IJRRECS/October 2013/Volume-1/Issue-6/ 991-1001 **ISSN 2321-5461**

INTERNATIONAL JOURNAL OF REVIEWS ON RECENT ELECTRONICS AND COMPUTER SCIENCE

A NOVEL DECISION TREE ALGORITHM FOR HANDLING UNCERTAIN DATA SETS

Budida Satish Kumar[1], P.R.Krishna Prasad[2]

[1]M.Tech Student, Dept of CSE, Vasireddy Venkatadri Institute of Technology, Nambur, Guntur(Dt), A.P, India

[2]Assistant Professor, Dept of CSE, Vasireddy Venkatadri Institute of Technology, Nambur, Guntur(Dt), A.P, India

ABSTRACT:

Existing decision tree algorithms work with data whose values are precise and known prior performing the classification. Uncertainty increases the noisy and reduces the accuracy in the data. We extend improved classifiers to handle data with uncertain information. Uncertainty arises in many applications during data collection and preparation stage. Example includes quantization errors with repeated measurements. By using uncertainty, the data value of a probability distribution is often represented not by one value, but by multiple values forming.. In this project, we present a scalable improved decision tree algorithm for classifying huge dataset with high speed, which requires less scan over the dataset. It overcomes the limitations of Existing C45 algorithm which addresses the scalability and noisy issue and more time for decision tree construction. The proposed Improved C45 algorithm significantly reduces the input and output time and also requires less time to find information for numeric attributes which gives better performance in terms of time complexity. Experimental results shows that proposed approach will acquires less execution time and improves accuracy over the C45 algorithm and attribute selection method is considered.

Keywords: Classification, Gain, Uncertain, Probability.

1. INTRODUCTION :

A primary reason for analyzing statistics mining would be to assist in the analysis of collections of observations of behaviour. Such data are sensitive to collinearity due to unknown interrelations. An unavoidable fact hard drive data mining could be that the (sub-)set(s) of data being analysed most likely are not representative of the entire domain, and therefore will possibly not contain some of certain critical

relationships and behaviours that exist across other parts of a given domain. To tackle this kind of issue, the analysis may be augmented using experiment-based together with other mining approaches, such as Choice Modelling for human-generated data. Of these situations, inherent correlations can easily be either controlled for, or removed altogether, while in the construction of the experimental design. Most classification algorithms perform batch classification, that's, they work towards the dataset directly in memory. More challenging methods often build large data structures which give them a more substantial memory footprint. This larger footprint prevents them away from being put on lots of the larger datasets. No matter if these datasets can be utilized, the complexity of one's algorithm could make the classification task spend time in an inordinately long term. There are resolutions in these problems by using adding more memory or waiting longer for experiments to complete. However both tend to have a cost in time and money and both might not ultimately solve the challenge in the event the dataset cannot in memory in the event the memory is at a maximum. An attribute of most large datasets happens to be the great many hard rive data points that contain similar information. So one alternative is to eliminate redundant information from the dataset to allow a classifier to focus on relevant data that represent a larger number of points. This allows the construction of a classier that correctly models the union expressed by information inside the dataset.Data mining is novel latent and a impressive new technology applied to data warehouses. Organizations must handle the challenge expertise overload on account of the explosive growth in their database size. Data mining tools predict future trends and behaviors, allowing businesses to make proactive, knowledge-driven decisions. Classification is taken into account clearly as the important tool in mining in an effort to turn great deal of \"passive data\" into useful \"actionable information\" [1]. It's a supervised machine learning technique which encompasses two steps, Model construction and Model usage [4]. Model construction describes a set of predetermined classes by means of decision tree. Model usage created for classifying unknown objects. The input data or training data is particular records. Each record involves values of many attributes. One of which is known as as Class Attribute and remaining are called as Predictor Attributes. Supervised learning will be achieved dependent on this class attribute. The values of sophistication attribute are named class labels. A predictor attribute may contain either continuous or discrete values (numerical attribute or categorical attribute). Classification is the problem of constructing a precise model (known as decision tree) [8] from the training dataset that encodes the distribution of class labels among different predictor attributes. The resulting model made with classification will be made use to classify new data whose class labels are unknown. Decision trees have proved to become valuable tools when it comes to the description, classification and generalization files [4]. Work towards constructing decision tree from data exists in multiple disciplines such as statistics, pattern recognition, decision theory, signal processing, machine learning and artificial neural networks [9]. Decision tree can be considered an acyclic graph and that is built by recursively partitioning the training data until all members of each partition belongs to same class label or there are actually you can forget attributes remaining for partitioning . Internal nodes

are referred as “Decision node” or “Test on node” that is attribute name. Leaf nodes are “Class node” that can be seen as corresponding class label. Each path of decision tree describes the choice rule. In order to encounter the mining of enormous datasets, many scalable algorithms have been proposed for constructing decision tree [1], [2], [3]. We only consider the well-known and an efficient means for constructing decision tree is C45 Framework. It bases toward the observation that, only aggregated information is mandatory to compute the splitting attribute on a node. Consequently, it proposes a relatively compact data structure whose sizes are proportional to the large number of different attribute values, not how many records. However, in an effort to get this aggregated information, needs to be matured pay an expensive price database scan conducted in every level of this very tree construction. proposed data structure has add-on information compared with the C45 framework[1][8].

2. LITERATURE SURVEY

CART methodology comprises three main stages. During the early stage a posh tree with maximum sizes are grown by recursively partitioning the data; this tree will have many terminal nodes. Despite the fact that the tree interprets data perfectly, when it over fits the comprehensive data the predictive ability becomes low. In the second stage, several nodes is shed to decrease the size of the tree, a procedure called pruning. Inside the final stage, predictive error is said to be as criterion to select optimal tree.

The process of growing the tree by CART is summarized as shown below:

1. Assign the data objects to some root node.

2. Select splitting criterion and explanatory variable that reduces impurity.

3. Split the root node into two child nodes by dividing the results objects in accordance with a splitting criterion and independent or explanatory variable selected seen from the groupping data objects.

4. Repeat all the above two steps considering each resulting node for being parent node so that the maximum size tree is obtained.

5. Prune the tree by eliminating several nodes using cross validation and price complexity [3].

Decision trees are popularly used to model unary interactions, e.g. but with two exceptions they have not been used for pairwise or higher-order interactions. The first exception is the paper of Glesner and Koller [1], where decision trees are used to model conditional probability tables over many discrete variables in a Bayesian network. The difference is that in [1] the decisions in the tree are evaluated on states of random variables, whereas in our work we evaluate the image content and thus require no change to the inference procedure.

A straight-forward way to deal with the uncertain information is to replace each pdf with its expected value, thus effectively converting the data tuples to point-valued tuples. This reduces the problem back to that for point-valued data, and hence traditional decision tree algorithms such as ID3. We call this approach AVG (for Averaging). We use an algorithm based on C4.5, using entropy as the dispersion measure. To alleviate the problem of overfitting, we apply the techniques of pre-pruning and postpruning[1].

Decision tree classification with missing data has been addressed for decades in the form of missing values[2], [3]. In C4.5[3], these are handled by using fractional tuples. In this work, we adopt the technique of fractional tuple for splitting tuples into subsets when the domain of its pdf spans across the split point. However, handling data values represented as pdf’s is unprecedented. In fuzzy decision tree classification, both attributes and class

labels can be fuzzy and are represented in fuzzy terms[8].

Given a fuzzy attribute of a data tuple, a degree (called

membership) is assigned to each possible value, showing the extent to which the data tuple belongs to a particular value. Our work instead gives classification results as a distribution: for each test tuple, we give a distribution telling how likely it belongs to each class[1,10].

SLIQ Algorithm

SLIQ (Supervised Learning in Quest, Mehta et al, 1996) was developed by the Quest team at IBM. Gini Index is used as a split measure. Gini index is minimized at each split, so that the tree becomes less diverse as we progress. The class histograms are built for each successive pairs of values of attributes. At any particular node, after obtaining all the

histograms for all attributes, the Gini index for each histogram is computed. Gini index for a sample histogram with two classes namely A and B is defined in Table 1. In Table 1, P denotes the splitting value for an attribute, a1and a2 denote the number of attributes which are less than P and belong to class A and B respectively. b1 and b2 denote the number of attributes which are greater than P and belong to class A and B respectively.

$$Gini\ Index = \frac{a1+a2}{n}\left[1-\left(\frac{a1}{a1+a2}\right)^2-\left(\frac{a2}{a1+a2}\right)^2\right]+ \frac{b1+b2}{n}\left[1-\left(\frac{b1}{b1+b2}\right)^2-\left(\frac{b2}{b1+b2}\right)^2\right]$$

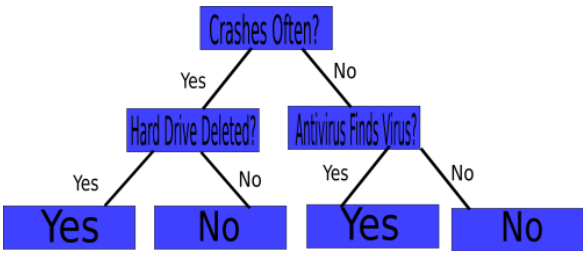
where n = Total number of records = a1+ a2 + b1+b2.

Once the Gini index for each histogram is known, the split attribute is chosen to be the one whose class histogram gives the least Gini index, and the split value equals the splitting point P for that histogram.

3. PROPOSED SYSTEM

Decision Trees:

A Decision Tree is a representation of how to make a decision according to a particular attribute set. Any given Decision Tree is completely deterministic, although some algorithms can alter their trees given additional knowledge. Each node of a decision tree is some attribute, with the branches representing alternative values of that attribute.To make a decision using a decision tree, select a set of values for an attribute and start at the root of the tree. Using the value of the root attribute, make a choice of which branch to take. Then continue making such choices at each node until a leaf node is reached. The decision in this leaf node is (hopefully) the best decision to make given the information available. Building Decision Trees is a form of supervised learning. A decision tree must use a heuristic to determine which attribute is the most informative attribute, since this attribute will be placed at the root of the tree. The most informative attribute' heuristic is taken from information theory, which is a method to calculate the information in a piece of data independent of its meaning. It seeks to calculate the information content of any piece of data by thinking of the answer as a series of bits. Therefore, a one bit answer encodes one bit of yes/no information. Decision trees are dependent on the order of the data, and it is possible for the optimality of the tree to change when changing the order of the data.



Decision Tree Induction

ID3, C4.5, and CART adopt a greedy (i.e., nonbacktracking) approach in which decision trees are constructed in a top-down recursive divide-and-conquer manner. Most algorithms for decision tree induction also follow such a top-down approach, which starts with a training set of tuples and their associated class labels. The training set I recursively partitioned into smaller subsets as the tree is being built. The leaf nodes of the decision tree contain the class name whereas a non-leaf node is a decision node. The decision node is an attribute test with each branch (to another decision tree) being a possible value of the attribute. ID3 uses information gain to help it decide which attribute goes into a decision node. ID3 is a non-incremental algorithm, meaning it derives its classes from a fixed set of training instances. An incremental algorithm revises the current concept definition, if necessary, with a new sample. The classes created by ID3 are inductive, that is, given a small set of training instances, the specific classes created by ID3 are expected to work for all future instances. The distribution of the unknowns must be the same as the test cases. Induction classes cannot be proven to work in every case since they may classify an infinite number of instances. Note that ID3 (or any inductive algorithm) may misclassify data.

1) ID3 ALGORITHM:

Algorithm: Generate decision tree. Generate a decision tree from the training tuples of data partition D.

Input: Data partition, D, which is a set of training tuples and their associated class labels; attribute list, the set of candidate attributes; Attribute selection method, a procedure to determine the splitting criterion that "best" partitions the data tuples into individual classes. This criterion consists of a splitting attribute and, possibly, either a split point or splitting subset.

Output: A decision tree.

Method:

(1) create a node N;

(2) if tuples in D are all of the same class, C then

(3) return N as a leaf node labeled with the class C;

(4) if attribute list is empty then

(5) return N as a leaf node labeled with the majority class in D; // majority voting

(6) apply Attribute selection method(D, attribute list) to find the "best" splitting criterion;

(7) label node N with splitting criterion;

(8) if splitting attribute is discrete-valued and multiway splits allowed then // not restricted to binary trees

(9) attribute list attribute list splitting attribute; // remove splitting attribute

(10) for each outcome j of splitting criterion

// partition the tuples and grow subtrees for each partition

(11) let Dj be the set of data tuples in D satisfying outcome j; // a partition

(12) if Dj is empty then

(13) attach a leaf labeled with the majority class in D to node N;

(14) else attach the node returned by Generate decision tree(Dj, attribute list) to node N; endfor

(15) return N;

Steps description:

1)The algorithm is called with three parameters: D, attribute list, and Attribute selection method.We refer to D as a data partition. Initially, it is the complete set of training tuples and their associated class labels. The parameter attribute list is a list of attributes describing the tuples. Attribute selection method specifies a heuristic procedure for selecting the attribute that "best" discriminates the given tuples according to class. This procedure employs an attribute selection measure, such as information gain or the gini index. Whether the tree is strictly binary is generally driven by the attribute selection measure. Some attribute selection measures, such as the gini index, enforce the resulting tree to be binary. Others, like information gain, do not, therein allowing multiway splits (i.e., two or more branches to be grown from a node).

2)The tree starts as a single node, N, representing the training tuples in D (step 1).

3) If the tuples in D are all of the same class, then node N becomes a leaf and is labeled with that class (steps 2 and 3). Note that steps 4 and 5 are terminating conditions. All of the terminating conditions are explained at the end of the algorithm.

4)Otherwise, the algorithm calls Attribute selection method to determine the splitting criterion. The splitting criterion tells us which attribute to test at node N by determining the "best" way to separate or partition the tuples in D into individual classes (step 6). The splitting criterion also tells us which branches to grow from node N with respect to the outcomes of the chosen test. More specifically, the splitting criterion indicates the splitting

attribute and may also indicate either a split-point or a splitting subset. The splitting criterion is determined so that, ideally, the resulting partitions at each branch are as "pure" as possible. A partition is pure if all of the tuples in it belong to the same class. In other words, if we were to split up the tuples in D according to the mutually exclusive outcomes of the splitting criterion, we hope for the resulting partitions to be as pure as possible.

5) The node N is labeled with the splitting criterion, which serves as a test at the node (step 7). A branch is grown from node N for each of the outcomes of the splitting criterion. The tuples in D are partitioned accordingly (steps 10 to 11).Let A be the splitting attribute. A has v distinct values, fa1, a2, : : : , avg, based on the training data.

Attribute Selection:

ID3 decide which attribute is the best using a statistical property, called information gain, is used. Gain measures how well a given attribute separates training examples into targeted classes. The one with the highest information (information being the most useful for classification) is selected. Given a collection S of c outcomes The expected information needed to classify a tuple in D is given by

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

where pi is the probability that an arbitrary tuple in D belongs to class Ci. A log function to the base 2 is used, because the information is encoded in bits. Info(D) is just the average amount of information needed to identify the class label of a tuple in D. Info(D) is also known as the entropy of D.

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

The term Dj /D acts as the weight of the jth partition. InfoA(D) is the expected information required to classify a tuple from D based on the partitioning by A. The smaller the expected information (still) required, the greater the purity of the partitions. Information gain is defined as the difference between the original

information requirement (i.e., based on just the proportion of classes) and the newrequirement .That is,

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

In other words, Gain(A) tells us how much would be gained by branching on A. It is the expected reduction in the information requirement caused by knowing the value of A. The attribute A with the highest information gain, (Gain(A)), is chosen as the splitting attribute at node N. This is equivalent to saying that we want to partition on the attribute A that would do the "best classification" so that the amount of information still required to finish classifying the tuples is minimal .

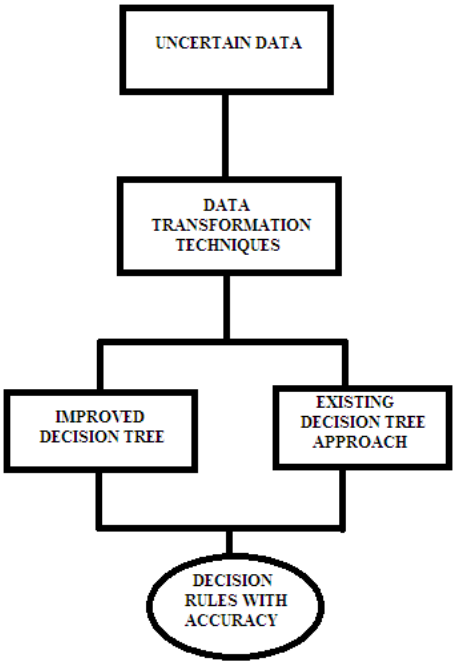
C4.5

C4.5 is the improved algorithm for ID3 follows same steps as in ID3 but the difference is using GainRatio for attribute selection measure for all the attributes.

IMPROVED C45 ALGORITHM

The C4.5 Algorithm is the extension of ID3 algorithm .It used a mechanism of learning from large datasets .The attribute selection of the algorithm is based on an assumption :the complexity of decision tree and the amount of information is represented by given attribute are closely related .C4.5 expands the classify range to digital attribute s.That metric standard of two-class entropy ,the most of the algorithm is based on the information entropy which is contained by produced nodal points of decision tree is least [9].The so called entropy is representative of degree of disorder of objects in the systematology. It is easy to understand that the smaller entropy the smaller disorder .In the other word the more sequential in the record collection, the more consistent .This is the target we seek ,too .Suppose the set S is a training sample ,the formula of entropy as follows:

Architecture:



C4.5 builds decision trees from a set of training data, using the concept of information entropy. The training data is a set S.s1,s2,s3 .... represents samples in the dataset S.Each si = xl,x2,... is a sample vector where xl,x2,... represents features or attributes of the sample. The training data associated with a vector C = cl,c2,... where cl,c2,...cn represents the class to which each sample belongs to dataset. At each node of the tree, C4.5 chooses one attribute of the data that most effectively splits its set of samples into subsets in one class or the other. Its criterion is the normalized information gain that results from choosing an attribute for splitting the data. The attribute with the highest information gain is chosen to make the decision. The C4.5 algorithm then recurs on the smaller sublists. This algorithm has a few base cases[5]. All the samples in the list belong to the same class. When this happens, it simply creates a leaf node for the decision tree same 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.

Instance of previously-unseen class encountered. Again, C4.5 creates a decision node higher up the tree using the expected value[10].

design the degree of balance coefficient of a certain

attribute as

Algorithm: Generate_decision_tree.

Narative : Generate a decision tree from the given training data.

Input: The training samples, samples, represented by discrete-valued attribute; the

set of candidate attributes, attribute-list.

Output: A decision tree.

Method:

(1) create a node N;

(2) **if** samples are all of the same class, C **then**

(3) return N as a leaf node labeled with the class C;

(4) **if** attribute-list is empty **then**

(5) return N as a leaf node labeled with the most common class in

samples;//majority voting

(6) select test-attribute, the attribute among attribute-list with the highest

information gain;

(7) label node N with test-attribute;

(8) **for each** known value ai of test-attribute;

(9) grow a branch from node N for the condition test-attribute = ai;

(10) let si be the set of samples in samples for which test-attribute = ai; // a

partition

(11) **if** si is empty **then**

(12) attach a leaf labeled with the most common class in samples;

(13) **else** attach the node returned by Generate_decision_tree (si, attribute-listtest-

attribute);

$$H\left(\frac{C}{V_\lambda}\right) = -\sum_j \left(p(v_j) + \lambda\right) \sum_i p(C_j / v_i) \log p(C_j / v_i) = Info_v(T_\lambda)$$

$$H(V_\lambda) = -\sum_i \left(p(v_i) + \lambda\right) \log p(v_i) = split\_Info(v_\lambda)$$

Modified Information or entropy is given as

ModInfo(D)= $-S_i \sum_{i=1}^{m} l \log \sqrt[3]{S_i}$,m different classes

ModInfo(D)= $-S_i \sum_{i=1}^{2} l \log \sqrt[3]{S_i}$

$$= -S_1 \log \sqrt[3]{S_1} + S_2 \log \sqrt[3]{S_2}$$

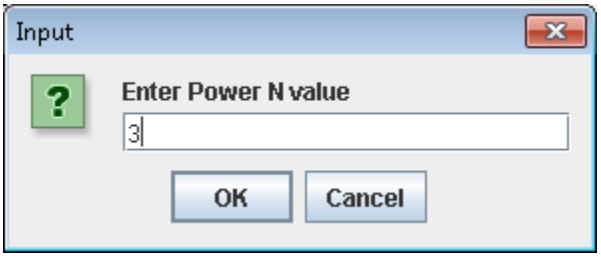
Where S_1 indicates set of samples which belongs to target class ‘anamoly’, S_2 indicates set of samples which belongs to target class ‘normal’.

Information or Entropy to each attribute is calculated using

$$Info_A(D) = \sum_{i=1}^{v} |D_i| / |D| \times ModInfo(D_i)$$

The term Di /D acts as the weight of the jth partition. ModInfo(D) is the expected information required to classify a tuple from D based on the partitioning by A.

4. RESULTS



input14 <= 50
| input16 <= 19
| | input11 <= 45
| | | input15 <= 46
| | | | input16 <= 1
| | | | | input6 <= 69
| | | | | | input2 <= 99: 0 (3.0)
| | | | | | input2 > 99: 4 (6.0)
| | | | | input6 > 69
| | | | | | input13 <= 27
| | | | | | | input3 <= 11
| | | | | | | | input1 <= 26: 7 (9.0)
| | | | | | | | input1 > 26: 1 (3.0)
| | | | | | | input3 > 11
| | | | | | | | input15 <= 16: 1 (141.0/1.0)
| | | | | | | | input15 > 16
| | | | | | | | | input4 <= 95: 1 (11.0)
| | | | | | | | | input4 > 95: 7 (4.0/1.0)
| | | | | | input13 > 27
| | | | | | | input1 <= 23
| | | | | | | | input4 <= 87: 1 (2.0)
| | | | | | | | input4 > 87: 7 (8.0/1.0)
| | | | | | | input1 > 23: 5 (8.0/1.0)
| | | | input16 > 1: 6 (24.0)
| | | input15 > 46
| input15 > 92
| | input14 <= 78
| | | input6 <= 40
| | | | input2 <= 45: 9 (3.0)
| | | | input2 > 45: 5 (6.0)
| | | input6 > 40
| | | | input1 <= 44: 7 (36.0/1.0)
| | | | input1 > 44: 8 (157.0)
| | input14 > 78: 5 (600.0/1.0)

Number of Leaves : 197

Size of the tree : 393

Time taken to build model: 45.46 seconds

Time taken to test model on training data: 0.21 seconds

=== CLASSIFICATION ACCURACY DETAILS ===

Correctly Classified Accuracy %	10916	99.7386
Incorrectly Classified Accuracy	76	0.2614 %

=== Confusion Matrix ===

a	b	c	d	e	f	g	h	i	j	<-- classified as
1128	1	0	0	3	0	3	1	6	1	a = 0
0	1075	38	12	2	1	0	7	0	8	b = 1
0	35	1098	3	0	0	2	6	0	0	c = 2
0	8	3	1018	1	14	0	2	3	6	d = 3
3	5	0	1	1124	2	1	2	0	6	e = 4
0	5	1	13	1	1018	0	3	4	10	f = 5
9	1	1	1	4	5	1029	1	5	0	g = 6

0 15 7 3 2 4 3 1100 7 1 | h = 7

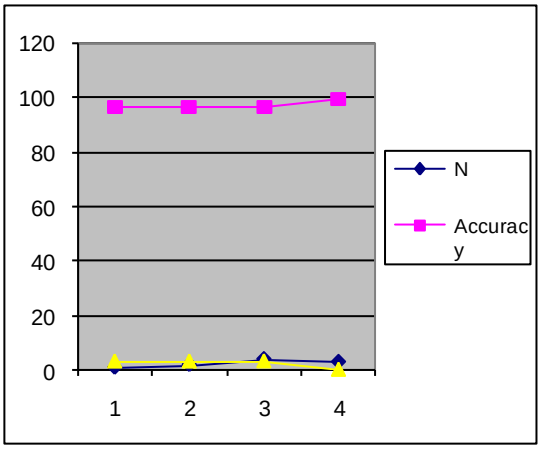
9 1 1 1 0 5 9 10 1017 2 | i = 8

2 7 0 4 6 16 0 1 3 1016 | j = 9

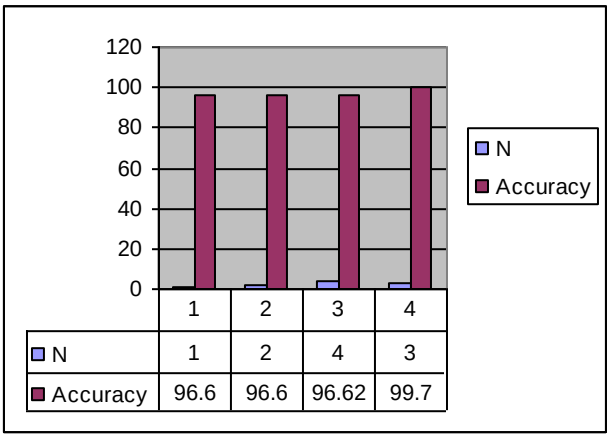
Table1

N	No of Leaves	Size of tree	Accuracy	Error
1	198	395	96.6	3.3
2	197	393	96.6	3.3
4	197	393	96.62	3.37
3	197	393	99.7	0.3

Above table displays various decision tree factors for the N input values.



Graph shows the relationship between the N values, Accuracy and Error.



Graph shows the relationship between the N values and Accuracy.

5. CONCLUSION AND FUTURE SCOPE

We have proposed an efficient scalable Improved decision tree construction algorithm which results in high processing speed and small scale. Because of this reason, it is most suitable for large datasets. Our proposed algorithm has many advantages, but the important thing is that it requires only one pass over the training dataset for the entire construction of decision tree. So it significantly reduces the IO cost. Moreover, our algorithm provides a general framework that can be used with any existing decision tree construction algorithms and requires only one time sorting for the numerical attribute. Hence, it reduces the sorting cost of numerical attributes and execution time of partitioning phase in the decision tree construction process. From the experimental evaluation, we have got a promising result, since our proposed algorithm outperforms the Existing C45 algorithm in execution time.

REFERENCES

[1] Decision Trees for Uncertain Data Smith Tsang..

[2] Yang Xue-bing,Zhang Jun, ”Decision Tree Algorithm and its core technology,” Computer Technology and Development,2007, 17(1),pp.43-45

[3] Qu Kai-she,Wen Cheng-li,Wang Jun-hong, ”An improved algorithm of ID3 algorithm,” Computer Engineering and Applications, 2003,(25),pp.104-107

[4] Mao Cong-li Yi Bo, ”The most simple decision tree generation algorithm based on decision-making degree of coordination ,“ Computer Engineering and Design,2008,29(5),pp.1250-1252

[5] Huang Ai-hui, ”Improvement and application of decision tree C4.5 algorithm ,“ Science Technology and Engineering,2009, (1),pp.34-37

[6] J. Gehrke, R. Ramakrishnan, and V. Ganti, "Rainforest, a framework for fast decision tree construction of large datasets", in Springer Netherlands-Data mining and knowledge discovery vol.4. Issue(2-3) July 2000.

[7] M. Kantardzic “Data Mining. Concepts, Models, Methods and Algoritms”. John Wiley and Sons Inc, 2003.

[8] Xu.M.Wang, J. and Chen.T. “Improved decision tree algorithm: ID3+” Intelligent Computing in Signal Processing and Pattern Recognition, Vol.345, pp.141-149, 2006.

[9] Quinlan, J. R. “C4.5: Programs for Machine Learning” Morgan Kaufmann, San Mateo, CA 1993.