A
Dissertation
On

The base strategy of ID3 algorithm of Data mining using Havrda and Charvat Entropy

Submitted in Partial fulfillment of the requirements
for the award of the degree of
MASTER OF ENGINEERING
in
(Computer Technology & Application)

Submitted By:
NISHANT MATHUR
College Roll No: 10/CTA/08
University Roll No. 8408

Under the Guidance of:
Mrs. Rajni Jindal
Dept. of Computer Engineering
Delhi College of Engineering, Delhi

DEPARTMENT OF COMPUTER ENGINEERING
DELHI COLLEGE OF ENGINEERING
DELHI UNIVERSITY
2010

CERTIFICATE

DELHI COLLEGE OF ENGINEERING
DELHI UNIVERSITY
2009-2010

This is to certify that the work contained in this dissertation entitled " The base strategy of ID3 algorithm of Data mining using Havrda and Charvat Entropy" , submitted by Nishant Mathur**,** University Roll No-8408 in the requirement for the partial fulfilment for the major project in Master of Engineering in Computer Technology & Application, Delhi College of Engineering is an account of his work carried out under my guidance and supervision in the academic year 2009-2010.

Mrs. Rajni Jindal
Assistant Professor
Project Guide
Dept. Of Computer Engineering
Delhi College of Engineering, Delhi

ACKNOWLEDGEMENT

It is a great pleasure to have the opportunity to extend my heartiest felt gratitude to everybody who helped me throughout the course of this project.

It is distinct pleasure to express my deep sense of gratitude and indebtedness to my learned supervisor **Mrs. Rajni Jindal** for her invaluable guidance, encouragement and patient reviews. With her continuous inspiration only, it becomes possible to complete this dissertation.

I would also like to take this opportunity to present my sincere regards to all the faculty members of the Department for their support and encouragement.

I am grateful to my parents for their moral support all the time, they have been always around to cheer me up, in the odd times of this work. I am also thankful to my classmates for their unconditional support and motivation during this work. Last but not the least, special thanks to the people who are active in the field of Data Mining.

NISHANT MATHUR
Master in Engineering
(Computer Technology & Application)
College Roll No. 10/CTA/08
University Roll No. 8408
Department of Computer Engineering
Delhi College of Engineering,
Bawana Road, Delhi-110042

ABSTRACT

D*ata mining algorithm* is the mechanism that creates a data mining model. To create a model, an algorithm first analyzes a set of data and looks for specific patterns and trends. The algorithm uses the results of this analysis to define the parameters of the mining model. These parameters are then applied across the entire data set to extract actionable patterns and detailed statistics.

Decision tree algorithm is used to select the best path to follow in the standard division. This thesis introduces the use of ID3 algorithm of decision tree. Instead of using Shannon Entropy, we use Havrda and Charvat Entropy. By computing information we set particular property from taken data as root of tree, also sub-root by repeating the process continually, to finally build the most optimized tree. This decision tree helps to take the decision for better analysis of data. We have implemented proposed algorithm in "C" programming language.

Table of Contents Page no.

CERTIFICATE……..……..…..ii

ACKNOWLEDGEMENT……..……..…..…iii

ABSTRACT……..…..iv

List of Figures……..…… ……...……...…vii

List of Tables…….……viii

1. Introduction of Data Mining…….……..1
   1.1 Overview ……..…....1
      1.1.1 The Foundation of Data Mining…….....1
      1.1.2 Application of Data Mining…….....4

   1.2 How Data Mining works..……...…….....10
      1.2.1 ID3 Basic…….....11
      1.2.2 Current concerns…….....12
      1.2.3 Problem Statement…….……12

   1.3 Organisation of report…….…....13

2. Design approaches and evaluation methods…….……15
   2.1 Our existing architecture……. 17
      2.1.1 Baseline Framework……...…….19

   2.2 Other approaches to focused Data mining……...…….20

   2.3 Evaluation strategies……...……..21

3. Classification algorithm.……..……..….23
   3.1 Introduction ……24

   3.2 Different types of classification algorithm…….…...…....25
      3.2.1 Statistical based algorithm…….…..…25
      3.2.2 Naïve Bayes classifier…….....25
      3.2.3 K-nearest neighbour algorithm…….....26
      3.2.4 Artificial neural network…….....27

4. Decision tree based algorithm……..……..……29
4.1 Representation and learning…….…...…...…30

4.2 ID3 algorithm…….…...…....31

4.3 C4.5 algorithm……...……...……..32

4.4 CART algorithm……..…...33

5. User driven Data mining…….....36
5.1 Issues in Id3 algorithm……...…....37

5.2 Pruning Decision Trees and Deriving Rule Sets.....40

6. The Havrda and Charvat Entropy…….....42
6.1 Definition and Role of Havrda and Charvat Entropy.....43

6.2 Using Havrda and Charvat Entropy in ID3 algorithm….....44
6.2.1 Analysis of new algorithm…….…….....44
6.2.2 Calculation of information gain……...….47
6.2.3 Generation of Decision Tree……...……....53

7. Implementation……...…….……..54
7.1 Coding……..……..55

7.2 Explanation of Coding……...……...…67

7.3 Output……...……...…68
7.3.1 In C compilation form……...…68
7.3.2 In Decision Tree form……...…….69

8. Conclusion and Future Work……...……....70
8.1 Conclusion……...……...…...……71

8.2 Scope for Future work…….71

9 Bibliography……...…….…….....72

List of Figures

Figure 2.1 (a) Traditionally Decision trees……..……..…16

Figure 4.1 (a) Decision Tree showing types of nodes……..……..30

Figure 5.1 (a) Decision tree for weather conditions……..……39

Figure 5.1 (b) Decision tree for stock market conditions…….……..39

Figure 6.2 (a) Decision tree for Customer Dispatch Goods.….....53

Figure 7.3 (a) Output in C compilation format……..…..…...68

Figure 7.3 (b) Output in Decision Tree format…….....69

List of Tables

2.1(a) Attribute table in existing architecture……..……..17
2.1(b) Training data table in existing architecture……..……..18
5.1 (a) Attribute table in Issues in Id3 algorithm…….…….38
5.1 (b) Training data table in Issues in Id3 algorithm……..38
6.2 (a) The Information of Customer Dispatch Goods……...……44
6.2 (b) The Information of Customer Dispatch Goods for freight fees (<100)…….……48
6.2 (c) The Information of Customer Dispatch Goods for freight fees (100-1000)……...……50
6.2 (d) The Information of Customer Dispatch Goods for freight fees (>1000)……52

CHAPTER 1

AN INTRODUCTION TO DATA MINING

1.1 Overview

Data mining, *the extraction of hidden predictive information from large databases,* is a powerful new technology with great potential to help companies focus on the most important information in their data warehouses[1]. Data mining tools predict future trends and *behaviours*, allowing businesses to make proactive, knowledge-driven decisions. The automated, prospective analyses offered by data mining move beyond the analyses of past events provided by retrospective tools typical of decision support system. Data mining tools can answer business questions that traditionally were time consuming to resolve. They scour database for hidden patterns, finding predictive information that experts may miss because it lies outside their expectations.

Most companies already collect and refine massive quantities of data. Data mining techniques can be implemented rapidly on existing software and hardware platform to enhance the value of existing information resources, and can be integrated with new product and systems as they are brought on-line. When implemented in high performance client/server or parallel processing computer, data mining tools can analyze massive database to deliver answers to questions such as, “which clients are most likely to respond to promotional mailing, and why?”

1.1.1 The Foundations of Data Mining

Humans have been "manually" extracting patterns from data for centuries, but the increasing volume of data in modern times has called for more automatic approaches. Early methods of identifying patterns in data include Bayes' theorem (1700s) and Regression analysis (1800s). The proliferation, ubiquity and increasing power of computer technology has increased data collection and storage. As data sets have grown in size and complexity, direct hands-on data analysis has increasingly been augmented with indirect, automatic data processing. This has been aided by other

[1] http://en.wikipedia.org/wiki/decision tree.
Reference paper no.[3] & [7]

discoveries in computer science, such as Neural networks, Clustering, Genetic algorithms (1950s), Decision trees(1960s) and Support vector machines (1980s). Data mining is the process of applying these methods to data with the intention of uncovering hidden patterns.[2] It has been used for many years by businesses, scientists and governments to sift through volumes of data such as airline passenger trip records, census data and supermarket scanner data to produce market research reports. (Note, however, that reporting is not always considered to be data mining).

A primary reason for using data mining is to assist in the analysis of collections of observations of behaviour. Such data are vulnerable to collinearity because of unknown interrelations. An unavoidable fact of data mining is that the (sub-)set(s) of data being analysed may not be representative of the whole domain, and therefore may not contain examples of certain critical relationships and behaviours that exist across other parts of the domain. To address this sort of issue, the analysis may be augmented using experiment-based and other approaches, such as Choice Modelling for human-generated data. In these situations, inherent correlations can be either controlled for, or removed altogether, during the construction of the experimental design.

There have been some efforts to define standards for data mining, for example the 1999 European Cross Industry Standard Process for Data Mining (CRISP-DM 1.0) and the 2004 Java Data Mining standard (JDM 1.0). These are evolving standards; later versions of these standards are under development. Independent of these standardization efforts, freely available open-source software systems like **RapidMiner, Weka, KNIME,** have become an informal standard for defining data-mining processes. Most of these systems are able to import and export models in PMML (Predictive Model Markup Language) which provides a standard way to represent data mining models so that these can be shared between different statistical applications. PMML is an XML-based language developed by the Data Mining Group (DMG), an independent group composed of many data mining companies. PMML version 4.0 was released in June 2009.

Data mining commonly involves four classes of task:

Classification - Arranges the data into predefined groups. For example an email program might attempt to classify an email as legitimate or spam. Common algorithms include nearest neighbor, Naive Bayes classifier and Neural network.

Clustering - Is like classification but the groups are not predefined, so the algorithm will try to group similar items together.

Regression - Attempts to find a function which models the data with the least error. A common method is to use Genetic Programming.

Association rule learning - Searches for relationships between variables. For example a supermarket might gather data of what each customer buys. Using association rule learning, the supermarket can work out what products are frequently bought together, which is useful for marketing purposes. This is sometimes referred to as "market basket analysis".

1.1.2 Application of Data Mining[2]

Surveillance

Previous data mining to stop terrorist programs under the U.S. government include the Total Information Awareness (TIA) program, Secure Flight (formerly known as Computer-Assisted Passenger Pre-screening System (CAPPS II)), Analysis, Dissemination, Visualization, Insight, Semantic Enhancement (ADVISE), and the Multistate Anti-Terrorism Information Exchange (MATRIX).These programs have been discontinued due to controversy over whether they violate the US Constitution's 4th amendment, although many programs that were formed under them continue to be funded by different organizations, or under different names.

[2] References paper no. [12]

Two plausible data mining techniques in the context of combating terrorism include "pattern mining" and "subject-based data mining".

Pattern mining

"Pattern mining" is a data mining technique that involves finding existing patterns in data. In this context *patterns* often means association rules. The original motivation for searching association rules came from the desire to analyze supermarket transaction data, that is, to examine customer behaviour in terms of the purchased products. For example, an association rule "beer => crisps (80%)" states that four out of five customers that bought beer also bought crisps.

In the context of pattern mining as a tool to identify terrorist activity, the National Research Council provides the following definition: *"Pattern-based data mining looks for patterns (including anomalous data patterns) that might be associated with terrorist activity — these patterns might be regarded as small signals in a large ocean of noise."* Pattern Mining includes new areas such a Music Information Retrieval (MIR) where patterns seen both in the temporal and non temporal domains are imported to classical knowledge discovery search techniques.

Subject-based data mining

"Subject-based data mining" is a data mining technique involving the search for associations between individuals in data. In the context of combating terrorism, the National Research Council provides the following definition: "Subject-based data mining[3] uses an initiating individual or other datum that is considered, based on other information, to be of high interest, and the goal is to determine what other persons or financial transactions or movements, etc., are related to that initiating datum."

[3] Advanced data mining and applications: third international conference, ADMA By Reda Alhajj

Games

Since the early 1960s, with the availability of oracles for certain combinatorial games, also called table bases (e.g. for 3x3-chess) with any beginning configuration, small-board dots-and-boxes, small-board-hex, and certain endgames in chess, dots-and-boxes, and hex; a new area for data mining has been opened up. This is the extraction of human-usable strategies from these oracles. Current pattern recognition approaches do not seem to fully have the required high level of abstraction in order to be applied successfully. Instead, extensive experimentation with the table bases, combined with an intensive study of table base-answers to well designed problems and with knowledge of prior art, i.e. pre-table base knowledge, is used to yield insightful patterns. Berlekamp in dots-and-boxes etc. and John Nunn in chess endgames are notable examples of researchers doing this work, though they were not and are not involved in table base generation.

Business

Data mining in customer relationship management applications can contribute significantly to the bottom line. Rather than randomly contacting a prospect or customer through a call centre or sending mail, a company can concentrate its efforts on prospects that are predicted to have a high likelihood of responding to an offer. More sophisticated methods may be used to optimize resources across campaigns so that one may predict which channel and which offer an individual is most likely to respond to — across all potential offers. Additionally, sophisticated applications could be used to automate the mailing. Data clustering can also be used to automatically discover the segments or groups within a customer data set. Data mining can also be helpful to human-resources departments in identifying the characteristics of their most successful employees. Information obtained, such as universities attended by highly successful employees, can help HR focus recruiting efforts accordingly.

Additionally, Strategic Enterprise Management applications help a company translate corporate-level goals, such as profit and margin share targets, into operational decisions, such as production plans and workforce levels.

Another example of data mining, often called the market basket analysis, relates to its use in retail sales. If a clothing store records the purchases of customers, a data-mining system could identify those customers who favour silk shirts over cotton ones. Although some explanations of relationships may be difficult, taking advantage of it is easier. The example deals with association rules within transaction-based data. Not all data are transaction based and logical or inexact rules may also be present within a database. In a manufacturing application, an inexact rule may state that 73% of products which have a specific defect or problem will develop a secondary problem within the next six months.

Market basket analysis has also been used to identify the purchase patterns of the Alpha consumer. Alpha Consumers are people that play a key roles in connecting with the concept behind a product, then adopting that product, and finally validating it for the rest of society. Analyzing the data collected on these type of users has allowed companies to predict future buying trends and forecast supply demands.

Data Mining is a highly effective tool in the catalog marketing industry. Catalogers have a rich history of customer transactions on millions of customers dating back several years. Data mining tools can identify patterns among customers and help identify the most likely customers to respond to upcoming mailing campaigns.

Related to an integrated-circuit production line, an example of data mining is described in the paper "Mining IC Test Data to Optimize VLSI Testing". In this paper the application of data mining and decision analysis to the problem of die-level functional test is described. Experiments mentioned in this paper demonstrate the ability of applying a system of mining historical die-test data to create a probabilistic model of patterns of die failure which are then utilized to decide in real time which die to test next and when to stop testing. This system has been shown, based on experiments with historical test data, to have the potential to improve profits on mature IC products.

Science and engineering

In recent years, data mining has been widely used in area of science and engineering, such as bioinformatics, genetics, medicine, education and electrical power engineering.

In the area of study on human genetics, the important goal is to understand the mapping relationship between the inter-individual variation in human DNA sequences and variability in disease susceptibility. In lay terms, it is to find out how the changes in an individual's DNA sequence affect the risk of developing common diseases such as cancer. This is very important to help improve the diagnosis, prevention and treatment of the diseases. The data mining technique that is used to perform this task is known as multifactor dimensionality reduction.

In the area of electrical power engineering, data mining techniques have been widely used for condition monitoring of high voltage electrical equipment. The purpose of condition monitoring is to obtain valuable information on the insulation's health status of the equipment. Obviously, different tap positions will generate different signals. However, there was considerable variability amongst normal condition signals for the exact same tap position. SOM has been applied to detect abnormal conditions and to estimate the nature of the abnormalities.

whereby descriptors of human expertise are extracted, normalized and classified so as to facilitate the finding of experts, particularly in scientific and technical fields. In this way, data mining can facilitate Institutional memory.

Other examples of applying data mining technique applications are biomedical data facilitated by domain ontology, mining clinical trial data, traffic analysis using SOM, etcetera.

In adverse drug reaction surveillance, the Uppsala Monitoring Centre has, since 1998, used data mining methods to routinely screen for reporting patterns indicative of emerging drug safety issues in the WHO global database of 4.6 million suspected

adverse drug reaction incidents. Recently, similar methodology has been developed to mine large collections of electronic health records for temporal patterns associating drug prescriptions to medical diagnoses.

Spatial Data mining

Spatial data mining is the application of data mining techniques to spatial data. Spatial data mining follows along the same functions in data mining, with the end objective to find patterns in geography. So far, data mining and Geographic Information Systems (GIS) have existed as two separate technologies, each with its own methods, traditions and approaches to visualization and data analysis. Particularly, most contemporary GIS have only very basic spatial analysis functionality. The immense explosion in geographically referenced data occasioned by developments in IT, digital mapping, remote sensing, and the global diffusion of GIS emphasizes the importance of developing data driven inductive approaches to geographical analysis and modeling. Data mining, which is the partially automated search for hidden patterns in large databases, offers great potential benefits for applied GIS-based decision-making. Recently, the task of integrating these two technologies has become critical, especially as various public and private sector organizations possessing huge databases with thematic and geographically referenced data begin to realize the huge potential of the information hidden there. Among those organizations are: offices requiring analysis or dissemination of geo-referenced statistical data, public health services searching for explanations of disease clusters, environmental agencies assessing the impact of changing land-use patterns on climate change, geo-marketing companies doing customer segmentation based on spatial location.

1.2 How data mining works

Data mining is the process of extracting patterns from data. As more data are gathered, with the amount of data doubling every three years, data mining is becoming an increasingly important tool to transform these data into information. It is commonly used in a wide range of profiling practices, such as marketing, surveillance, fraud detection and scientific discovery[4].

While data mining can be used to uncover patterns in data samples, it is important to be aware that the use of non-representative samples of data may produce results that are not indicative of the domain. Similarly, data mining will not find patterns that may be present in the domain, if those patterns are not present in the sample being "mined". There is a tendency for insufficiently knowledgeable "consumers" of the results to attribute "magical abilities" to data mining, treating the technique as a sort of all-seeing crystal ball. Like any other tool, it only functions in conjunction with the appropriate raw material: in this case, indicative and representative data is what the user must first collect.

Further, the discovery of a particular pattern in a particular set of data does not necessarily mean that pattern is representative of the whole population from which that data was drawn. Hence, an important part of the process is the verification and validation of patterns on other samples of data.

The term data mining has also been used in a related but negative sense, to mean the deliberate searching for apparent but not necessarily representative patterns in large numbers of data. To avoid confusion with the other sense, the terms *data dredging* and *data snooping* are often used. Note, however, that dredging and snooping can be (and sometimes are) used as exploratory tools when developing and clarifying hypotheses.

[4] references paper no. [16]

1.2.1 ID3 Basic

ID3 is a simple decision tree learning algorithm developed by Ross Quinlan (1983)[5]. The basic idea of ID3 algorithm is to construct the decision tree by employing a top-down, greedy search through the given sets to test each attribute at every tree node. In order to select the attribute that is most useful for classifying given sets, we introduce a metric---information gain. To find an optimal way to classify a learning set, what we need to do is to minimize the questions asked (i.e. minimizing the depth of the tree). Thus, we need some function which can measure which questions provide the most balanced splitting. The information gain metric is such a function. In order to define information gain precisely, we need to discuss entropy first. First, lets assume, without loss of generality, that the resulting decision tree classifies instances into two categories, we'll call them P(positive)and N(negative).

Given a set S, containing these positive and negative targets, the entropy of S related to this Boolean classification is:

Entropy(S)= - P(positive)log2P(positive) - P(negative)log2P(negative)

P(positive): proportion of positive examples in S

P(negative): proportion of negative examples in S

For example, if S is (0.5+, 0.5-) then Entropy(S) is 1, if S is (0.67+, 0.33-) then Entropy(S) is 0.92, if P is (1+, 0 -) then Entropy(S) is 0. Note that the more uniform is the probability distribution, the greater is its information. You may notice that entropy is a measure of the impurity in a collection of training sets. But how it is related to the optimisation of our decision making in classifying the instances, What you will see at the following will answer this question.

[5] References paper no. [16]

1.2.2 Current concerns

Information Gain

As we mentioned before, to minimize the decision tree depth, when we traverse the tree path, we need to select the optimal attribute for splitting the
tree node, which we can easily imply that the attribute with the most entropy reduction is the best choice. We define information gain as "the expected reduction of entropy related to specified attribute when splitting a decision tree node".

The information gain, Gain(S, A) of an attribute A,

Gain(S, A) = Entropy(S) - Sum for v from 1 to n of (|Sv|/|S|) * Entropy (Sv)

We can use this notion of gain to rank attributes and to build decision trees where at each node is located the attribute with greatest gain among the attributes not yet considered in the path from the root.

The intention of this ordering is:

1. To create small decision trees so that records can be identified after only a few decision tree splitting.
2. To match a hoped for minimalism of the process of decision making

1.2.3 Problem Statement

Shannon Entropy finds its application in many fields. Here in Data Mining, Shannon Entropy has been used in ID3 algorithm to calculate Information Gain contained by data, which help in making Decision Tree.
However, It has been observed that the results obtained from Shannon Entropy, are rather complex, have more number of node and leaf and decision rules thus obtained are more in number which makes the decision making process time consuming.

Therefore, to minimize these problems, new algorithm has been proposed by modifying ID3 algorithm using Havrda and Charvat Entropy instead of Shannon Entropy.

1.3 Organisation of thesis

The rest of the thesis is organized as follows:

Chapter 2 We discuss different design approaches in related work used to implement ID3. We give description of existing architecture of our ID3 focused Entropy developed by Shannon. Finally, we provide an insight into the various techniques used in literature to evaluate focused ID3.

Chapter 3 We discuss different types of classification algorithm such as statistical based, Naive Bayes classifier, k-nearest neighbor algorithm, artificial neural algorithm and decision tree based algorithm. Estimation and Prediction can be viewed as types of classification. Further, we discuss the classification problem.

Chapter 4 We describe in detail Decision tree based algorithm, its representation and learning, also comparing ID3, C4.5 and CART algorithm.

Chapter 5 We talk about how use of Data mining took place. It's different issues in using ID3 algorithm, how it helps in taking decision. Monitoring the detailed work of ID3 algorithm.

Chapter 6 We discuss the concept of Havrda and Charvat Entropy. The role and non-additive nature of Havrda and Charvat. We also talk about how this entropy works in decision making in ID3 algorithm. Further, we show the calculation of information gain using Havrda and Charvat entropy, this information gain result helps in making Decision tree.

Chapter 7 We show the output of proposed algorithm, by implementing the code in C language. In the code we have taken the table as the input file named as input.nish., also we construct one classes.h as header file, which defines the number of classes, their definition and function to calculate the Information Gain to construct tree, which we import in main C source file. After compilation we get tree detailed output in C format.

Chapter 8 We conclude our achievement and indicate potential areas for future works.

Finally, we give the Bibliography, containing the details of the papers related to the subject.

CHAPTER 2

DESIGN APPROACHES AND EVALUATION METHODS

A **decision tree** (or **tree diagram**) is a decision support tool that uses a tree-like graph or model of decisions and their possible consequences, including chance event outcomes, resource costs, and utility. Decision trees are commonly used in operations research, specifically in decision analysis, to help identify a strategy most likely to reach a goal. Another use of decision trees is as a descriptive means for calculating conditional probabilities.

In decision analysis, a "decision tree" — and the closely-related influence diagram — is used as a visual and analytical decision support tool, where the expected values (or expected utility) of competing alternatives are calculated[6].
Decision trees have traditionally been created manually, as the following example shows:

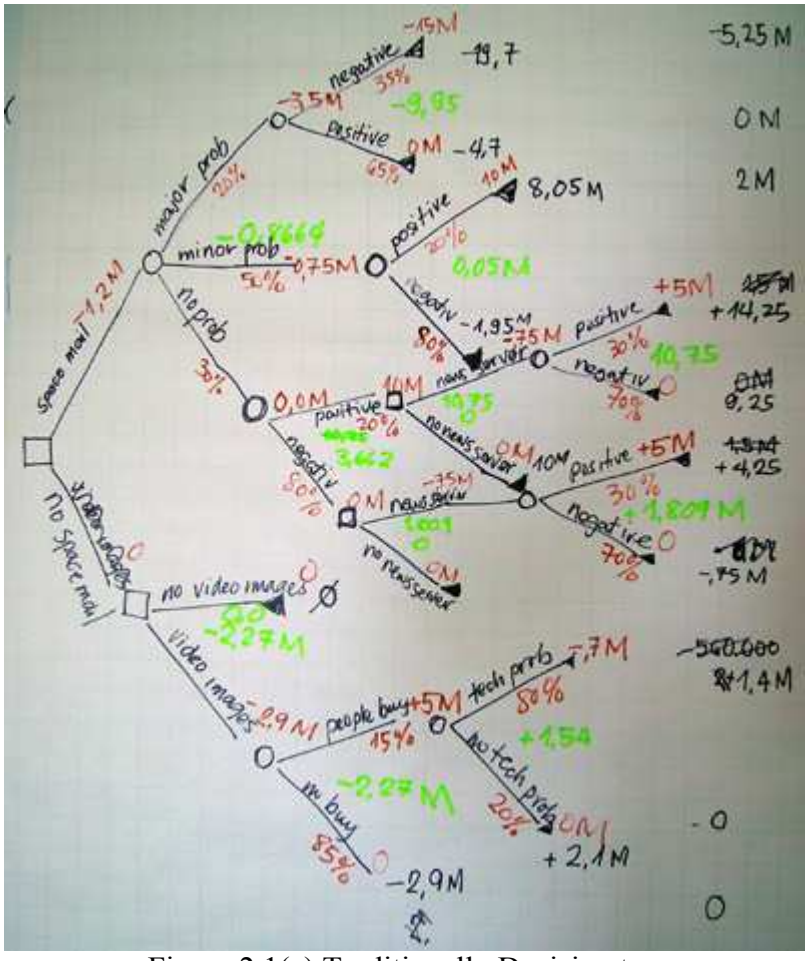


Figure 2.1(a) Traditionally Decision trees

[6] http://en.wikipedia.org/wiki/ decision-tree

The base strategy of ID3 algorithm of Data mining using Havrda and Charvat Entropy.

2.1 Our existing architecture

ID3 algorithms were introduced by Quinlan for inducing *Classification Models*, also called *Decision Trees*, from data. We are given a set of records. Each record has the same structure, consisting of a number of attribute/value pairs. One of these attributes represents the *category* of the record. The problem is to determine a decision tree that, on the basis of answers to questions about the non-category attributes, predicts correctly the value of the category attribute. Usually the category attribute takes only the values {true, false}, or {success, failure}, or something equivalent. In any case, one of its values will mean failure.

For example[7], we may have the results of measurements taken by experts on some widgets. For each widget we know what is the value for each measurement and what was decided, if to pass, scrap, or repair it. That is, we have a record with measurements as the non categorical attributes and disposition as the categorical attribute for the widget.

Here is a more detailed example. We are dealing with records reporting on weather conditions for playing golf. The categorical attribute specifies whether or not to play. The non-categorical attributes are:

ATTRIBUTE	POSSIBLE VALUES
outlook	sunny, overcast, rain
temperature	continuous
humidity	continuous
windy	true, false

2.1(a) Attribute table in existing architecture

and the training data is:

[7] Building Classification Models: ID3 and C4.5

OUTLOOK	TEMPERATURE	HUMIDITY	WINDY	PLAY
sunny	85	85	false	Don't Play
sunny	80	90	true	Don't Play
overcast	83	78	false	Play
rain	70	96	false	Play
rain	68	80	false	Play
rain	65	70	true	Don't Play
overcast	64	65	true	Play
sunny	72	95	false	Don't Play
sunny	69	70	false	Play
rain	75	80	false	Play
sunny	75	70	true	Play
overcast	72	90	true	Play
overcast	81	75	false	Play
rain	71	80	true	Don't Play

2.1(b) Training data table in existing architecture

Notice that in this example two of the attributes have continuous ranges, Temperature and Humidity. ID3 does not directly deal with such cases, though below we examine how it can be extended to do so. A decision tree is important not because it summarizes what we know, i.e. the *training set*, but because we hope it will **classify correctly** new cases. Thus when building classification models one should have both *training data* to build the model and *test data* to verify how well it actually works.

A simpler example from the stock market involving only discrete ranges has Profit as categorical attribute, with values {up, down}. Its non categorical attributes are:

ATTRIBUTE	POSSIBLE VALUES
age	old, midlife, new
competition	no, yes
type	software, hardware

and the training data is:

AGE	COMPETITION	TYPE	PROFIT
old	yes	swr	down
old	no	swr	down
old	no	hwr	down
mid	yes	swr	down

mid	yes	hwr	down
mid	no	hwr	up
mid	no	swr	up
new	yes	swr	up
new	no	hwr	up
new	no	swr	up

2.1.1 Baseline Framework

The basic ideas behind ID3 are that: In the decision tree each node corresponds to a non-categorical attribute and each arc to a possible value of that attribute. A leaf of the tree specifies the expected value of the categorical attribute for the records described by the path from the root to that leaf. [This defines what is a Decision Tree.] In the decision tree at each node should be associated the non-categorical attribute which is *most informative* among the attributes not yet considered in the path from the root. [This establishes what is a "Good" decision tree][8]

Entropy is used to measure how informative a node is. [This defines what we mean by "Good". This notion was introduced by Claude Shannon in Information Theory.]

[8] Building Classification Models: ID3 and C4.5

The base strategy of ID3 algorithm of Data mining using Havrda and Charvat Entropy.

2.2 Other approaches to focused Data mining

C4.5 is an extension of ID3 that accounts for unavailable values, continuous attribute value ranges, pruning of decision trees, rule derivation, and so on. In building a decision tree we can deal with training sets that have records with unknown attribute values by evaluating the gain, or the gain ratio, for an attribute by considering only the records where that attribute is defined.

In using a decision tree, we can classify records that have unknown attribute values by estimating the probability of the various possible results. In our golfing example, if we are given a new record for which the outlook is sunny and the humidity is unknown, we proceed as follows:

We move from the Outlook root node to the Humidity node following the arc labeled 'sunny'. At that point since we do not know the value of Humidity we observe that if the humidity is at most 75 there are two records where one plays, and if the humidity is over 75 there are three records where one does not play. Thus one can give as answer for the record the probabilities (0.4, 0.6) to play or not to play.

We can deal with the case of attributes with continuous ranges as follows. Say that attribute Ci has a continuous range. We examine the values for this attribute in the training set. Say they are, in increasing order, A1, A2, .., Am. Then for each value Aj, j=1,2,..m, we partition the records into those that have Ci values up to and including Aj, and those that have values greater than Aj. For each of these partitions we compute the gain, or gain ratio, and choose the partition that maximizes the gain. In our Golfing example, for humidity, if T is the training set, we determine the information for each partition and find the best partition at 75. Then the range for this attribute becomes {<=75, >75}. Notice that this method involves a substantial number of computations.

2.3 Evaluation strategies

If there are n equally probable possible messages, then the probability p of each is 1/n and the information conveyed by a message is -log(p) = log(n). [In what follows all logarithms are in base 2.] That is, if there are 16 messages, then log(16) = 4 and we need 4 bits to identify each message.

In general, if we are given a probability distribution P = (p1, p2, .., pn) then the Information conveyed by this distribution, also called the Entropy of P, is:

$$I(P) = -(p1*\log(p1) + p2*\log(p2) + .. + pn*\log(pn))$$

For example, if P is (0.5, 0.5) then I(P) is 1, if P is (0.67, 0.33) then I(P) is 0.92, if P is (1, 0) then I(P) is 0. [Note that the more uniform is the probability distribution, the greater is its information.]

If a set T of records is partitioned into disjoint exhaustive classes C1, C2, .., Ck on the basis of the value of the categorical attribute, then the information needed to identify the class of an element of T is **Info(T)** = I(P), where P is the probability distribution of the partition (C1, C2, .., Ck):

$$P = (|C1|/|T|, |C2|/|T|, ..., |Ck|/|T|)$$

In our golfing example, we have Info(T) = I(9/14, 5/14) = 0.94, and in our stock market example we have Info(T) = I(5/10,5/10) = 1.0.

If we first partition T on the basis of the value of a non-categorical attribute X into sets T1, T2, .., Tn then the information needed to identify the class of an element of T becomes the weighted average of the information needed to identify the class of an element of Ti, i.e. the weighted average of Info(Ti):

$$Info(X,T) = \text{Sum for i from 1 to n of } \frac{|Ti|}{|T|} * Info(Ti)$$

In the case of our golfing example, for the attribute Outlook we have

Info(Outlook,T) = 5/14*I(2/5,3/5) + 4/14*I(4/4,0) + 5/14*I(3/5,2/5)

= 0.694

Consider the quantity Gain(X,T) defined as

Gain(X,T) = Info(T) - Info(X,T)

This represents the difference between the information needed to identify an element of T and the information needed to identify an element of T after the value of attribute X has been obtained, that is, this is the gain in information due to attribute X.

In our golfing example, for the Outlook attribute the gain is:

Gain(Outlook,T) = Info(T) - Info(Outlook,T) = 0.94 - 0.694 = 0.246[9].

If we instead consider the attribute Windy, we find that Info(Windy,T) is 0.892 and Gain(Windy,T) is 0.048. Thus Outlook offers a greater informational gain than Windy.

We can use this notion of **gain** to rank attributes and to build decision trees where at each node is located the attribute with greatest gain among the attributes not yet considered in the path from the root.

The intent of this ordering is twofold:

To create small decision trees so that records can be identified after only a few questions.

To match a hoped for minimality of the process represented by the records being considered (Occam's Razor).

[9] Building Classification Models: ID3 and C4.5, *en.wikipedia.org/wiki/**Data_mining***

CHAPTER 3

CLASSIFICATION ALGORITHM

3.1 Introduction

Classification is perhaps the most familiar and the most popular data mining technique. Examples of classification application include images and pattern recognition, medical diagnosis, loan approval, detecting faults in industrial application, and classifying market trends[10]. Estimation and prediction can be viewed as types of classification. Prediction can be thought of as classifying an attribute value into one of set possible values. It is often viewed as forecasting value, while classification forecasts a discrete value.

All approaches to performing classification assume some knowledge of the data. Often a training set is used to develop the specific parameters required by the technique. Training data consist of sample input data as well as the classification assignment for the data.
The classification problem is stated as
Definition: Given a database D = {t1, t2, t3……tm} of tuples (items, records)
And a set of classes c = {c1, c2, c3…..cm} the classification problem is to define a mapping f: D-> C where ti is assigned to one class. A class, cj contain precisely those mapped to it; that is, cj = {ti/f(ti) = cj, 1<=i<=n, and t1 belong to D}

Our definition views classification as a mapping from the database to the set of classes. Each tuple in the database is assigned to exactly one class.
The classes that exist for a classification problem are indeed equivalence classes. In actuality, the problem usually is implemented in two phases:

1. Create a specific model by evaluating the training data. This step has as input the training data and as output a definition of the model developed. The model created classifies the training data as accurately as possible.
2. Apply the model developed in step 1 by classifying tuples from the target database.

[10] Jaideep Srivastava @http://www.cs.umn.edu/faculty/srivasta.html
Mike Kassoff @http://logic.stanford.edu/classes/cs246/lectures2001/mkassoff_lecture.ppt

3.2 Different types of classification algorithm

Statistical based algorithm

Statistics is the science of making effective use of numerical data relating to groups of individuals or experiments. It deals with all aspects of this, including not only the collection, analysis and interpretation of such data, but also the planning of the collection of data, in terms of the design of surveys and experiments.

A statistician is someone who is particularly versed in the ways of thinking necessary for the successful application of statistical analysis. Often such people have gained this experience after starting work in any of a number of fields. There is also a discipline called *mathematical statistics*, which is concerned with the theoretical basis of the subject.

The word *statistics* can either be singular or plural. When it refers to the discipline, "statistics" is singular, as in "Statistics is an art." When it refers to quantities (such as mean and median) calculated from a set of data, *statistics* is plural, as in "These statistics are misleading."

Naive Bayes classifier

A Bayes classifier is a simple probabilistic classifier based on applying Bayes' theorem[11] (from Bayesian statistics) with strong (naive) independent assumptions. A more descriptive term for the underlying probability model would be "independent feature model".

In simple terms, a naive Bayes classifier assumes that the presence (or absence) of a particular feature of a class is unrelated to the presence (or absence) of any other feature. For example, a fruit may be considered to be an apple if it is red, round, and about 4" in diameter. Even if these features depend on each other or upon the existence of the other features, a naive Bayes classifier considers all of these properties to independently contribute to the probability that this fruit is an apple.

[11] Baron, Jonathan (1994). *Thinking and Deciding* (2 ed.). Oxford University Press. pp. 209–210.

Depending on the precise nature of the probability model, naive Bayes classifiers can be trained very efficiently in a supervised learning setting. In many practical applications, parameter estimation for naive Bayes models uses the method of maximum likelihood; in other words, one can work with the naive Bayes model without believing in Bayesian probability or using any Bayesian methods.

An advantage of the naive Bayes classifier is that it requires a small amount of training data to estimate the parameters (means and variances of the variables) necessary for classification. Because independent variables are assumed, only the variances of the variables for each class need to be determined and not the entire covariance matrix.

k-nearest neighbor algorithm

The *k*-nearest neighbors algorithm (*k*-NN)[12] is a method for classifying objects based on closest training examples in the feature space. *k*-NN is a type of instance-based learning, or lazy learning where the function is only approximated locally and all computation is deferred until classification. The *k*-nearest neighbor algorithm is amongst the simplest of all machine learning algorithms: an object is classified by a majority vote of its neighbors, with the object being assigned to the class most common amongst its *k* nearest neighbors (*k* is a positive integer, typically small). If *k* = 1, then the object is simply assigned to the class of its nearest neighbor.

The same method can be used for regression, by simply assigning the property value for the object to be the average of the values of its *k* nearest neighbors. It can be useful to weight the contributions of the neighbors, so that the nearer neighbors contribute more to the average than the more distant ones. (A common weighting scheme is to give each neighbor a weight of 1/*d*, where *d* is the distance to the neighbor. This scheme is a generalization of linear interpolation.The neighbors are taken from a set of objects for which the correct classification (or, in the case of regression, the value of the property) is known. This can be thought of as the training set for the algorithm, though no explicit training step is required. The

[12] Cover TM, Hart PE (1967). "Nearest neighbor pattern classification". *IEEE Transactions on Information Theory* **13** (1): 21-27.

k-nearest neighbor algorithm is sensitive to the local structure of the data. Nearest neighbor rules in effect compute the decision boundary in an implicit manner. It is also possible to compute the decision boundary itself explicitly, and to do so in an efficient manner so that the computational complexity is a function of the boundary complexity.

Artificial neural network

An artificial neural network (ANN), usually called "neural network" (NN), is a mathematical model or computational model that tries to simulate the structure and/or functional aspects of biological neural networks. It consists of an interconnected group of artificial neurons and processes information using a connectionist approach to computation. In most cases an ANN is an adaptive system that changes its structure based on external or internal information that flows through the network during the learning phase. Modern neural networks are non-linear statistical data modeling tools. They are usually used to model complex relationships between inputs and outputs or to find patterns in data. The utility of ANN models lies in the fact that they can be used to infer a function from observations. This is particularly useful in applications where the complexity of the data or task makes the design of such a function by hand impractical.

Summary

When looking at the approaches based on the analysis of complexities we see that they all are very efficient. This is due to the fact that once the model is created; applying it for classification is relatively straight forward. The statistical technique , regression and naïve bayes, requires constant time to classify a tuple once the models are built .The distance based approaches, simple and KNN , is also constant but require that each tuple be compared either to a representative for each

class or to all items in the training set. Assuming there are q of these, the KNN then requires O(q) time per tuple. DT classification techniques, ID3, C4.5 and CART require a number of comparisons that are (in the worst case) equal to the longest path from a root to a leaf node. Thus, they require O (log q) time per tuple. Since q is a constant, we can view these as being performed in a constant time as well. The NN approaches again require that tuple be propagated through the graphs. Since the size of the graph is constant, this can be viewed as being performed in constant time. Thus, all algorithms are O(n) to classify the n items in the database.

CHAPTER 4

DECISION TREE BASED ALGORITHM

4.1 Representation and learning

The Decision Tree algorithm creates hierarchical structure of classification rules“ If ... Then ...” looking like a tree. To decide which type to assign for an object or situation, we need to answer the questions, standing in the branches of the tree, starting from the root. The questions look like this: “Is the value of the parameter A greater than X?”. If the answer is positive, a pass to the right performs, if it is negative – to the left; then a question related to the new branch follows. 'Decision tree learning is a method for approximating discrete-valued target functions, in which the learned function is represented by a decision tree.Decision tree learning is one of the most widely used and practical methods for inductive inference'. **(Tom M. Mitchell, 1997, p52)**

A decision Tree consists of 3 types of nodes[13]:-

1. Decision nodes - commonly represented by squares
2. Change nodes - represented by circles
3. End nodes - represented by triangles

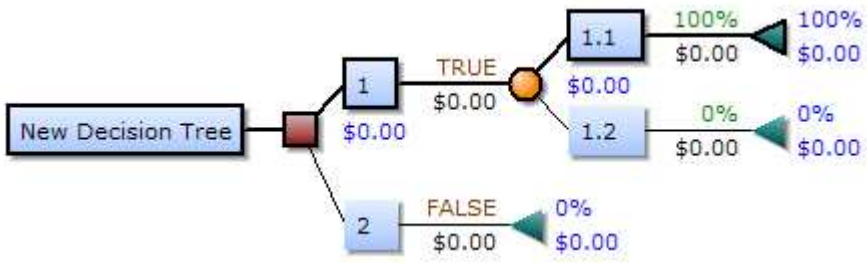


Figure 4.1(a) Decision Tree showing types of nodes

Drawn from left to right, a decision tree has only burst nodes (splitting paths) but no sink nodes (converging paths). Therefore, used manually, they can grow very big and are then often hard to draw fully by hand.

[13] Decision-Tree-Elements.png
http://en.wikipedia.org/wiki/ decision-tree

4.2 ID3 algorithm

ID3 (Iterative Dichotomiser 3) is an algorithm used to generate a decision tree invented by Ross Quinlan[14]. ID3 is the precursor to the C4.5 algorithm.

The ID3 algorithm can be summarized as follows:

1. Take all unused attributes and count their entropy concerning test samples
2. Choose attribute for which entropy is minimum (or, equivalently, information gain is maximum)
3. Make node containing that attribute

The algorithm is as follows:

ID3 (Examples, Target_Attribute, Attributes)

- Create a root node for the tree
- If all examples are positive, Return the single-node tree Root, with label = +.
- If all examples are negative, Return the single-node tree Root, with label = -.
- If number of predicting attributes is empty, then Return the single node tree Root, with label = most common value of the target attribute in the examples.
- Otherwise Begin
    - A = The Attribute that best classifies examples.
    - Decision Tree attribute for Root = A.
    - For each possible value, v_i, of A,
        - Add a new tree branch below Root, corresponding to the test A = v_i Let Examples(v_i), be the subset of examples that have the value v_i for A
        - If Examples(v_i) is empty
            - Then below this new branch add a leaf node with label = most common target value in the examples

[14] www.comp.dit.ie/btierney/oracle11gdoc/.../**algo_decisiontree**.htm
Reference paper no.[22]

        - Else below this new branch add the subtree ID3 (Examples(v_i), Target_Attribute, Attributes – {A})
- End
- Return Root

4.3 C4.5 algorithm

C4.5 is an algorithm used to generate a decision tree developed by Ross Quinlan[15]. C4.5 is an extension of Quinlan's earlier ID3 algorithm. The decision trees generated by C4.5 can be used for classification, and for this reason, C4.5 is often referred to as a statistical classifier

C4.5 builds decision trees from a set of training data in the same way as ID3, using the concept of information entropy. The training data is a set $S = s_1,s_2,...$ of already classified samples. Each sample $s_i = x_1,x_2,...$ is a vector where $x_1,x_2,...$ represent attributes or features of the sample. The training data is augmented with a vector $C = c_1,c_2,...$ where $c_1,c_2,...$ represent the class to which each sample belongs.

At each node of the tree, C4.5 chooses one attribute of the data that most effectively splits its set of samples into subsets enriched in one class or the other. Its criterion is the normalized information gain (difference in entropy) that results from choosing an attribute for splitting the data. The attribute with the highest normalized information gain is chosen to make the decision. The C4.5 algorithm then recurs on the smaller sub lists.

This algorithm has a few base cases. All the samples in the list belong to the same class. When this happens, it simply creates a leaf node for the decision tree saying to choose that class. None of the features provide any information gain. In this case, C4.5 creates a decision node higher up the tree using the expected value of the class.

[15] Reference paper no. [7]

Instance of previously-unseen class is encountered. Again, C4.5 creates a decision node higher up the tree using the expected value.

In pseudo code the algorithm is:

Check for base cases

For each attribute *a*

Find the normalized information gain from splitting on *a*

Let *a_best* be the attribute with the highest normalized information gain

Create a decision *node* that splits on *a_best*

Recur on the sublists obtained by splitting on *a_best*, and add those nodes as children of *node*

J48 is an open source Java implementation of the C4.5 algorithm in the weka data mining tool.

4.4 CART algorithm

In most general terms, the purpose of the analyses via tree-building algorithms is to determine a set of if-then logical (split) conditions that permit accurate prediction or classification of cases[16].

Classification Trees

The purpose of the analysis is to learn how we can discriminate between the three types of flowers, based on the four measures of width and length of petals and sepals. Discriminate function analysis will estimate several linear combinations of predictor variables for computing classification scores (or probabilities) that allow the user to determine the predicted classification for each observation. A classification tree will determine a set of logical if-then conditions (instead of linear equations) for predicting or classifying cases instead.

[16] Reference paper no.[12],&[17]

Regression Trees

The general approach to derive predictions from few simple if-then conditions can be applied to regression problems as well. This example is based on the data file *Poverty*, A reanalysis of those data, using the regression tree analysis [and v-fold cross-validation, yields the following results:

A quick review of the scatter plot of observed vs. predicted values shows how the discrimination between the latter two groups is particularly well "explained" by the tree model.

Advantages of Classification and Regression Trees (C&RT) Methods

As mentioned earlier, there are a large number of methods that an analyst can choose from when analyzing classification or regression problems. Tree classification techniques, when they "work" and produce accurate predictions or predicted classifications based on few logical if-then conditions, have a number of advantages over many of those alternative techniques.

Simplicity of results: In most cases, the interpretation of results summarized in a tree is very simple. This simplicity is useful not only for purposes of rapid classification of new observations (it is much easier to evaluate just one or two logical conditions, than to compute classification scores for each possible group, or predicted values, based on all predictors and using possibly some complex nonlinear model equations), but can also often yield a much simpler "model" for explaining why observations are classified or predicted in a particular manner (e.g., when analyzing business problems, It is much easier to present a few simple if-then statements to management, than some elaborate equations).

Tree methods are nonparametric and nonlinear: The final results of using tree methods for classification or regression can be summarized in a series of (usually few) logical if-then conditions (tree nodes). Therefore, there is no implicit assumption that the underlying relationships between the predictor variables and the dependent variable are linear, follow some specific non-linear link function [e.g., see Generalized Linear/Nonlinear Models (GLZ)], or that they are even monotonic in nature. For example, some continuous outcome variable of interest could be positively related to a variable Income if the income is less than some certain amount, but negatively related if it is more than that amount (i.e., the tree could reveal multiple splits based on the same variable Income, revealing such a non-monotonic relationship between the variables).

Thus, tree methods are particularly well suited for data mining tasks, where there is often a little prior knowledge, nor any coherent set of theories or predictions regarding which variables are related and how. In those types of data analyses, tree methods can often reveal simple relationships between just a few variables that could have easily gone unnoticed using other analytic techniques

CHAPTER 5

USER DRIVEN DATA MINING

5.1 Issues in Id3 algorithm

The ID3 algorithm is used to build a decision tree, given a set of non-categorical attributes C1, C2, .., Cn, the categorical attribute C, and a training set T of records.

```
function ID3 (R: a set of non-categorical attributes,
              C: the categorical attribute,
              S: a training set) returns a decision tree;
begin
    If S is empty, return a single node with value Failure;
    If S consists of records all with the same value for
      the categorical attribute,
      return a single node with that value;
    If R is empty, then return a single node with as value
      the most frequent of the values of the categorical attribute
      that are found in records of S; [note that then there
      will be errors, that is, records that will be improperly
      classified];
    Let D be the attribute with largest Gain(D,S)
      among attributes in R;
    Let {dj| j=1,2, .., m} be the values of attribute D;
    Let {Sj| j=1,2, .., m} be the subsets of S consisting
      respectively of records with value dj for attribute D;
    Return a tree with root labeled D and arcs labeled
      d1, d2, .., dm going respectively to the trees

        ID3(R-{D}, C, S1), ID3(R-{D}, C, S2), .., ID3(R-{D}, C, Sm);
end ID3;
```

For example[17], we may have the results of measurements taken by experts on some widgets. For each widget we know what the value for each measurement is and what

[17]Building Classification Models: ID3 and C4.5

Was decided, if to pass, scrap, or repair it. That is, we have a record with measurements as non categorical attributes and as disposition categorical attribute for the widget.

Here is a more detailed example. We are dealing with records reporting on weather conditions for playing golf. The categorical attribute specifies whether or not to play. The non-categorical attributes are:

ATTRIBUTE	POSSIBLE VALUES
outlook	sunny, overcast, rain
temperature	continuous
humidity	continuous
windy	true, false

5.1 (a) Attribute table in Issues in Id3 algorithm

and the training data is:

OUTLOOK	TEMPERATURE	HUMIDITY	WINDY	PLAY
sunny	85	85	false	Don't Play
sunny	80	90	true	Don't Play
overcast	83	78	false	Play
rain	70	96	false	Play
rain	68	80	false	Play
rain	65	70	true	Don't Play
overcast	64	65	true	Play
sunny	72	95	false	Don't Play
sunny	69	70	false	Play
rain	75	80	false	Play
sunny	75	70	true	Play
overcast	72	90	true	Play
overcast	81	75	false	Play
rain	71	80	true	Don't Play

5.1(b) Training data table Issues in Id3 algorithm

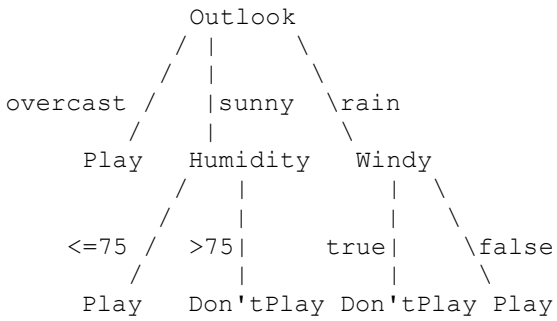


Figure 5.1(a) Decision tree for weather conditions

In the stock market case the decision tree is:

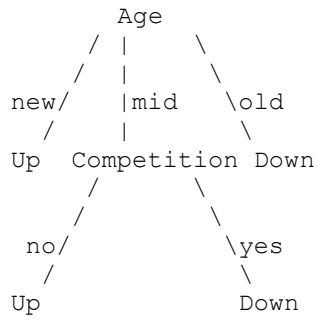


Figure 5.1(b) Decision tree for stock market conditions

Using Gain Ratios

The notion of Gain introduced earlier tends to favour attributes that have a large number of values. For example, if we have an attribute D that has a distinct value for each record, then Info(D,T) is 0, thus Gain(D,T) is maximal. To compensate for this Quinlan suggests using the following ratio instead of Gain:

$$GainRatio(D,T) = \frac{Gain(D,T)}{SplitInfo(D,T)}$$

Where SplitInfo(D,T) is the information due to the split of T on the basis of the value of the categorical attribute D. Thus SplitInfo(D,T) is

$$I(|T1|/|T|, |T2|/|T|, .., |Tm|/|T|)$$

where {T1, T2, .. Tm} is the partition of T induced by the value of D.

In the case of our golfing example SplitInfo(Outlook,T) is

-5/14*log(5/14) - 4/14*log(4/14) - 5/14*log(5/14) = 1.577 thus the GainRatio of Outlook is 0.246/1.577 = 0.156. And SplitInfo(Windy,T) is

-6/14*log(6/14) - 8/14*log(8/14) = 6/14*0.1.222 + 8/14*0.807

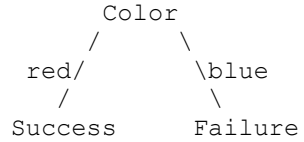
= 0.985

thus the Gain Ratio of Windy is 0.048/0.985 = 0.049

5.2 Pruning Decision Trees and Deriving Rule Sets

The decision tree built using the training set, because of the way it was built, deals correctly with most of the records in the training set. In fact, in order to do so, it may become quite complex, with long and very uneven paths.

Pruning [18]of the decision tree is done by replacing a whole subtree by a leaf node. The replacement takes place if a decision rule establishes that the expected error rate in the subtree is greater than in the single leaf. For example, if the simple decision tree



is obtained with one training red success record and two training blue Failures, and then in the Test set we find three red failures and one blue success, we might consider replacing this subtree by a single Failure node. After replacement we will have only two errors instead of five failures.

[18] Reference paper no. [1],
Advanced data mining and applications: third international conference, ADMA By Reda Alhajj

Winston shows how to use *Fisher's exact test* to determine if the category attribute is truly dependent on a non-categorical attribute. If it is not, then the non-categorical attribute need not appear in the current path of the decision tree.

Quinlan and Breiman suggest more sophisticated pruning heuristics.

It is easy to *derive a rule set from a decision tree*: write a rule for each path in the decision tree from the root to a leaf. In that rule the left-hand side is easily built from the label of the nodes and the labels of the arcs.

The resulting rules set can be simplified:

Let LHS be the left hand side of a rule. Let LHS' be obtained from LHS by eliminating some of its conditions. We can certainly replace LHS by LHS' in this rule if the subsets of the training set that satisfy respectively LHS and LHS' are equal.

A rule may be eliminated by using meta conditions such as *"if no other rule applies"*.

CHAPTER 6

THE HAVRDA AND CHARVAT ENTROPY

In information theory, entropy is a measure of the uncertainty associated with a random variable. The term by itself in this context usually refers to the Shannon entropy, which quantifies, in the sense of an expected value, the information contained in a message, usually in units such as bits. Equivalently, the Shannon entropy is a measure of the average information content one is missing when one does not know the value of the random variable. The concept was introduced by Claude E. Shannon in his 1948 paper "A Mathematical Theory of Communication".

Entropy as information content[19]

Entropy is defined in the context of a probabilistic model. Independent fair coin flips have Entropy of 1 bit per flip. A source that always generates a long string of B's has Entropy of 0, since the next character will always be a 'B'. The entropy rate of a data source means the average number of bits per symbol needed to encode it.

6.1 Definition and Role of Havrda and Charvat Entropy

Let P = (p1, p2 …pn) be a probability distribution, p denotes the probability mass function of X and α is its inherent parameter[20]. Then Havrda and Charvat[21] gave the entropy measure by formula shown below,

$$h(p) = \frac{1}{1-a}\left(\sum_{i=1}^{n} X_i^a - 1\right)$$

This formula calculates Entropy. To avoid deduced solution in decision tree making process, Havrda and Charvat entropy based ID3 algorithm is proposed which gives good solution in reasonable time. Such algorithm can give short and fast decision for supply of good in company

[19] http://en.wikipedia.org/wiki/Entropy_%28information_theory%29
[20] Reference paper no. [15]
[21] J. Havrda, F. Charvat, Kybernetika 30 (1967) 30.

6.2 Using Havrda and Charvat Entropy in ID3 algorithm.

The measure of tree component is one of the most important problems of ID3. Such problems occur when we have to take decision for who will be the root of the tree. So to find one we have to calculate first the **needed information, (Tom M. Mitchell, 1997, p57).**

So for needed information we divided our data[22] into category for which we are making tree. Here we divided data according to customer type, taking sample data into two parts: Sing_Customer (B) and Normal_Customer (N).

6.2.1 Analysis of new algorithm

S no.	Freight fee	Payment	Weight	Dispatch times	Customer Type
1	100-1000	<100	100-500	<5	B
2	>1000	<100	>500	>20	B
3	>1000	<100	100-500	5-20	B
4	>1000	<100	>500	5-20	B
5	100-1000	<100	>500	<5	N
6	100-1000	100-2000	100-500	>20	B
7	100-1000	>2000	<100	5-20	N
8	100-1000	<100	<100	5-20	N
9	100-1000	100-2000	<100	>20	N
10	100-1000	<100	100-500	>20	B
11	<100	>2000	100-500	5-20	N
12	<100	<100	<100	5-20	N
13	<100	<100	<100	5-20	B
14	<100	<100	<100	<5	N
15	<100	>2000	<100	5-20	B
16	<100	<100	<100	<5	N
17	<100	100-2000	<100	<5	N
18	<100	100-2000	<100	<5	N
19	<100	<100	<100	<5	N

6.2(a) the Information of Customer Dispatch Goods

[22] Reference paper no.[2]

This summarized data of customer dispatch information in a section period (one month) from an information system database of a 3PL, which included 19 items in this sample data set. In this example, all sample data is divided by Customer Type (CT) into two classes, which are Sign_Customer (B) and Normal_Customer (N) respectively, and has four properties: Freight Fee, Payment, Weight, and Dispatch Time. On the one hand, the summarizing data is integrated data from different sections and different consignment nodes. On the other hand, it is the process of generalizing the sample data, namely, the low level data are substituted by high level convenient to data mining. The values of these four properties are: Freight Fee (<100,100~1000, >1000); Payment (<100, 100~2000, <2000); Weight (<100kg, 100kg~500kg, >500kg); Dispatch Times (<5, 5~20, >20)[23]. The meanings of these properties are: The freight fee is paid by customer for the transport cost; the payment is Transportation Company bring the money of the goods from the receiver to dispatcher; the weight is measured by kilogram; the dispatch time is the sum of times during the summarized period.

We can calculate needed information by taking probality of customer type. Here the B class has 8 items and N has 11 items. Therefore, needed information gain of taken sample by putting α=0.25 in Havrda and Charvat Entropy formula is shown below.

Assuming α=0.25

The needed information gain will be

$$I(8,11) = \frac{\left[\left(\frac{8}{19}\right)^{.25} + \left(\frac{11}{19}\right)^{0.25} - 1\right]}{1-0.25} = \frac{0.677}{0.75} = 0.903$$

Then we divide the sample data by four properties: freight fee, payment, weight, and dispatch time respectively. Therefore the corresponding anticipated information of sample data are:

[23] Reference paper no.[2]

$$E(\text{Freight fee}) = \frac{\frac{9}{19}\left[\left(\frac{2}{9}\right)^{.25}+\left(\frac{7}{9}\right)^{0.25}-1\right]}{0.75} + \frac{\frac{7}{19}\left[\left(\frac{4}{7}\right)^{.25}+\left(\frac{3}{7}\right)^{0.25}-1\right]}{0.75} + \frac{\frac{3}{19}\left[\left(\frac{3}{3}\right)^{.25}-1\right]}{0.75}$$

$$=0.395 + 0.333 + 0 = 0.728$$

$$E(\text{Payment}) = \frac{\frac{12}{19}\left[\left(\frac{6}{12}\right)^{.25}+\left(\frac{6}{12}\right)^{0.25}-1\right]}{0.75} + \frac{\frac{4}{19}\left[\left(\frac{1}{4}\right)^{.25}+\left(\frac{3}{4}\right)^{0.25}-1\right]}{0.75} + \frac{\frac{3}{19}\left[\left(\frac{2}{3}\right)^{.25}+\left(\frac{1}{3}\right)^{0.25}-1\right]}{0.75}$$

$$= 0.574 + 0.179 + 0{,}139 = 0.893$$

$$E(\text{Weight}) = \frac{\frac{11}{19}\left[\left(\frac{2}{11}\right)^{.25}+\left(\frac{9}{11}\right)^{0.25}-1\right]}{0.75} + \frac{\frac{5}{19}\left[\left(\frac{4}{5}\right)^{.25}+\left(\frac{1}{5}\right)^{0.25}-1\right]}{0.75} + \frac{\frac{3}{19}\left[\left(\frac{2}{3}\right)^{.25}+\left(\frac{1}{3}\right)^{0.25}-1\right]}{0.75}$$

$$= 0.466 + 0.216 + 0.139 = 0.822$$

$$E(\text{Time}) = \frac{\frac{7}{19}\left[\left(\frac{1}{7}\right)^{.25}+\left(\frac{6}{7}\right)^{0.25}-1\right]}{0.75} + \frac{\frac{8}{19}\left[\left(\frac{4}{8}\right)^{.25}+\left(\frac{4}{8}\right)^{0.25}-1\right]}{0.75} + \frac{\frac{4}{19}\left[\left(\frac{3}{4}\right)^{.25}+\left(\frac{1}{4}\right)^{0.25}-1\right]}{0.75}$$

$$= 0.283 + 0.383 + 0.179 = 0.845$$

6.2.2 corresponding information gain are :

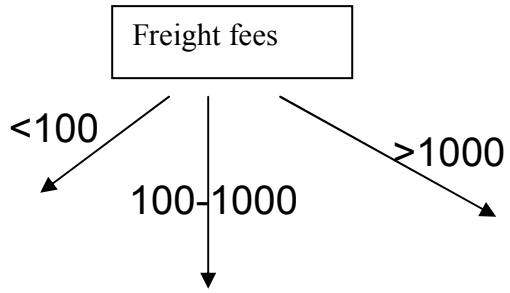
Gain (Freight fee) = I (S1, S2) – E (Freight fee)
= 0.903 – 0.728 = 0.175

Gain (Payment) = I (S1, S2) – E (Payment)
= 0.903 – 0.893 = 0.011

Gain (Weight) = I (S1, S2) – E (Weight)
= 0.903 – 0.822 = 0.082

Gain (Time) = I (S1, S2) – E (Time)
= 0.903 – 0.845 = 0.058

The information gain for freight fee is largest. Therefore freight fee will be root of decision tree.



Now considering table where freight fee is <100, from main table.

S no.	Freight fee	Payment	Weight	Dispatch times	Customer Type
11	<100	>2000	100-500	5-20	N
12	<100	<100	<100	5-20	N
13	<100	<100	<100	5-20	B
14	<100	<100	<100	<5	N
15	<100	>2000	<100	5-20	B
16	<100	<100	<100	<5	N
17	<100	100-2000	<100	<5	N
18	<100	100-2000	<100	<5	N
19	<100	<100	<100	<5	N

6.2(b) The Information of Customer Dispatch Goods for freight fee (<100)

So, here B=2, N= 7 we calculated needed information

$$I(2, 7) = \frac{\left[\left(\frac{2}{9}\right)^{.25} + \left(\frac{7}{9}\right)^{0.25} - 1\right]}{0.75} = 0.625/.75 = 0.834$$

Corresponding anticipated information of different properties are:

E (Freight fee) = 0.834

$$\text{E (Payment)} = \frac{\frac{5}{9}\left[\left(\frac{1}{5}\right)^{.25} + \left(\frac{4}{5}\right)^{0.25} - 1\right]}{0.75} + \frac{\frac{2}{9}\left[0 + \left(\frac{2}{2}\right)^{0.25} - 1\right]}{0.75} + \frac{\frac{2}{9}\left[\left(\frac{1}{2}\right)^{.25} + \left(\frac{1}{2}\right)^{0.25} - 1\right]}{0.75}$$

$$= 0.455 + 0 + 0.202 = 0.657$$

$$\text{E (Weight)} = \frac{\frac{8}{9}\left[\left(\frac{2}{8}\right)^{.25} + \left(\frac{6}{8}\right)^{0.25} - 1\right]}{0.75} + 0 + 0 = 0.756$$

$$\text{E (Time)} = \frac{\frac{5}{9}\left[0 + \left(\frac{5}{5}\right)^{0.25} - 1\right]}{0.75} + \frac{\frac{4}{19}\left[\left(\frac{2}{4}\right)^{.25} + \left(\frac{2}{4}\right)^{0.25} - 1\right]}{0.75} + 0$$

$$= 0 + 0.404 + 0 = 0.404$$

Now the corresponding Information Gain for above properties are:

Gain (Freight fee) = I (S1, S2) – E (Freight fee)
= 0.834 – 0.834 = 0

Gain (Payment) = I (S1, S2) – E (Payment)
= 0.834 – 0.657 = 0.177

Gain (Weight) = I (S1, S2) – E (Weight)
= 0.834 – 0.756 = 0.078

Gain (Time) = I (S1, S2) – E (Time)
= 0.834 – 0.404 = 0.430

Here Information gain of Time is maximum, therefore Time will be SUBROOT under Freight Fee (<100) root.

Now consider table for freight fee (100-1000) from main table.

S no.	Freight fee	Payment	Weight	Dispatch times	Customer Type
1	100-1000	<100	100-500	<5	B
5	100-1000	<100	>500	<5	N
6	100-1000	100-2000	100-500	>20	B
7	100-1000	>2000	<100	5-20	N
8	100-1000	<100	<100	5-20	N
9	100-1000	100-2000	<100	>20	N
10	100-1000	<100	100-500	>20	B

6.2(c) The Information of Customer Dispatch Goods for freight fee (100-1000)

Here B=3 and N=4

Now the Needed Information Gain here will be,

$$I(3, 4) = \frac{\left[\left(\frac{3}{7}\right)^{.25} + \left(\frac{4}{7}\right)^{0.25} - 1\right]}{0.75} = 0.905$$

Also, corresponding anticipated information for different properties are:

E (Freight Fee) = 0.905

$$\text{E (Payment)} = \frac{\frac{4}{7}\left[\left(\frac{2}{4}\right)^{.25}+\left(\frac{2}{4}\right)^{0.25}-1\right]}{0.75} + \frac{\frac{2}{7}\left[\left(\frac{1}{2}\right)^{.25}+\left(\frac{1}{2}\right)^{0.25}-1\right]}{0.75} + \frac{\frac{1}{7}\left[0+\left(\frac{1}{1}\right)^{0.25}-1\right]}{0.75}$$

$$= 0.519 + 0.259 + 0 = 0.778$$

$$\text{E (Weight)} = \frac{\frac{3}{7}\left[0+\left(\frac{3}{3}\right)^{0.25}-1\right]}{0.75} + \frac{\frac{3}{7}\left[0+\left(\frac{3}{3}\right)^{0.25}-1\right]}{0.75} + \frac{\frac{1}{7}\left[\left(\frac{1}{1}\right)^{0.25}-1\right]}{0.75}$$

$$= 0 + 0 + 0 = 0$$

$$\text{E (Time)} = \frac{\frac{2}{7}\left[\left(\frac{1}{2}\right)^{.25}+\left(\frac{1}{2}\right)^{0.25}-1\right]}{0.75} + \frac{\frac{3}{7}\left[\left(\frac{3}{3}\right)^{.25}-1\right]}{0.75} + \frac{\frac{2}{7}\left[\left(\frac{2}{2}\right)^{.25}-1\right]}{0.75}$$

$$= 0.638 + 0 + 0 = 0.638$$

The corresponding information gains are:

Gain (Freight fee) = I (S1, S2) – E (Freight fee)

= 0

Gain (Payment) = I (S1, S2) – E (Payment)

= 0.905 – 0.778 = 0.127

Gain (Weight) = I (S1, S2) – E (Weight)

= 0.905 – 0 = 0.905

Gain (Time) = I (S1, S2) – E (Time)

= 0.905 – 0.638 = 0.265

Information Gain for Weight is largest, so it will be sub root, under Freight fee (100-1000) category.

Now Consider table for Freight fee (>1000)

S no.	Freight fee	Payment	Weight	Dispatch times	Customer Type
2	>1000	<100	>500	>20	B
3	>1000	<100	100-500	5-20	B
4	>1000	<100	>500	5-20	B

6.2(d) The Information of Customer Dispatch Goods for freight fee (>1000)

From above table we can conclude that under freight fee **(>1000)** sub root will be PAYMENT, under this for payment <100, customers will be of B type only.

Similarly, we will conclude for sub root WEIGHT under freight fee **(100-1000**). From the table we observe that for sub root weight <100 customer is of N type, for 100-500 customer is of B type and for weight >500 customer is of N type.
In the same way, for freight fee **(<100)** sub root will be TIME. Under this for time <5 customer is of N type and for time 5-20 customer is of B/N type.

From above calculation and observation we have drawn the following tree.

6.2.3 Generation of Decision Tree

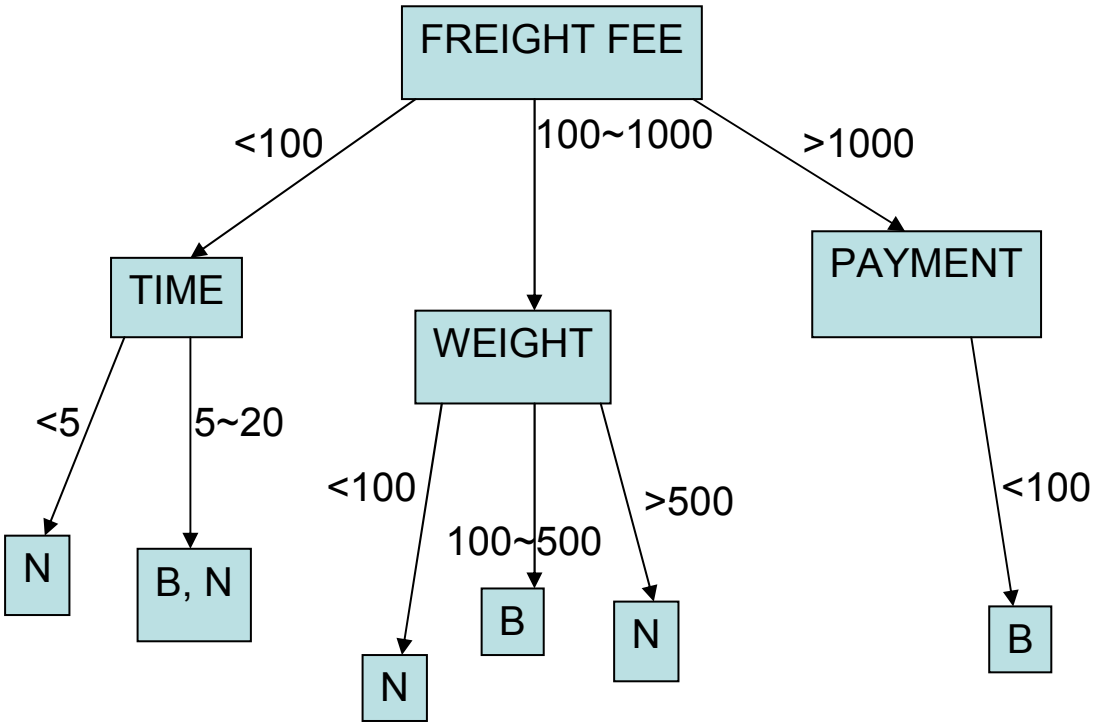


Figure 6.2(a) Decision tree for Customer Dispatch Goods

CHAPTER 7
IMPLEMENTATION

7.1 Coding

```
#include        <stdio.h>
#include        <stdlib.h>
#include        <math.h>
#include        <string.h>

#include        "classes.h"
node root;
#define DEBUG        1        /* change to 0 for avoiding debug messages. */

void error(char *str){
        printf("Error: %s\n",str);
        exit(EXIT_FAILURE);
}
int main()
{
        /* Dynamically allocating the table. */
        int **rptr;
        int *ptr;
        int i,j;
        int b,n;
#ifdef DEBUG
        printf("Building table and reading entries.\n");
#endif
                /* Allocating memory for the array. */
        if((ptr=(int *)malloc(nrows*ncols*sizeof(int)))==NULL)
                error("Cant' allocated table for data");
                /* allocating mem for pointer to rows. */
```

```
            if((rptr=(int**)malloc(nrows*sizeof(int *)))==NULL)
                error("Can't allocate mem for row pointers.");
        for (i=0;i<nrows;i++)
                rptr[i]=ptr+(i*ncols);
        b=n=0;
        /* Enter the table from the user. */

/* Uncomment this code and comment the below code is data is to be read from keyboard.
        for(i=0;i<nrows;i++)
                for(j=0;j<ncols.j++)
                        scanf("%d",&rptr[i][j]);
*/
        /* Reading input from a file.
                Manually entering 19*5 inputs every time can lead a
                        person to incurable mental illness. */
        FILE *fp;
        if((fp=fopen(".\\input.nish","rb"))==NULL)
                error("Can't open input file.");
        for (i=0;i<nrows;i++){

        fscanf(fp,"%d\t%d\t%d\t%d\t%d\n",&rptr[i][0],&rptr[i][1],&rptr[i][2],&rptr[i]
[3],&rptr[i][4]);
                if(rptr[i][4]==0) b++; else n++;
        }
        fclose(fp);

        /* Start building the tree. */
        /* Initialize the root ie property having highest gain in full tree. */
        root.num=highest_IG(rptr,nrows,ncols);
        root.nr_childs=nr_class;
```

```
        root.childs=(node*)malloc(root.nr_childs*sizeof(node));
        root.nr_leafs=0;
        root.leaves=NULL;
        used[root.num]=1;
        /* Build the rest of the tree. */
        build_tree(&root,rptr);
        /* print the Tree. */
        traverse_tree(&root);
        getch();
        return EXIT_SUCCESS;
}

/* Finds I(b,n) value. */
double I(int b,int n){
        if(b==0 && n==0) return (double)0.0;
        int sum=b+n;
        double result;
        result=((double)(pow((double)b/sum,alpha)+pow((double)n/sum,alpha)-
1))/(1-alpha);
        return result;
}
/*
        This function if the heart of whole programs for a given table this funtion
        Returns the nr of the property having the highest Info Gain.
        Initially it is used on the full table to find root.
        Then the extracted sub table is passed to it to calculate rest of nodes.
        Return the number of the property having the highest gain in the
        given table with nr rows and nc columns.
*/
int highest_IG(int **table,int nr,int nc)
{
```

```
/* Nr of properties are nc-1 */

#ifdef DEBUG
        printf("\nFinding the property having highest information gain.\n");
#endif

        int b,n,i;
        double gain=0.0;
        double temp,best_ig;
        int best_p=-1;
        temp=best_ig=0.0;
        /* Find gain of full table. */
        find_bn(table,nr,nc,&b,&n);
        gain=I(b,n);

#ifdef DEBUG
        printf("Information Gain : %lf \n",gain);
#endif

        /* for each property find the entropy then difference with gain and find
                the property having highest IG. */
        for(i=0;i<nc-1;i++){   /* For each property. */
#ifdef  DEBUG
        printf("\tProperty %d",i);
#endif
                temp=gain-E(i,table,nr,nc);    //calculating information gain
#ifdef DEBUG
        printf("\t Info Gain : %lf \n",temp);
#endif

                if(temp>best_ig){
```

```
                        best_ig=temp;
                        best_p=i;
                }

        }
        if(best_p ==-1){
                        for(i=0;i<4;i++)if(used[i]!=1) best_p=i;
        }
#ifdef DEBUG
        printf("\tBest Information Gain for Property %d\n",best_p);
#endif

        return best_p;
}

/* find b and n in whole given table. */
/* Nr of B type customers are returned in b and Nr of N in n. */
void find_bn(int **table,int nr,int nc,int *b,int *n){
        int i;
        *b=*n=0;
        for (i=0;i<nr;i++){
                if(table[i][nc-1]==0) (*b)++;
                else if (table[i][nc-1]==1) (*n)++;
                else ;
        }
}

/* Given a table this function calculates the Entropy for given property. */
/* Returns the calculated property. p is the property as input. */
double E(int p,int **table,int nr,int nc){
```

```
        int ret;
        ret=calc_bn(table,nr,nc,p);
        if(ret!=0) error("can't calculate \n");

        double temp=0.0,temp1;
        int total;
        int i;
        for (i=0;i<3;i++){

                total=res[i][0]+res[i][1];
                temp1=(double)total/(double)nr;
                temp+=temp1*I(res[i][0],res[i][1]);
        }
#ifdef DEBUG
        printf("\t Entropy %lf ",temp);
#endif
        return temp;
}
/* Given a Row and a property it checks whether this entry belongs to class c. */
int check_class(int *row,int p,int c){
        if(c==0) return (*(row+p) < classes[p][0]);
        else if (c==1) return (*(row+p) > classes[p][0] && *(row+p) < classes[p][1]);
        else if (c==2) return (*(row+p) > classes[p][1]);
        else return (-1);
        return -1;
}
/* Given a Row and a property This function returns the Class it belongs to. */
int get_class(int *row,int p){
                if(*(row+p) < classes[p][0]) return 0;
        else if (*(row+p) > classes[p][0] && *(row+p) < classes[p][1]) return 1;
```

```
                else if (*(row+p) > classes[p][1]) return 2;
                 else return -1;
}
/* This function is used when we're calculating the best IG. "
/* A table with 'nrow' rows and 'ncol' columns is given as input.
        The property 'prop' will have three classes and Nr of N and B type customers
        are required for each class. This function finds the Nr of N and B in each Class.
*/
int calc_bn(int **table,int nrow,int ncol,int prop){
        int i;
        int j;

        for(i=0;i<3;i++)
                for(j=0;j<2;j++) res[i][j]=0;

        for (i=0;i<nrow;i++){
                if(table[i][4]==0)
                        res[get_class(table[i],prop)][0]++;
                else if(table[i][4]==1)
                        res[get_class(table[i],prop)][1]++;
                else ;
        }
        return 0;
//      return result;
}

/* This function builds the rest of the tree after root node is found. */
void build_tree(node *root,int ** table){
        /* make a temp table which is a table of entries of each class of root. */
        int num;int i,j;
        int **rptr,*ptr;
```

```
        int num_class;
                /* Allocating memory for the array. */
                        if((ptr=(int *)malloc(nrows*ncols*sizeof(int)))==NULL)
                                        error("Cant' allocated memory for subtable");
                                /* allocating mem for pointer to rows. */
                        if((rptr=(int**)malloc(nrows*sizeof(int *)))==NULL)
                                error("Can't allocate mem for row pointers.");
                for (i=0;i<nrows;i++)
                rptr[i]=ptr+(i*ncols);
                int ret;

        for(num_class=0;num_class<3;num_class++){

#ifdef DEBUG
                printf("Sub root node %d \n",num_class);
#endif
                num=num_entries(table,root->num,num_class);

                j=0;
                /* Extract the sub table for each class. */
                for(i=0;i<nrows;i++){
                        if(get_class(table[i],root->num)==num_class)
                        memcpy(rptr[j++],table[i],ncols*sizeof(int));
                }
                root->childs[num_class].num=highest_IG(rptr,num,ncols);
                root->childs[num_class].nr_childs=0;
                root->childs[num_class].childs=NULL;
                root->childs[num_class].nr_leafs=3;
                root->childs[num_class].leaves=(struct      leaf*)malloc(3*sizeof(struct
leaf));
                used[root->childs[num_class].num]=1;
```

```
                /* Fill the leaf nodes. */
                 ret=calc_bn(rptr,num,ncols,root->childs[num_class].num);
                 if(ret!=0) error("can't calculate \n");
                 for(j=0;j<3;j++)        /* 3 classes of current property. */{
                         if(res[j][0]>0 && res[j][1]>0)
                                 root->childs[num_class].leaves[j].flag=Both_NB;
                         else if (res[j][0]>0 && res[j][1]==0)
                                 root->childs[num_class].leaves[j].flag=ONLY_B;
                         else if (res[j][0]==0 && res[j][1]>0)
                                 root->childs[num_class].leaves[j].flag=ONLY_N;
                         else
                                 root->childs[num_class].leaves[j].flag=NONE;
                 }
        }
        free(rptr);
        free(ptr);
}
/* Find the nr of entries for given class 'c' and property 'p' in main table. */
int num_entries(int **table,int p,int c)
{
        int i;
        int num=0;
        for (i=0;i<nrows;i++)
                 if(get_class(table[i],p)==c)
                         num++;
        return num;

}
char *print_leaf(int flag){
        char *str_none="None";
        char *str_onlyb="Only B";
```

```
        char *str_onlyn="Only N";
        char *str_both="Both N & B";
        char *str_error="Error";
        if(flag==NONE) return str_none;
        else if (flag==ONLY_B) return str_onlyb;
        else if (flag==ONLY_N) return str_onlyn;
        else if (flag==Both_NB) return str_both;
        else return str_error;
}
void traverse_tree(node * root){
        int i,j;
        printf("root node:\n\t property %d Nr of child nodes %d\n",root->num,root-
>nr_childs);
        printf("Sub nodes \n");
        for (i=0;i<3;i++){
                printf("\tSub Node %d Property %d Nr of leaf nodes %d \n",
                                        i,root->childs[i].num,root->childs[i].nr_leafs);
                printf("\tLeaf Nodes\n");
                for(j=0;j<3;j++){
                        printf("\t\tLeaf Node %d %s\n",j,print_leaf(root-
>childs[i].leaves[j].flag));
                }
        }
}
```

/* Defining of "classes.h" Header file */

```
int nrows=19;
int ncols=5;
float alpha=0.25;
int nr_class=3;          /* number of classes for each property. */
/* two different types of nodes are there normal nodes and leaf nodes. */
/* the number of child corresponds to the class. for eg child no. 1 means corresponding to class1. */
struct leaf;
typedef struct Node{
        int num;                                    /* number corresponding to property. */
        int nr_childs;                              /* number of child nodes always equal to nr of classes. */
        struct Node* childs;            /* pointer to child nodes. */
        int nr_leafs;                               /* nr of leaf nodes from this node. equal to nr of classes. */
        struct leaf * leaves;           /* Pointer to the leaf nodes. */
}node;

struct leaf {
        int flag;
};
int used[4]={-1,-1,-1,-1};
/* flag field in leaf node. */
#define NONE        -1
#define ONLY_N      0
#define ONLY_B      1
#define Both_NB     2
```

```
int res[3][2]={{0}};
int classes[4][2]={{100,1000},
                   {100,2000},
                   {100,500},
                   {5,20}};
void build_tree(node*,int **);
void traverse_tree(node*);
int check_class(int *row,int p,int c);
/* Each property has 3 classes (0,1,2) and each class has some n customers and some
b customers.
    this function returns the number of n customers and num of b customers for
each class
    for given property p in the table 'result'.
    thus result [1][0] will contain number of b's in class 1 of property and
result[1][1] will
    give number of n customers in class 1.
*/
int calc_bn(int **table,int nrow,int ncol,int prop);

/* Calculates the information for a given table. */
double I(int b,int n);

int get_class(int *row,int p);

double E(int p,int **table,int nr,int nc);
int highest_IG(int **table,int nr,int nc);
void find_bn(int **table,int nr,int nc,int *b,int *n);

int num_entries(int **table,int p,int c);
```

7.2 Explanation of Coding.

The properties are actually columns of the input table so property and column are the same thing. In context of given table, property 0 corresponds to Freight Fee,
Property 1 is payment, 2 is weight and property 3 is Dispatch times.

The input can be given from keyboard but it is very tedious so I read the input from a file (input file contains the value for the table). Name of the file is input.nish.
The input file should be written in same format. The entries in the rows are separated by tabs and the rows are separated by carriage return (new line or enter).

First the Info gain is found from whole table and the root node is selected.
Then the number of child nodes is equal to the number of classes. So the sub table corresponding to each class is extracted and the above procedure is repeated.
I noticed that the classes are separated as
for input x
Class 1 if x<a
class 2 if x>=a and x<=b
class 3 if x>b

This means there are two boundary values a and b. If for each property we know a and b then we can know the classes. So I have used 2-D array classes for this purpose.
Most of the functions and techniques I have used in this program are generalized, So full fledged general implementation can be easily built from this program. Like, the class boundary values can be changed to change class definitions. highest_IG can be used to find best information gain in any table etc.

The code I have commented is extensive so that you can easily understand. Compile the program and run it, input file and header file classes.h must be in same folder as source code itself. Input file can be opened in any text editor. In the output property “no” is printed for the nodes.

7.3 Output

7.3.1 In C compilation form,

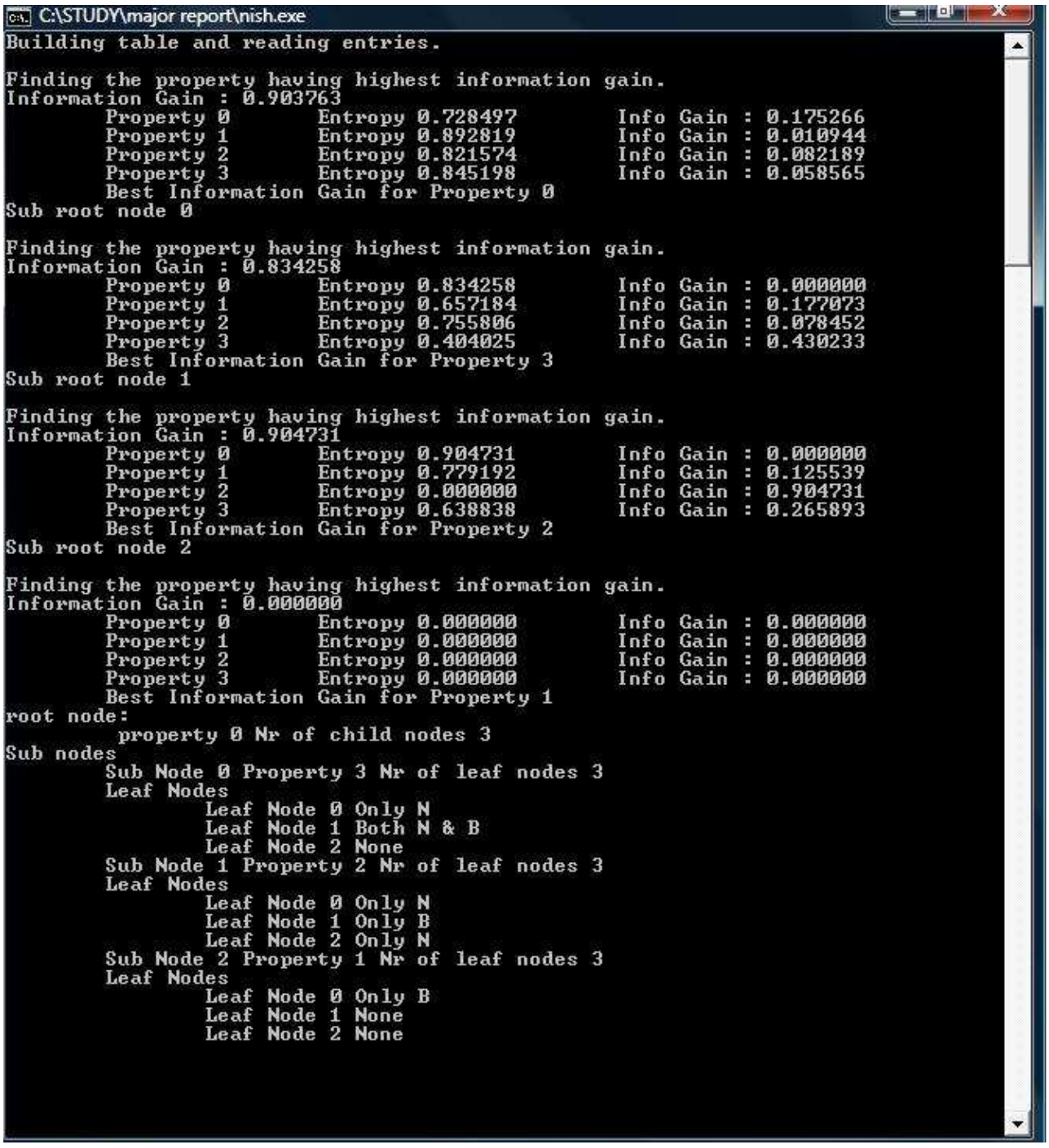


Figure 7.3(a) Output in C compilation format

7.3.2 In Decision Tree form,

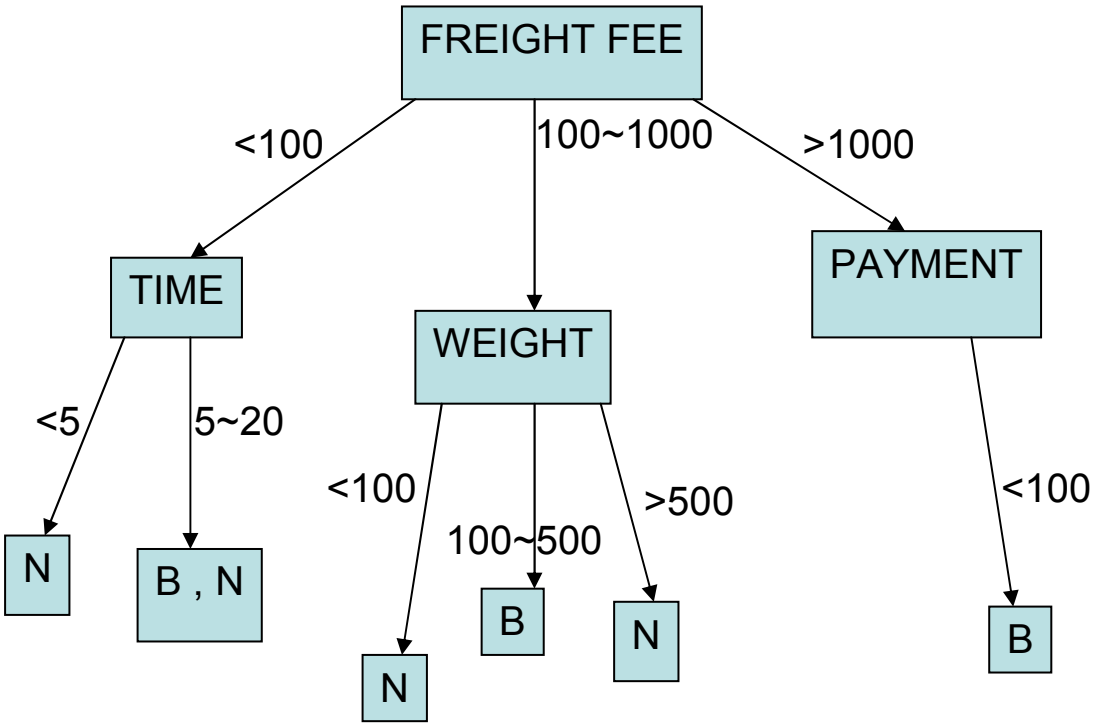


Figure 7.3(b) Output in Decision Tree format

CHAPTER 8

CONCLUSION AND FUTURE WORK

8.1 Conclusion

Data mining as a technology can be used to analyze the customer data to find their exact need. This will help us to give more value to customer by increasing their information, also help in providing high quality services to them by understanding them. The decision tree tells what customers want the most. In this thesis ID3 algorithm is used, but modification is done. Instead of using Shannon Entropy, Havrda and Charvat Entropy has been used to find the information of different properties which is used as the node of decision tree. This modification has reduced the size of tree as well as decreased the rules, which will help to understand customer characteristics by which company growth and profit can be increased. I have proposed a new algorithm, i.e. instead of Shannon entropy I have used the Havrda and Charvat Entropy. As a result for lower value of alpha (α) =0.25, tree is small and less complex as compared to the use of Shannon entropy. To conclude, I can say that if we want to get less number of node and leaf in a tree and to make it more effective and less complex, we can use the Havrda and Charvat entropy instead of Shannon entropy and value of alpha (α) less than one will give decision tree with less number of nodes.

8.2 Scope for future work

In this research, the value of alpha (α) has been put as 0.25, but for further research we can take varying values of alpha (α) which may give different trees.

Various values of α have already been put in this study i.e. alpha (α) = 0.5, 0.75, 0.99, 5, 10, 100 which gave same tree as in the case of alpha (α) = 0.25. But, it has been observed that in case of alpha (α) = 2, 3, 4 the tree turned out to be different with more number of leaf and high complexities.

Also, Instead of using Havrda and Charvat Entropy, different Entropies can also be used for further research like Arimoto, Sharma-Mittal, Taneja, Sharma-Taneja, Ferreri, Sant'anna—Taneja ,Picard, Aczel-Daróczy.

BIBLIOGRAPHY

[1] Shannon Entropy, Renyi Entropy, and Information P.A. Bromiley, N.A. Thacker and E. Bouhova-Thacker. Imaging Science and Biomedical Engineering, School of Cancer and Imaging Sciences, Stopford Building, University of Manchester, Oxford Road, Manchester M13 9PT, U.K, 2004.

[2] The Applied Research based on Decision Tree of Data Mining In Third-Party Logistics Wang Qian, Wu Yaohua,Xiao Jiwei The Logistics Institute Shandong University Jinan, Shandong Province,China & Pan Guang-feng Department of Computer engineering Shandong Province, China. Proceedings of the IEEE International Conference on Automation and Logistics, Jinan China, August 18 - 21, 2007.

[3] J. Han and M. Kamber, "Data Mining: Concepts and Techniques", Morgan Kaufmann. Second edition published on 8 Jul 2007

[4] Havrda and Charvat Entropy Based Genetic Algorithm for Traveling Salesman Problem Baljit Singh, Arjan Singh and Akashdeep Department of Computer Science & Engineering, BBSB Engineering College, Fatehgarh Sahib-140407, Punjab, India. IJCSNS International Journal of Computer Science and Network Security, VOL.8 No.5, May 2008

[5] J. Havrda, F. Charvat, Kybernetika 30, 1967.

[6] RuleQuest Research Pty Ltd. Data mining tools see5 and c5.0.http://www.rulequest.com/see5-info.html,2003.

[7] J.R. Quinlan, editor. C4.5: Programs for Machine Learning. Morgan Kauffman, 1993.

[8] Wing-Ho Shum, Kwong-Sak Leung, and Man-Leung Wong. Learning functional dependency networksbased on genetic programming. In *ICDM05, Proceedings of IEEE International Conference on Data Mining*, pages 232–230, 2005.

[9] E. Altman, “Financial ratios, discriminant analysis, and the prediction of corporate bankruptcy,” *J. Finance*, vol. 23, pp. 589–609, 1968.

[10] T. K. Sung, N. Chang, and G. Lee, “Dynamics of modeling in data mining: interpretive approach to bankruptcy prediction,” *J. Manag. Inform. Syst.*, vol. 16, pp. 63–85, 1999.

[11] C. Y. Shirata, “Peculiar behavior of Japanese bankrupt firms: discovered by AI-based data mining technique,” presented at the 4th Int. Conf. Knowledge-Based Intelligent Engineering Systems and Allied Technologies, Tokyo, Japan, 2000.

[12] Advancing Knowledge Discovery and Data Mining by Qi Luo, School of Electrical and Information Engineering, Wuhan Institude of Technology, Wuhan, 430073, China Information Engineering School, Wuhan University of Science and Technology Zhongnan Branch, Wuhan, 430223, Hubei, China whluo2008@gmail.com,2008.

[13] C. Neely, P. Weller, and R. Dittmar, “Is technical analysis in the foreign exchange market profitable? A genetic programming approach,” *J. Financial Quant. Anal.*, vol. 32, pp. 405–426, 1997.

[14] F. Westerhoff and C. Lawrenz. Explaining exchange rate volatility with a genetic algorithm. Computing in Economics and Finance. [Online] Available: http://econpapers.hhs.se/paper/scescecf0/325.htm,2000.

[15] Group-Wise Point-Set Registration Using a Novel CDF-Based Havrda-Charvát Divergence Ting Chen · Baba C. Vemuri · Anand Rangarajan · Stephan J. Eisenschenk Int J Comput Vis 86: 111–124 DOI 10.1007/s11263-009-0261-x, 2010.

[16] An Implementation of ID3 - Decision Tree Learning Algorithm Wei Peng, Juhua Chen and Haiping Zhou Project of Comp 9417: Machine Learning University

of New South Wales, School of Computer Science & Engineering, Sydney, NSW 2032, Australia weipengtiger@hotmail.com ,1997.

[17] Breiman, L., Friedman, J., Olshen, L and Stone, J. Classification and Regression trees. Wadsworth Statistics/Probability series. CRC press Boca Raton, Florida, USA, 1984.

[18] Du, W., Zhan, Z. Building decision tree classifier on private data, Proceedings of the IEEE international conference on Privacy, security and data mining, pp.1-8, Maebashi City, Japan,2002.

[19] Gehrke, J., Ramakrishnan, R., Ganti, V. RainForest - a Framework for Fast Decision Tree Construction of Large Datasets. Proceedings of the 24th VLDB conference, New York, USA. pp.416- 427, 1998.

[20] Kufrin, R. Decision trees on parallel processors. In J. Geller, H. Kitano, and C. B. Suttner, editors, parallel Processing for Artificial Intelligence 3 Elsevier Science,1997.

[21] Lewis, R.J. (200). An Introduction to Classification and Regression Tree (CART) Analysis.Annual Meeting of the Society for Academic Emergency Medicine, Francisco, California, 2000.

[22] Peng, W., Chen, J. and Zhou,H. An Implementation of IDE3 Decision Tree Learning Algorithm. From web.arch.usyd.edu.au/wpeng/DecisionTree2.pdf Retrieved date: May 13, 2009.