Online Rule Generation Software Process Model

Rajni Jain[1], Satma M C[2], Alka Aroa[3], Sudeep Marwaha[4] and R C Goyal[5]

Submitted in April, 2012; Accepted in October, 2012

Abstract - For production systems like expert systems, a rule generation software can facilitate the faster deployment. The software process model for rule generation using decision tree classifier refers to the various steps required to be executed for the development of a web based software model for decision rule generation. The Royce's final waterfall model has been used in this paper to explain the software development process. The paper presents the specific output of various steps of modified waterfall model for decision rules generation.

Index Terms - Software Process model, Modified waterfall model, Decision Rule, Decision Tree

1. INTRODUCTION

Classification is the discovery of a predictive learning model that classifies a data item into one of several predefined classes [4]. The classification model can predict the class of objects whose class label is unknown. It is also called classifier [8,16]. Patil et. al. have done work on fault classification of mechanical System using self organizing techniques [16]. Verma et. al. have used rough set techniques for 24 hour knowledge factory [17]. But none of these algorithms are available online. We have made attempt to develop a software process model for online rule generation. The model can be used by other researchers for their own algorithms to make online software.

A decision tree is a classifier expressed as a recursive partition of the instance space. It consists of nodes that form a rooted tree. The leaf nodes denote class labels or class distribution. The non-leaf nodes denote a test on an attribute and branches denote outcome of the test [12]. An online rule generation software is required by researchers and data mining personnel who have to generate rules to facilitate the development of expert systems or pattern recognizing in various domains. Presently researchers use their individual program running on their desktop [18]. Online software enables the easy access to it using the default browser on the client machine. But it is not available yet. So there is a need to develop the online decision rule generation software referred to as 'GenRule'.

To build software, it is important to go through a series of predictable steps. The steps are like a roadmap that helps to develop a high quality system. This roadmap is also called a software process.

[1]*NCAP, Pusa, New Delhi-12*
[2, 3, 4, 5] *IASRI, Pusa, New Delhi-12*
E-mail: [1]rajni@ncap.res.in and [2]satmaktm@gmail.com

The software development life cycle (SDLC) is the entire process of formal, logical steps taken to develop a software product [13]. There are five phases that are part of the SDLC [3]. These phases are requirements definition, design, coding, testing and maintenance. SDLC models are created based on the order in which they occur and the interaction between them [5]. The modified waterfall model developed by Royce has been used in the development of GenRule.

The present paper is an attempt to identify and document the requirements for developing the online software described above. The rest of the paper is organized as follows. The section 2 presents the software process model concepts. Its sub-sections deal with the various phases of SDLC namely requirement analysis, design, coding, testing and maintenance. Section 3 presents the conclusion.

2. SOFTWARE PROCESS MODEL

A software process model is an abstract representation of the architecture, design or definition of the software process [14]. There are varieties of software development process models to show how organizing the process activities can make the development more effective [1]. One of the basic software process models is waterfall model. But it is not flexible. Its phases are strictly linear [9]. So the Royce's modified final waterfall model has been used in the development of the rule generation software using decision tree classifier.

The advantage of the modified waterfall model is that it is a more relaxed approach to formal procedures, documents and reviews. It also reduces the huge bundle of documents. Due to this, more time can be devoted to coding without bothering about the procedures. This in turn helps to finish the product faster [9]. The different phases in the Royce's final waterfall model [15] with reference to the development of GenRule are explained in the subsequent sub sections.

2.1 Requirement Analysis

Requirements are set of functionalities and constraints that end-user expects from the system. For GenRule, users are mainly developers of expert systems, students and data mining researchers who are interested in generating rules from data. Recently, expert systems are being developed for various agricultural crops like wheat, maize, mustard etc. In production systems like expert systems, knowledge is required to be fed in the form of rules. Usually these rules are made at the expense of valuable time of experts. In the field of agriculture, vast amounts of research data are generated every day and to convert those huge amounts of data to useful and knowledgeable decision rules that can help in crucial decision making, decision rule generation software is required. Consequently the experts could spend their time only on

validating the generated rules that are provided along with accuracy measures. Rule generation using decision tree classifier is providing additional benefits of visualization effect in the form of decision tree, which adds to the understandability of rules.

Information about user's requirements was gathered from literature [11,7] and also by consulting the prospective users like researchers involved in development of expert systems. It helped to know what should be accomplished by the application. These requirements were further analyzed for their validity and the feasibility of incorporating them in GenRule were explored.

User requirements are broadly categorized into two types namely functional and non-functional requirements. Functional requirements describe what the software should do. They include user requirements, input requirements, computational requirements, output requirements, exception handling etc. Non-functional requirements refer to the requirements that are not directly concerned with the specific functions delivered by the system [14]. They are broadly categorized into performance requirements and system requirements. The first one deals with the level of performance required by users. It includes various other requirements like usability, human factor and security issues etc. The system requirements for GenRule are identified at three levels namely, client level, server level and at the programmer level.

2.1.1 Functional Requirements

The sequence diagram facilitates the understanding of user requirements [14]. On the basis of interaction with various categories of users, sequence diagram for GenRule is presented in Fig 1. It explains the sequence of actions to be followed by user to generate rules using GenRule. The sequence diagram clearly exhibits the following functional requirements of the users.

i. The user has to be validated by checking the user name and password. Only the valid users should have the facility to access the software.

ii. Input Requirements: Facility should be provided to input the data in excel or CSV (Comma Separated Values) file format. The user has to enter the partition preference of the dataset and select the required attributes from the available attributes and finally the target attribute from the selected attributes.

iii. Computational Requirements:

(a)The input data should be validated for non-categorical and missing values. If they are present, exception will occur and error message will be displayed. There should be partition of input data into training and test dataset randomly according to the partition preferences given by the user. It should have the provision to select the required attributes from the whole set of attributes. It should also provide facility to the user to select the class attribute from the already selected list of attributes.

(b)User should have the facility to classify future data instances if its classifier model is already built in the software. The user may be allowed to choose a classifier already built and stored by GenRule. If the model is not learned, user has to generate the rules first and go for prediction of future instances. (c)Data flow model is an intuitive way of showing how data is processed by a system. The data flow diagram shown in Fig.2&3 illustrates how the data flows through a sequence of processing steps in GenRule to generate decision rules. The graphical user interface is well explained with the help of use-case diagram (Fig. 4-7). Use-case diagram identifies the user interactions with the software. The various interactions involved are described in Fig.5, 6 and 7 respectively. The generation of rules from the training data using ID3 algorithm in the process 1.5 (Fig.2) is further explained using the data flow diagram in Fig 3.

iv.Output Requirements: Facility should be provided to display the generated rules and the corresponding decision tree view along with evaluation measures like rule coverage, rule accuracy, precision, recall, F-measure, confusion matrix, training accuracy and test accuracy. Exporting and saving facility should be provided in Excel, text and XML file format for the generated rules for further implementation. For improving the understandability of the rules, the corresponding decision tree should be displayed and there should be provision to save it in XML format.

v.On logging out, user should return to the home page.

2.1.2 Non-Functional Requirements

i. The software should be friendly and available on the internet, with the authentication of user name and password.

ii. The software should meet all kinds of user requirements efficiently.

iii. It should provide accurate output in all aspects.

iv. Online help facility should be included.

v. Results should be reliable.

vi. Client level specification: Any browser with latest facility like IE6 or higher, Excel 2003 or 2007.

vii. Server level specification: Windows 7, IIS 7.0, Microsoft .NET Framework Version 3.0, 2 GB RAM, 2.53 GHz Processor, 320 GB Hard Disk,

viii.Programmer level specification: Microsoft Windows 7, Visual Studio 2008, IIS 7.0, 2 GB RAM, Excel 2007, IE8, 320 GB Hard Disk.

The ID3 algorithm is a recursive algorithm for building decision tree and decision rules [10]. As a part of requirement analysis, it is important to get an understanding of the basic ID3 algorithm used. The ID3 algorithm is explained using flow chart shown in Fig. 8.

2.2 Design

The requirement specifications from first phase were studied in this phase and system design was prepared. System design helped in specifying the hardware and system requirements and also helped in defining the overall system architecture. A set of software design concepts has evolved over the history of software engineering [14].

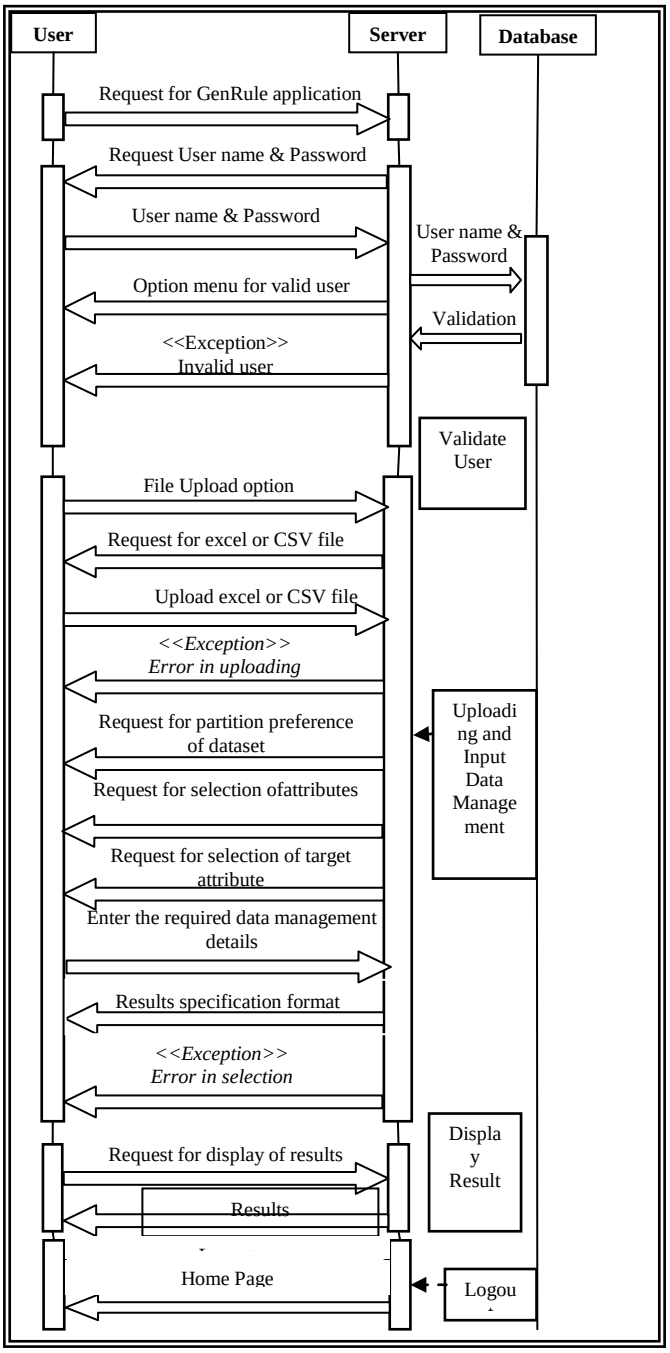

Figure 1: Sequence diagram for rule generation

The design of GenRule is presented with the help of input design, output design, database design and design of modules. Modularity is one of the powerful concepts for software design. Most complex design tasks are solved by breaking them down into manageable part called modules [2].

2.2.1 Input Data Design

Data may be entered to the software using Excel or CSV file. In the data, the columns should represent the attributes and the rows should contain the dataset instances. One of the attributes should be the class attribute. Table1 represent one such sample dataset where lodging is class attribute.

Variety	Crop_duration	Climate	N_fertilizer	Lodging
resistant	early	rainy	low	no
resistant	early	rainy	high	no
tolerant	medium	rainy	low	yes
susceptible	early	rainy	low	yes

Table 1: Sample dataset

Database Design

Database for the system is maintained using MS SQL server. Database should contain a table that is useful to store the generated classifier for subsequent use (Table 2).

Column Name	DataType	Allow Nulls
id	int	no
Parent_id	int	yes
Node name	varchar(50)	yes

Table 2: Classifier table schema

The database constructed corresponding to the sample input data given in the Table 1 is shown in Table 3. The last column denotes the name of the nodes. The id for each node is given in the first column. The parent_id column gives the parent node of each node. If parent_id is zero, it is a root node. This classification table is useful to classify the unseen cases.

id	Parent_id	Node name
1	0	variety
2	1	resistant
3	2	climate
4	3	dry
5	4	yes
6	3	rainy
7	6	no
8	1	susceptible
9	8	yes
10	1	tolerant
11	10	N_fertilizer
12	11	high
13	12	no

Table 3: Classifier table for crop lodging dataset

There is a database namely 'aspnetdb' that contains tables with the login details of users under the sql membership provider functionality of ASP.NET.

2.2.2 Output Design

Outputs for GenRule software are decision rules in table format and decision tree in tree view format. It also computes various evaluation measures like confusion matrix, precision, recall, F-measure, training accuracy and test accuracy. The decision rules output table contains rule id and corresponding to each rule id, the class attribute column and other attributes column with respective values for the rule. The value of '*' given to attributes represents any value of the given attribute. Each rule is associated with its evaluation measures coverage and

accuracy, as coverage explains the data instances covered by a rule and accuracy explains the validity of a rule in the data instances. The coverage and accuracy of ith rule ($Rule_i$) can be computed using the given formula.

$$Coverage(Rule_i) = \frac{ncovers(Rule_i)}{|training\ dataset|}$$

$$Accuracy(Rule_i) = \frac{ncorrect(Rule_i)}{ncovers(Rule_i)}$$

Where ncovers ($Rule_i$) represents the number of data instances satisfying the antecedent of $Rule_i$ and ncorrect ($Rule_i$) represents the number of data instances correctly classified by $Rule_i$.The decision rule output table of the data given in Table 1 should be like the Table 4. Rules should be displayed in classic if-then format also (Table 5). A confusion matrix contains information about actual and predicted classification done by a classification system [6]. The performance of such systems is commonly evaluated using the data in the confusion matrix. Confusion matrix can be worked out for both training and test dataset. All the performance measures computed are functions of the confusion matrix (Table 6). Precision, recall and F-measure can be computed for the two class values.

Id	lodging	variety	Crop_duration	climate	N_fertilizer	Rule Coverage (%)	Rule Accuracy (%)
1	yes	susceptible	*	*	*	28	100
2	yes	resistant	*	dry	*	14	100
3	no	resistant	*	rainy	*	21	100
4	no	tolerant	*	*	high	14	100
5	yes	tolerant	*	*	low	21	100

Table 4: Decision rule table for crop lodging dataset

Rule_id	Decision Rule	Rule Coverage (%)	Rule Accuracy (%)
1	If [variety='susceptible'], then lodging='yes'	28	100
2	If [variety='resistant'], [climate='dry'], then lodging='yes'	14	100
3	If [variety='resistant'], [climate='rainy'], then lodging='no'	21	100
4	If [variety='tolerant'], [N_fertilizer='high'], then lodging='no'	14	100
5	If [variety='tolerant'], [N_fertilizer='low'], then lodging='yes'	21	100

Table 5: classic if-then rule for crop lodging dataset

Class values	Recall	Precision	F-measure	Accuracy (%)
yes	0.5	1	0.66	66
no	1	0.5	0.66	

Table 6: performance measures on crop lodging classifier

2.3 Coding

The functionalities of GenRule can be achieved by developing various modules and assigning different tasks (Table 7).
The implementation of the defined modules in the design phase was done using classes and methods (Table 8).

2.4 Testing, Integration and Maintenance

Each of the modules (Table 7) should be tested for their functionality individually during the unit testing for the desired output. These units should be integrated into a complete system during integration phase and should be tested to check if all modules/ units coordinate between each other and the system as a whole behaves as per the specifications. Bottom up integration was used for combining various modules [2].
The software has to be maintained as long as it is used for various applications. There should be proper documentation to facilitate the operation and maintenance as and when required.

Module Name	Description
Registration	Provide facility of sign up to new user
Login	Provide facility of login to users
Update	An option for change of password
Decision Rule Generation	The main module, which generate decision rules from the training dataset along with rule evaluation measures like rule coverage and accuracy
Decision Rule Validation	The module which validates the generated decision rules using the test dataset and provide evaluation measures
Decision Tree	Constructs the decision tree corresponding to the generated rule set

Module Name	Description
Classification	Provide the prediction of target attribute values for the future datasets which are unclassified. It is useful if the classifier is preexisting.
Prediction	Provide the prediction of target attribute values for the future datasets which are unclassified, successively after building the classifier
Sample Data	Provide sample data for the user
Contact Us	Provide contact details of the development team
Help	Provide online help about software

Table 7: Description of various modules in GenRule

Class name	Description
getSheetName	Establish connection to Excel file and get all sheet names from Excel file
fillInDataSet	Data present in various sheets of Excel file is uploaded into different dataset
DStoDT	Convert dataset to corresponding data table
Attribute	Designate column names as attributes with certain properties
ID3	Class that performs the ID3 algorithm functionality
TreeNode	Create tree node for allocating the attributes coming out of the ID3 algorithm

Table 8: Description of different classes in GenRule

3. CONCLUSIONS AND FUTURE SCOPE

Software process model explains the requirement specifications, input design, output design, database design, implementation, testing and maintenance phases. In this paper the software development life cycle of decision rule generation software is presented from its conception to its maintenance and implementation stage. The process model helped to develop GenRule software that can fulfill the requirements of agriculturalresearchers, teachers, students and other data mining personnels handling huge amounts of data. The model will also be helpful in future for any enhancements or maintenance of the software.

The software process explained above can be utilized for many other decision tree algorithms.

REFERENCES

[1]. S. T. Acuna, A. D. Antonio, X. Ferre, M. Lopez, and L. Mate. *The Software Process: Modelling, Evaluation and Improvement, Handbook of Software Engineering and Knowledge Engineering*. World Scientific Publishing Company.2000.

[2]. A. Goswami, N. Arora, and A. Sharma. *Fundamentals of Software Engineering*. Lakhanoal Publishers, Amritsar, India.2010.

[3]. N. Jenkins. *A Software Testing Primer*. Creative Commons, California.2008.

[4]. M. Kantardzic**.** *Data Mining: Concepts, Models, Methods, and Algorithms*. John Wiley and Sons, Inc Publication, New Jersy.2011.

[5]. R. Kissel, K. Stine, M. Scholl, H. Rossman, J. Fahlsing, and J. Gulick. *Information Security*. NIST Special Publication, Gaithersburg. 2008.

[6]. R. Kohavi, and F. Provost. Glossary of Terms. *Machine Learning*, Vol. 30, No.3: 106-110.1998.

[7]. P. Langley and H. A. Simon. *Applications of Machine Learning and Rule Induction*. Institute for the Study of Learning and Expertise.1995.

[8]. A. M. Mahmood, N. Satuluri, and M. R. Kuppa. An Overview of Recent and Traditional Decision Tree Classifiers in Machine Learning. *Indian Journal of Research and Reviews in Ad Hoc Networks*, Vol.1,No.1: 10-12. 2011.

[9]. N. M. A. Munassar, and A. Govardhan. A Comparison Between Five Models of Software Engineering. *IJCSI International Journal of Computer Science*, Vol.7, No.5: 94-101.2010.

[10]. J. R. Quinlan. Induction of Decision Trees. *Machine learning*, Vol.1, No.1: 81-106.1986.

[11]. J. R. Quinlan. *Generating Production Rules from Decision Trees*. Massachusetts Institute of Technology, USA.2007.

[12]. L. Rokach, and O. Z. Maimon. *Data Mining with Decision Trees: Theory And Applications*. World Scientific Publishing Co Pvt Ltd. 2008.

[13]. W. Scacchi. *Process Models in Software Engineering*. John Wiley & Sons, Inc, New York. 2001.

[14]. I. Sommerville. *Software Engineering: A Practitioner's Approach*. Tata McGraw Hill. 2009.

[15]. J. Xiong. *New Software Engineering Paradigm Based on Complexity Science*. Springer Publication. 2011.

[16]. J. K. Patil, P.B. Ghewari and S.S. Nagtilak. Iterative Self Organised Data Algorithm for Fault Classification of Mechanical System, BIJIT - BVICAM's International Journal of Information Technology, issue 5, 3(1). 2011.

[17]. N. Verma, N. Nerma and A.B. Patki. Rough Set Technique for 24 Hour Knowledge Factory. BIJIT - BVICAM's International Journal of Information Technology, issue 7, 4(1), 2012.

[18]. N. Aggarwal, N. Prakash, S. Sofat. Mining Techniques for Integrated Multimedia Repositories: A Review, BIJIT - BVICAM's International Journal of Information Technology, Issue 1, 1(1), 2008.

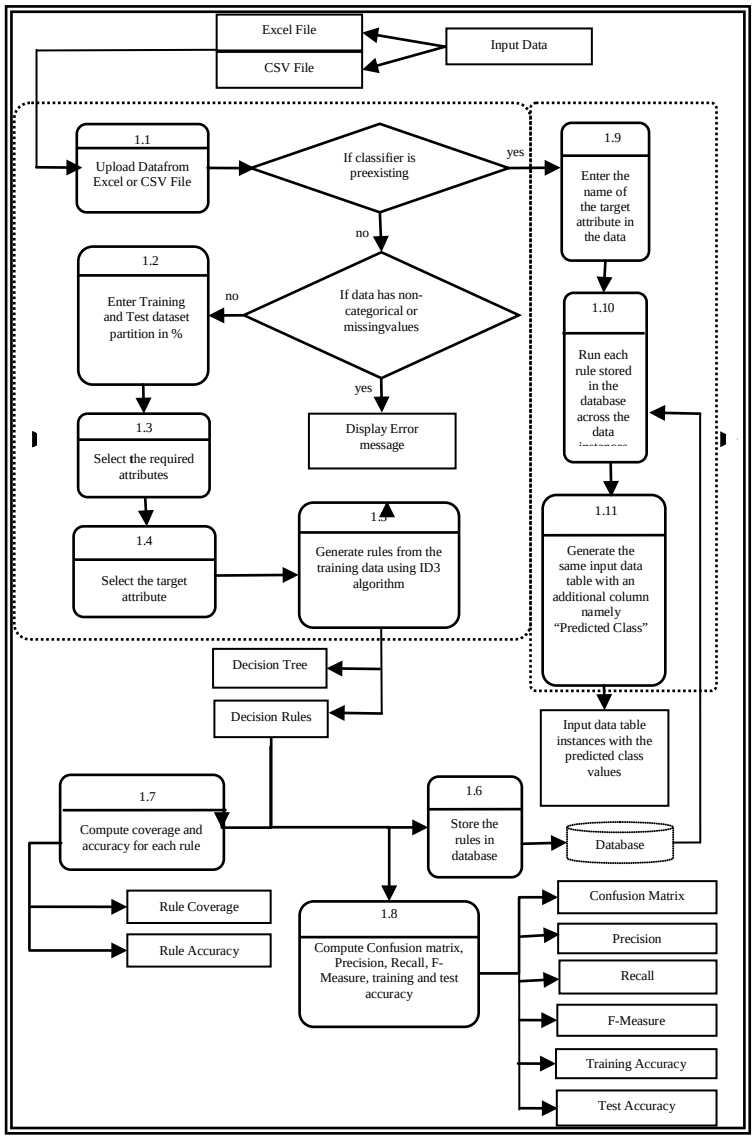

Figure 2: Data flow diagram for GenRule

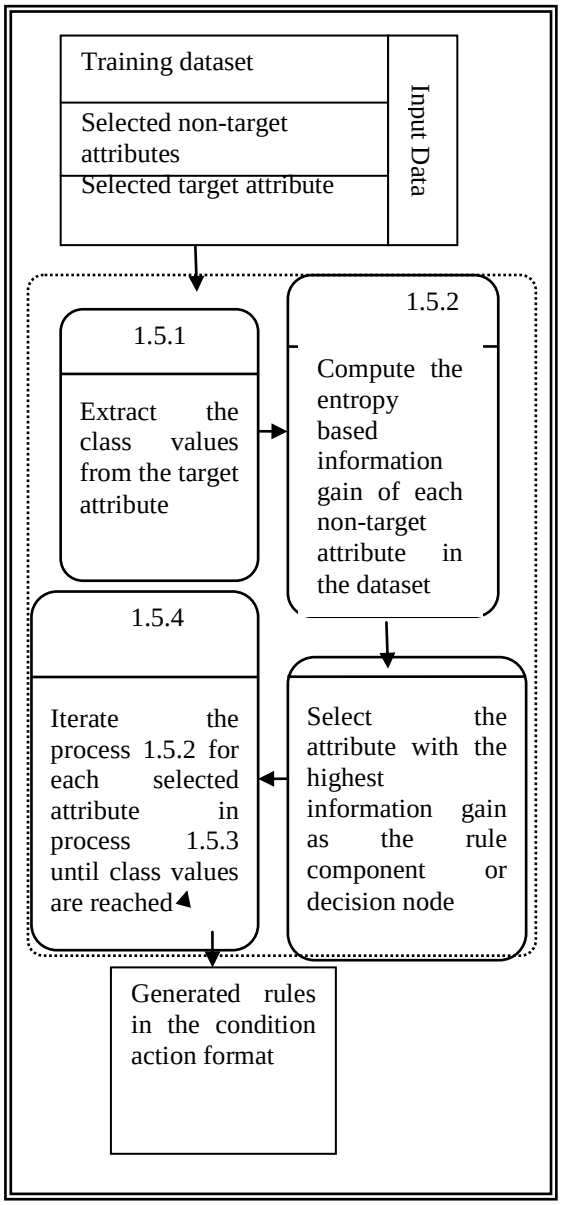

Figure 3: Data flow diagram for process 1.5 in Figure 2

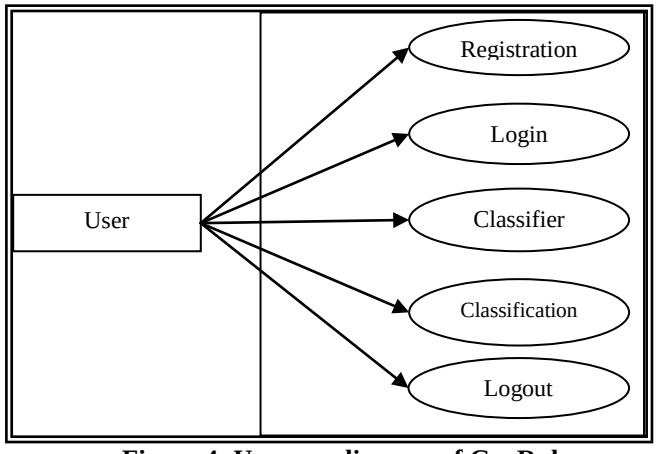

Figure 4: Use-case diagram of GenRule

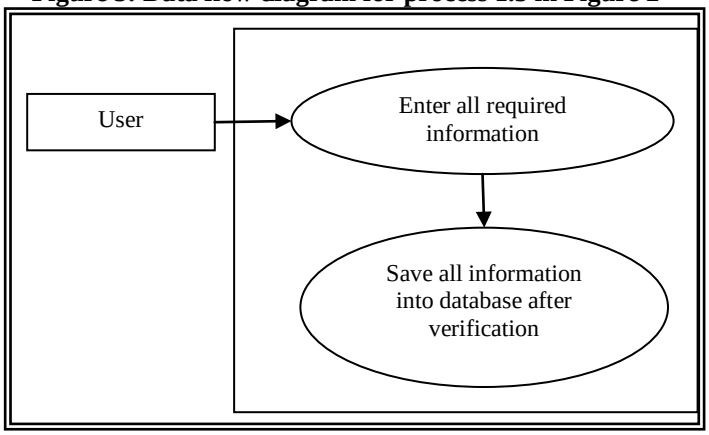

Figure 5: Use-case diagram for registration

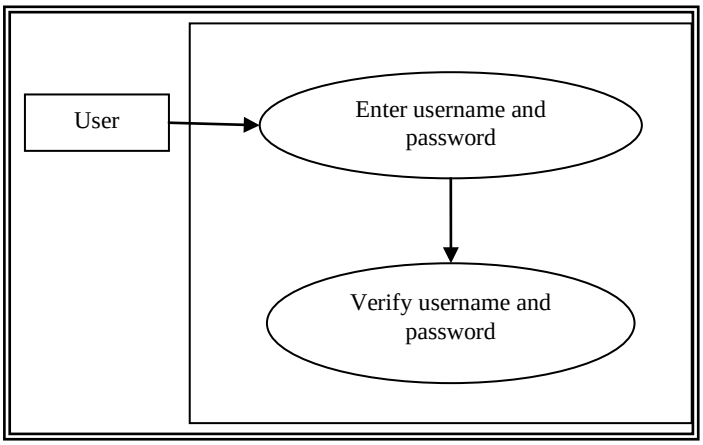

Figure 6: Use- case diagram for login

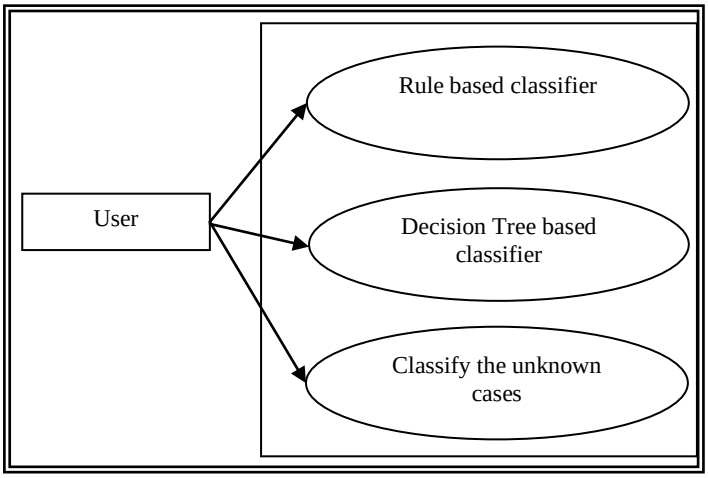

Figure 7: Use-case diagram for classification

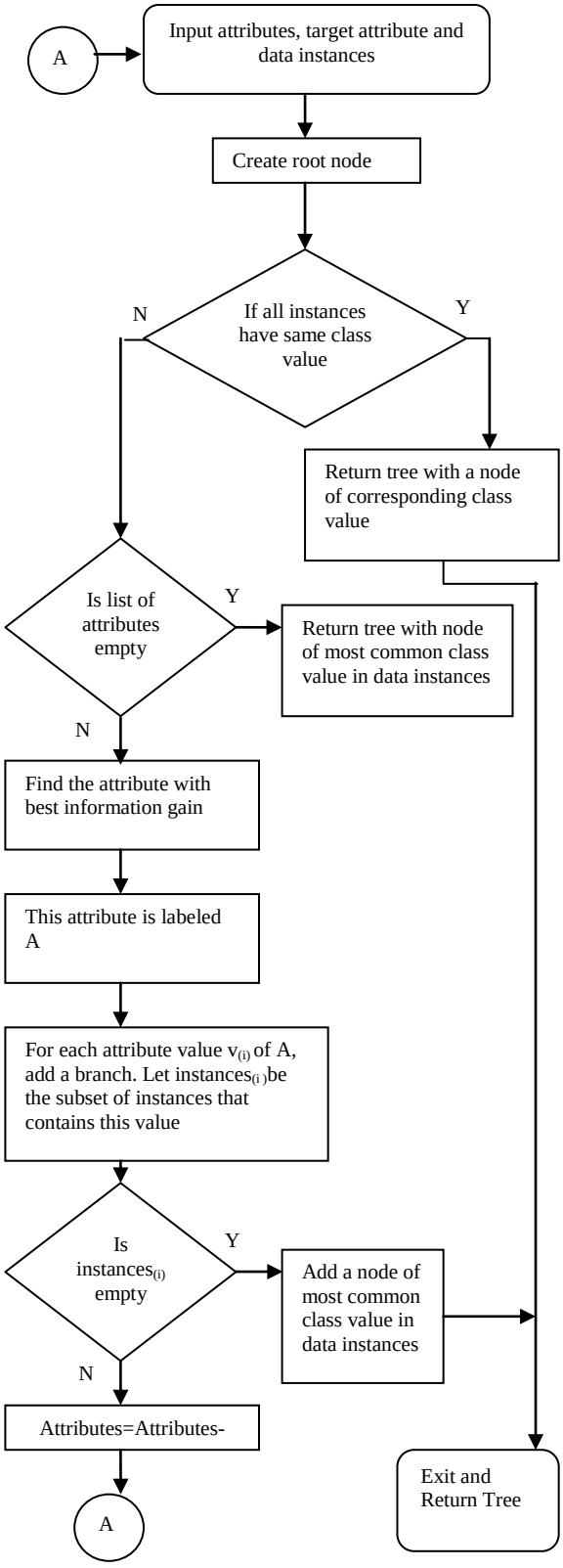

Figure 8: ID3 Algorithm Flowchart