is not the same label as the one in the tuple. The first tuple is classified correctly, while the second one is not.

2.7 Example of Rule Tracing

After the decision tree is built, we also need to trace the paths through the tree to obtain the rules.

From the tree, we can get the rules for testing data from leaf to root by using the example from Chapter 2:

1. label:Y,Residence=N; Country=Canada
2. label:N,Program=CPSC; Residence=Y;Country=Canada
3. label:Y,GPA=B; Program=FREN; Residence=Y;Country=Canada
4. label:N,Program=MATH; Residence=Y; Country=Canada
5. label:N,GPA=C; Country=China
6. label:Y,GPA=B; Country=China
7. label:Y,Country=India
8. label:Y,Country=Bangladesh
9. label:N,Country=Iran

Chapter 3

Implementation

In this chapter we will describe some implementation issues in the project and the solutions to these issues.

3.1 Training, Testing, and Predicting

We implemented the following step by step:

1. Preprocess training data. In our project, transformation is used on both the percentage-based and 4-point GPAs to achieve almost the same results with reduced concepts.

2. Read the training tuples into the classification algorithm to build a decision tree by using gain ratio attribute selection.

3. Determine the rules of the decision tree by tracing the paths from the root to each leaf node.

4. Test the test tuples against the rules of the generated decision tree to evaluate the accuracy of the algorithm.

5. Predict the class of future tuples by using the rules of the generated decision tree.

3.2 Tree and Node Structure

For this project, the tree is represented as an array of nodes. We define a C++ class to store all the information of decision tree and nodes, including node index, node label, node

branch, and node parent. Boolean values are used to distinguish for root node and leaf nodes. The definition of class nodeD is as in Figure 3.1:

```
class NodeD {
public:
string label; //after the attribute is selected, the node
               is labeled as the name of attribute;
int order; //the ith node generated;
bool leaf;
bool root;
string branch; //name of branch, linked to the nodes parent
string parent; //name of parent;
vector<string> classtest; // the values of children;
bool nodeset; // judge if already set all the information of the node
bool setparent; //judge if already set the information of nodes parent
} ;
```

Figure 3.1: The Definition of Class nodeD

The decision tree will be built based on all the node information. The parent and branch are used to link the nodes together for the tree structure. Every time a new node is generated, the algorithm increases the node index (M, an integer), and then labels the node by the selected attribute (excluding the leaf nodes, which are labeled by the classes). The branch will be set as the value of last selected attribute which this new node belongs to. Then the algorithm will recursively select an attribute for the training tuples in D, which will generate another new node (N) which is labeled by the new selected attribute, and also the parent of the node N will be set as M. After this step, group D will be separated as D_j by the number of values of the attribute (j). The program then stores the children of this node N. The algorithm will continue on each branch until the terminal condition is met.

During the tree building process, each time the algorithm generates a new leaf node, it means that there is a new path (from leaf to root) generated in the decision tree. So, this

path is a new rule, which is traced in the following way.

Beginning with the leaf node, read the label, the branch and the parent. Next, find the closest preceding node that has the same label as the parent of the leaf node. This is the parent node for the leaf node. The remainder of the path is obtained by doing this recursively until the root is reached. This rule is added to a rule file, which is used to test tuples and predict class labels.

Although strings are used for node identifiers, the above strategy works for the following reasons. First, after a parent node is created, its child nodes (and any of its descendents) are created using recursion, and will be placed in the node array immediately following the parent. Also, because the attribute selected for a node cannot be selected again for any descendent node, it is impossible for more than one node in a subtree to be labeled with the same attribute. Therefore, the parent node will be the closest preceeding node with a matching label.

For example, there are two GPAs node in the example in chapter 2. However, each is selected from different tuple groups and therefore have different ancestors. The first occurrence of GPA is a child of “Program = FREN”, which is a child of “Residence = Y” followed by “Country = Canada”. The second occurrence of GPA is a child of “Country = China”.

Table 3.1 is the node information for the tuples from the example in Chapter 2.

3.3 Mapping

It is difficult to handle continuous data in the decision tree structure. Data preprocessing will deal with continuous data by using data mapping techniques, which are one kind of

Table 3.1: Node Information for the Example of Test Tuples of Students' Records

index	parent	branch	label
0	none	root	Country
1	Country	Canada	Residence
2	Residence	N	Y
3	Residence	Y	Program
4	Program	CPSC	N
5	Program	FREN	GPA
6	GPA	B	Y
7	Program	MATH	N
8	Country	China	GPA
9	GPA	C	N
10	GPA	B	Y
11	Country	India	Y
12	Country	Bangladesh	Y
11	Country	Iran	N

the data reduction technique.

As an example, GPA can be expressed on a percentage basis, or on a four-point scale. The percentage basis GPA will introduce a huge number of values, which will slow the process of tree building and even affect the result for some attribute selection methods because they can change the information gain values. So, using data mapping to reduce the possible values from thousands to less than ten is really an effective solution.

In the project, we map percentage basis GPA into A-F ranges as follows based on the University of Lethbridge system: $A = 80+$, $B = [70,80)$, $C = [60,70)$, $D = [50,60)$, $F = [0,50)$. For 4-point basis GPA, we map as: $A = 3.65+$, $B = [2.65,3.65)$, $C = [1.65,2.65)$, $D = [1.0,1.65)$, $F = [0,1.0)$. Values that are close together but mapped to different final values will not affect classification accuracy. As long as the same A-F range is used all values will be mapped properly. However, accuracy may be an issue if the values of testing tuples are mapped using a different a-F range.

3.4 Store Tuples

For each attribute selection step in the algorithm, each distinct tuple group will be separated and used for the next partition. We generate an empty file for each value of the attribute, and then write the tuples belonging to this value into this file, which is from the tuple group from the previous attribute selected. For the next recursive step, we will process the file to finish the recursive selection. Using memory would be faster, but we lose the opportunities to check the tuples in each file.

3.5 Unclassifiable tuples

After building the tree and getting the rules of the tree, data files will be tested on the tree for the classification prediction of these data files. A test tuple in the data file will follow the rules step by step until it reaches its final class label. Some tuples that are not seen during the training may be unclassifiable.

Testing the class labels of the test tuples is one of the tasks of evaluation. So, how to find unclassifiable tuples and then how to deal with them needs to be handled for this project.

Tuple groups are formed using all domain values of an attribute. So, empty tuple groups can occur during the tree building process. In this case, the node repesenting them is labeled by the majority class of its parent (Section 2.3). Due to the nature of the project, we felt that labeling tuples as unclassifiable if some or all of its values were not seen during training was a better strategy than “guessing” its label, which has normally done. Therefore we modified the partition process by forming tuple groups for the values that appear in the training set, this means that all possible domain values for a partition attribute may not be

Table 3.2: Example of Test Tuples of Students' Records

Number	Program	Country	GPA	Residence	Dropped Out?
2	CHEM	Canada	A	Y	N
3	CHEM	China	C	Y	N

available. So, when predicting the class label of a test tuple, if an attribute value does not exist in the tree as an option, the tuple is instead labelled as unclassifiable.

We will use the tree built by the example in chapter 2 to test the following example in Table 3.2, which will show the tuples that cannot be classified for the above reasons.

Tuple 1 is tested in the steps as below:

1. compare the value of Country, which is Canada;
2. compare the value of Residence, which is Y;
3. compare the value of Program, which is CHEM.

At this point, we can tell that no path matches this tuple. Therefore, tuple 1 cannot be classified by the decision tree, so it is unclassifiable.

The second tuple follows the steps as below:

1. compares the value of Country, which is China;
2. compares the value of GPA, which is C.

At this point, the final class is reached. The second tuple is classified as N. Therefore it is classifiable.

Chapter 4

Evaluation

In this chapter we will perform an empirical evaluation of the implementation of the decision tree classification algorithm and discuss our experimental results.

4.1 Model

All the test files were run on a computer with Intel Pentium 4, 3.00GHz, 1GB of RAM, Linux operating system. The programming language is C++, gcc version 4.1.2 (Red Hat 4.1.2-33), with an optimization level at 0.

4.2 Calculation of Accuracy

We preprocessed the data for tree building, and selected the gain ratio partition method to build a decision tree. The rules were obtained from the decision tree and used to predict data and to get reasonable guesses for the student outcome (knowledge results).

We had four training files to build the decision tree. For each tree, three testing files are used to test how the patterns predict the final class results in different situations.

Accuracy is an important factor to assess the result of data mining. We will use some terms to describe the accuracy for the data analysis for the tree classification [2]:

1. True positives ($tpos$): the positive tuples which were labeled correctly with a positive class.

2. False positives ($fpos$): the negative tuples which were labeled incorrectly, e.g, la-

beled with a positive class.

3. True negatives (*tneg*): the negative tuples which were labeled correctly with a negative class.

4. False negatives (*fneg*): the positive tuples which were labeled incorrectly, e.g, labeled with a negative class.

These four terms are used in the following formulas to determine the accuracy of the algorithm for predicting the class of tuples: *Sensitivity*, also known as "true positive rate"; *Specificity*, the "true negative rate"; and *Precision*, the percentage of positive tuples labeled correctly. These three will be expressed as follows [2]:

$$Sensitivity = \frac{tpos}{pos} \tag{4.1}$$

$$Specificity = \frac{tneg}{neg} \tag{4.2}$$

$$Precision = \frac{tpos}{tpos + fpos} \tag{4.3}$$

The above formulas do not combine the positive and negative tuples together for the method of the analysis. *Accuracy* is a function of *Sensitivity* and *Specificity*, which provides more information on the accurate prediction of data classification for unknown data classes. Accuracy is the probability of choosing true positives and negatives from all positive and negative tuples, or the probability of correct prediction for data. We use the following function for determining *Accuracy* [2]:

$$Accuracy = Sensitivity\frac{pos}{pos + neg} + Specificity\frac{neg}{neg + pos}$$
$$= \frac{tpos + tneg}{pos + neg}$$

The unclassifiable tuples are not used for accurcay calculation.

4.3 Evaluation of Correctness

Several training and testing files were used to illustrate the correctness of the classification algorithm in different situations:

1. Same file as the training test file;
2. The file in which all tuples can be classified but at the same time may have false positives and negatives;
3. The file in which some tuples may be unclassified.

We tested four groups of data by using the above formulas. Each group contains four files: one file to build the tree, and three files to test against the tree for predicting. Each file contains 15 tuples, which is enough for our preliminary experiments. Table 4.1 summarizes the results. “unt” means “unclassified tuples”, “acc” means “accuracy”.

The test 0 files are designed exactly the same as the data files used to build the decision tree. The test 1 files contain many misjudged false positives and false negatives, which means these situations are handled inefficiently by the decision tree classification using the gain ratio attribute selection method. The test 2 files are designed to contain some tuples which cannot be classified by the tree, but all of the classified tuples are usually accurately labeled.

For the first tree, there is one false positive tuple in test file 0. This is because when we built the tree, there are equal numbers of “Y” and “N” tuples, and no more attributes were

Table 4.1: Experimental Results for the Test Files of Students' Records

file	unt	tpos	fpos	tneg	fneg	acc
test0	0	8	0	6	1	0.933333
test1	0	5	4	4	2	0.6
test2	3	6	1	3	2	0.75
test0	0	8	0	7	0	1
test1	0	4	4	4	3	0.533333
test2	3	7	0	5	0	1
test0	0	8	0	7	0	1
test1	0	4	4	4	3	0.533333
test2	2	6	0	7	0	1
test0	0	4	0	11	0	1
test1	0	0	4	5	6	0.333333
test2	1	4	0	10	0	1

available for further partition. For this leaf node, we use "Y" to label it. So when we test the file on the tree, the "N" tuples under this node will be treated as false positive tuples because they are classified as "Y" by the tree.

Except for the test 0 file for group 1, we can see that all other test 0 files obtain 100 percent accuracy, which is expected because they are the ones used to build the decision trees.

All test 1 files, as mentioned before, have false positive and false negative tuples as judged by the tree classification, but all tuples in the data files can be labeled. It means each tuple in the file can be traced by the tree and have its class determined. The accuracy of the test 1 files is low because there is a significant number of the falsely classified tuples.

All test 2 files contain some tuples that cannot be classified by the tree, but all other tuples usually can be correctly labeled. But as we can see, the first test file still has false positive and false negative tuples. As we explained in the previous chapter, in certain situations tuples cannot be classified by the tree classifier. So in this testing group, although

Table 4.2: Experimental Results for the Test Files of Transfer Student Records

file	training	test	unt	tpos	fpos	tneg	fneg	acc
test1	1000	3000	684	92	603	1236	322	0.589436
test2	2000	2000	522	108	79	1038	253	0.775372
test3	3000	1000	166	107	237	432	58	0.646283

unclassifiable tuples are found, high accuracy is usually obtained.

4.4 Evaluation of Accuracy

We also evaluate the accuracy of classification using a large data set of student records. The data file is sorted by the attribute “Major”, and tuples are selected from the top portion of the file. This allows us to easily select a group of tuples with a random number of “Y” and “N” class labels.

We use three groups of files. The first group uses 1000 tuples from the begining of data file to build the tree, and then use 3000 different tuples as a test file to predict the accuracy, which is obtained by comparing the final class label with the “Dropped Out?” class label for each tuple in the test file. The second group uses 2000 tuples from the begining of data file to build the tree and then another 2000 tuples to predict; the third one uses 3000 tuples from the begining of data file to build the tree and then 1000 different tuples to predict. Table 4.2 shows the accuracy of predicting large data sets of student records. In the table, “ training” is the number of tuples used to build the decision tree, “ test” is the number of tuples used to test the accuracy against the tree.

The accuracy is mainly determined by the percentage of the true positive and the true negative tuples found in the data files, which depends on the nature of the data. Usually, the accuracy is acceptable which means the tree is useful for data prediction. More testing

Table 4.3: Timing Results for the Test Files of Transfer Students' Records

number of tuples	nodes generated	time(s)
100	39	0.226
200	60	0.506
300	220	1.258
400	277	1.977
500	340	2.965
1000	883	12.585
2000	1400	37.944

Table 4.4: Timing Results for the Test Files of High School Students' Records

number of tuples	nodes generated	time(s)
100	3	0.072
200	3	0.135
300	5	0.396
400	5	0.602
500	73	2.435
1000	488	11.393
2000	2007	52.021
3000	2160	83.097

is necessary for the evaluation of accuracy.

4.5 Evaluation of Running Time

The running time experiments show the running time when different number of tuples are used to build a decision tree. The data were chosen so that 50 percent of the records were labeled as "Y" and 50 percent were labeled as "N". We do not have enough tuples labeled as "N" for the transfer students data, so the 3000 tuples experiment was not performed in the running experiment of transfer students' records.

We can see that the running time increases when the number of nodes generated in-

creases. Also, the tree nodes generated increase with the number of the input testing data. With the increasing number of values of each attribute which can be chosen from, then the calculations needed for the gain ratios for each round become more time consuming. On the other hand, the number of nodes generated is mainly determined by the data set, which means it is not necessarily proportional to the number of tuples. For example: in Table 4.3, 100 tuples leads to 39 nodes while in Table 4.4, 100 tuples leads to only 3 nodes, one root and two leaves. It is because only one rule is found in this file: the students who take student loans will not drop out. So only two leaf nodes are generated, the tuples for which loan is "N" (which means money is borrowed) belong to the node "Y" for "Dropped out?"; the tuples which loan are "Y" (which means money is not borrowed) belong to the node "N" for "Dropped out?". For another example, in Table 4.3, 2000 tuples generates 1400 nodes while in Table 4.4, 2000 tuples generates 2007 nodes. In the first set, there are a lot of tuples that share the same paths of the tree. But in the second one, a lot of individual tuples generate their own path, so in the end, the number of nodes generated is more than the number of tuples in the file.

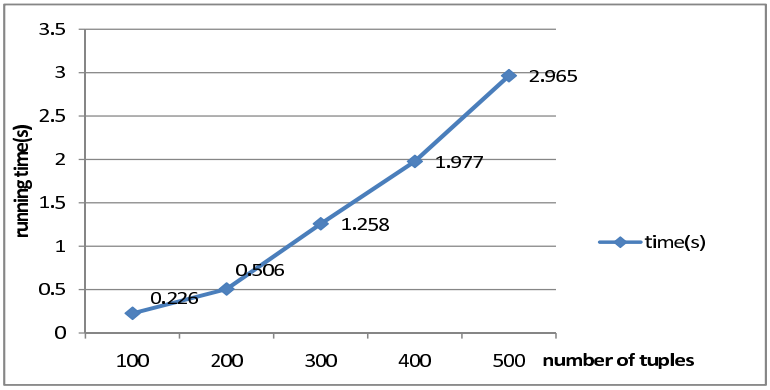

Figure 4.1: The Running Experiment of Transfer Students' Records part 1

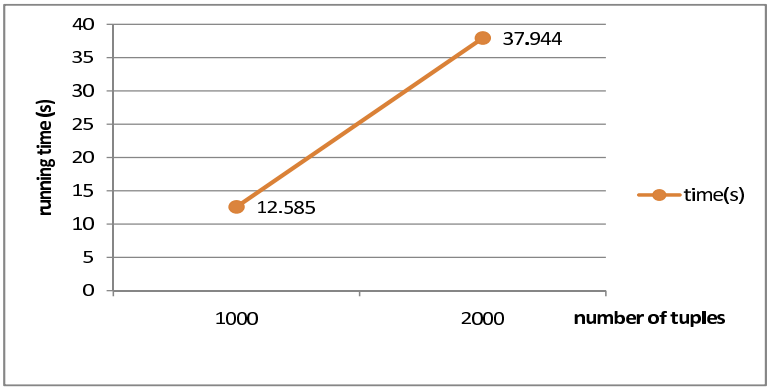

Figure 4.2: The Running Experiment of Transfer Students' Records part 2

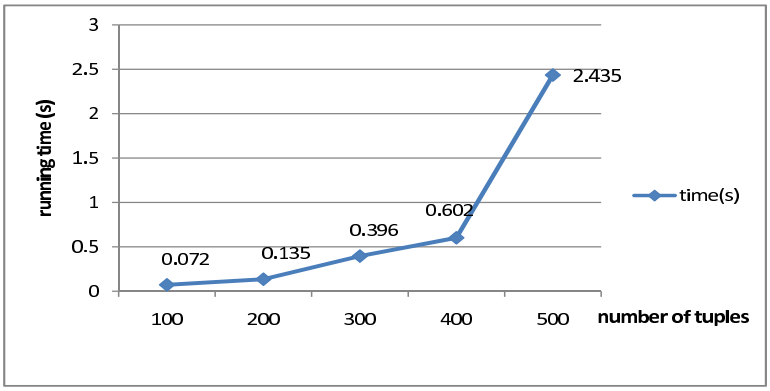

Figure 4.3: The Running Experiment of High School Students' Records part 1

4.6 Evaluation of Complexity

Assume that the tuple group D has n attributes describing the tuples in it. In the n attribute selection loop, select an attribute for $|D|$ (the number of tuples in D) tuples, and then search values of attributes and then add nodes to the tree, which is $O(log|D|)$. So the computational complexity of the decision tree classification algorithm is $O(n \times |D| \times log|D|)$ [2].

We also obtain estimation equations for transfer students' records and high school students' records. Assume that $t \sim b|D|^k$, where t is time and $|D|$ is number of tuples. We use

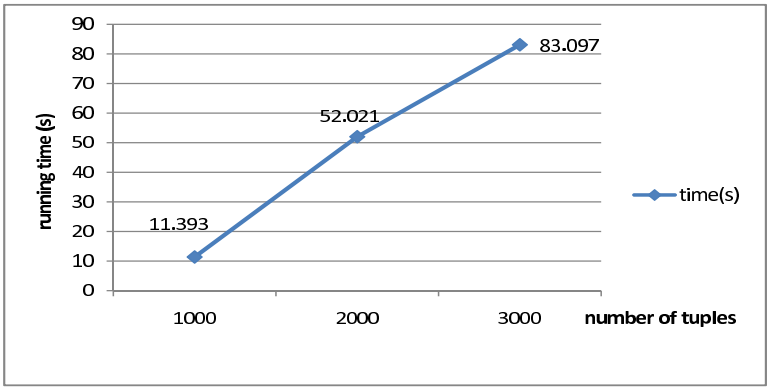

Figure 4.4: The Running Experiment of High School Students' Records part 2

logarithm on both sides: $\log_{10} t = \log_{10} b + k \log_{10} |D|$. For high school student records, we get the estimation equation $\log_{10} t = 2.57 \log_{10} |D| - 7.05$ by choosing two points in the Figure of the running experiment of high school students' records. Using the same method, we can get the estimation equation for transfer student's records: $\log_{10} t = 1.8 \log_{10} |D| - 4.36$. The two estimation equations are different, which is because the slopes of the running times for transfer students' record and high school students' records are different. This is shown in the Figure 4.5.

The estimation equation can be used for larger values of tuples but can not always be useful. There are situations where a larger number of tuples can build a smaller tree with less nodes than a smaller number of tuples. This is because the tree is determined by the nature of chosen tuples, not the number of tuples. For example, with the increase in the number of tuples, the best attribute selected for each step can change. Therefore, the tree is also changed, with a simpler tree being generated, compared with those created with a smaller number of training tuples.

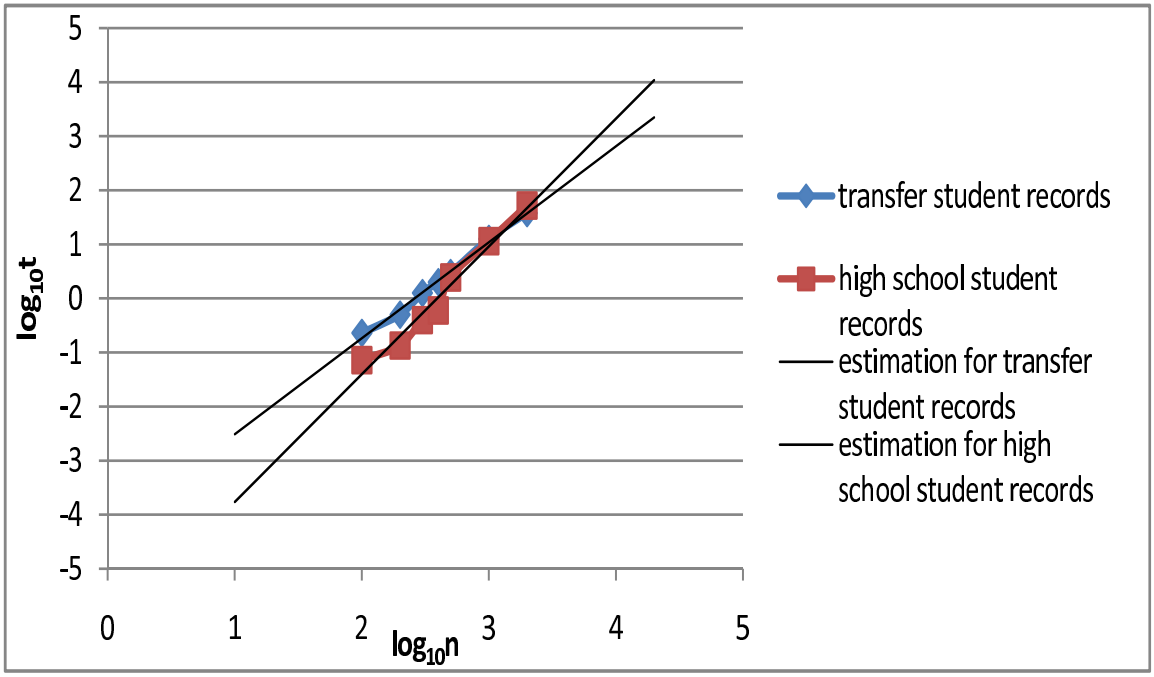

Figure 4.5: The estimation equation of student records

Chapter 5

Conclusion

5.1 Review

The goal of the project was to analyze the data of student records and generate predictions from the data based on the classifiers generated from the decision tree classification algorithm. We preprocessed the data which were used to build the tree. Also we use the gain ratio attribute selection method when building the decision tree for classification. From there we generated the rules or knowledge patterns based on the classifiers and the tree paths. Finally we used the rules of data files to predict the class labels for other unclassified data.

5.2 Application

This work will be used by Institutional Analysis for student record prediction. Using the students' records where the class label is known (i.e., whether they dropped out), the decision tree will be built and then used to predict the final class labels for another file of student records who are still registered to predict if any students are at risk for dropping out. The rules obtained from the different training files are different based on the information hidden in the data. Also, the rules only can be applied to the data which have the same structure as the training data used to build the decision tree.

5.3 Future Improvement

Memory management is one aspect that we can improve upon in this project. In the algorithm, arrays were used to store data for the tree building process. This required a certain amount of continuous space (but more than enough) for each required array which results in a waste of the memory. So as the number of new arrays is increased, so does wasted memory. If we change these arrays to linked lists, the efficiency of the memory usage will be improved, because the continuous memory is no longer required. Better yet, use pointers to appropriate node objects, and use inheritance so each node is of the right kind.

Secondly, the design of the classification algorithm implementation can be improved. Reasonable recursive processes which can handle more operations at the same time will be helpful for memory saving, for example, the searching of correct class labels for different tuples groups.

The functions also need to be handled more efficiently. For example, there are several functions for dealing with different file reading requirements. Since the central code of these functions is the same, these parts can be separated and made into one function which can be called whenever needed.

5.4 Limitations and Future Work

There are additional limitations which can be taken care of in future work, even for the classification algorithm. This algorithm is too particular, which means it generates too many details. For example, it will unnecessarily generate more nodes and paths in the tree to hold some special cases which may just arise from one training tuple. Ultimately, this will make the generated tree much more complex than people need. The branches

grown for these special tuples carry little information. As an important part of future work, the algorithm must be modified for this problem. One potential solution is the following: the program can count the number of tuples under each group after an attribute has been selected, and if the number of tuples is less than a certain percentage of the total number of tuples in the data file, then a leaf node can be generated based on the majority class in this group.

Bibliography

[1] L. Breiman, J. Friedman, R. Olshen, and C. Stone. *Classification and Regression Trees*. Wadsworth and Brooks, Monterey, CA, 1984.

[2] J Han and M Kamber. *Data Mining: Concepts and Techniques (The Morgan Kaufmann Series in Data Management Systems)*. Morgan Kaufmann, San Francisco, CA, 2000.

[3] J. R. Quinlan. Induction of decision trees. *Mach. Learn.*, 1(1):81–106, 1986.

[4] J. R Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Francisco, CA, 1993.