An Efficient Technique to protect unrealized data over classification

[1]V.Sita Rama Prasad, [2]Dr L PRASANNA KUMAR

[1] **M.tech II Year(Pursuing) ,** DADI INSTITUTE OF ENGINEERING & TECHNOLOGY
[2]ASSOCIATE PROFESSOR,DEPT OF CSE,DADI INSTITUTE OF ENGINEERING & TECHNOLOGY

ABSTRACT:

Data-mining helps to extract hidden predictive information from large databases. There are several tech-techniques and algorithms used for extracting the hidden patterns from the large data sets and finding the relationships between them Privacy preservation is an important factor in data mining. The problem of privacy preservation in data mining has become more important in recent years because of increasing need to store vast data about users. In this research work, a new privacy preserving approach is applied to decision tree learning. This approach converts the original sample datasets into a group of unreal datasets. The original sample datasets cannot be reconstructed from it. Meanwhile, an accurate decision tree is built using those unreal datasets. C4.5 algorithm is used to build decision tree. The experimen-tal results show that accurate and efficient decision tree is built in C4.5 algorithm than existing algorithm

1. INTRODUCTION

Data mining is the non-trivial process of identifying valid, novel, potentially useful, and ultimately understandable patterns in data [1]. In data mining and machine learning, preserving of privacy is important process. Privacy-preserving processes have been developed to sanitize private information from the samples while keeping their utility.

The problem of privacy-preserving data mining has become more important in recent years, because of the increasing ability to store personal data about users. Many privacy protection approaches preserve private information of sample datasets, but not precision of data mining outcomes. Hence, the utility of the sanitized datasets is downgraded.

This paper provides an approach that preserves privacy and utility of sample datasets for decision-tree data mining. In data collection processes, a sufficiently large number of sam-ple data sets have been collected to achieve significant data mining results covering the whole research target.

This approach converts original samples into a group of unreal datasets from which the original samples cannot be reconstructed without the entire group of unreal data sets. Meanwhile, an accurate decision tree can be built directly from those unreal data sets.

This paper is organized as follows: Section 2 explains a brief discussion about the decision tree learning. Section 3 provides discussion on the previous works related to the topic. Section 4 describes the existing approaches of decision tree learning and the proposed algorithm for decision tree learning. Section 5 involves the Conclusion and future works.

2. DECISION TREE LEARNING

Decision tree learning is one of the most widely used and practical methods for inductive inference. Decision tree learning is a method for approximating discrete-valued target functions, in which the learned function is represented by a decision tree. Data comes in records of the form:

$(X, Y) = (x_1, x_2, x_3, \ldots\ldots, x_k, Y)$

The dependent variable, Y, is the target variable that we are trying to understand, classify or generalize. The vector x is composed of the input variables, x_1, x_2, x_3 etc., that are used for that task.

Decision trees used in data mining are of two main types:

- Classification tree analysis is when the predicted outcome is the class to which the data belongs.
- Regression tree analysis is when the predicted outcome can be considered a real number.

DECISION TREE LEARNING ALGORITHM

Most Algorithms that have been developed for learning decision trees are variations on a core algorithm that employs a top-down, greedy search through the space of possible decision trees. This approach is exemplified by the ID3 algorithm (Quinlan 1986) and it's Successor C4.5 (Quinlan 1993).

ID3 ALGORITHM

ID3 is a nonincremental algorithm, meaning it derives its classes from a fixed set of training instances. The classes cre-ated 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.

C4.5 ALGORITHM

C4.5 builds decision trees from a set of training data in the same way as ID3, using the concept of information entro-py.

The training data is a set $S=s_1, s_2...$ of already classified samples. Each sample s_i consists of a p-dimensional vector, $(x_{1,i}, x_{2,i}, \ldots, x_{p,i})$ where the x_j represent attributes or features of the sample, as well as the class in which s_i falls.

The general algorithm for building decision tree is:

1. Check for base cases
2. For each attribute a
1. Find the normalized information gain from splitting on a
3. Let a' be the attribute with the highest normalized information gain
4. Create a decision node that splits on a'
5. Recurse on the sublists obtained by splitting on a', and add those nodes as children of node

At each node of the tree, C4.5 chooses the attribute of the data that most effectively splits its set of samples into subsets enriched in one class or the other. The splitting criterion is the normalized information gain .The attribute with the highest normalized information gain is chosen to make the decision. The C4.5 algorithm then recurses on the smaller sublists.

ID3 ALGORITHM

ID3 (Learning Sets S, Attributes Sets A, Attributes values V) Return Decision Tree.

Begin

Load learning sets first, create decision tree root node 'rootNode', add learning set S into root node as its subset.

For rootNode, we compute Entropy(rootNode.subset) first

If Entropy(rootNode.subset)==0, then rootNode.subset consists of records all with the same value for the categorical attribute, return a leaf node with decision attribute:attribute value;

If Entropy(rootNode.subset)!=0, then compute information gain for each attribute left(have not been used in splitting), find attribute A with

Maximum(Gain(S,A)). Create child nodes of this rootNode and add to rootNode in the decision tree.

For each child of the rootNode, apply ID3(S,A,V) recursively until reach node that has entropy=0 or reach leaf node.

End ID3

The principle of the ID3 algorithm is as follows. The tree is constructed top-down in a recursive fashion. At the root, each attribute is tested to determine how well it alone classifies the transactions. The "best" attribute (to be discussed below) is then chosen and the remaining transactions are partitioned by it. ID3

is then recursively called on each partition (which is a smaller database containing only the appropriate transactions and without the splitting attribute).

ID3(R; C; T)

1. If R is empty, return a leaf-node with the class value assigned to the most transactions in T.

2. If T consists of transactions which all have the same value c for the class attribute, return a leaf-node

with the value c (finished classification path). 3. Otherwise,

(a) Determine the attribute that best classifies the transactions in T, let it be A.

(b) Let a1; :::; am be the values of attribute A and let T(a1); :::; T(am) be a partition of T such that

every transaction in T(ai) has the attribute value ai

.

(c) Return a tree whose root is labeled A (this is the test attribute) and has edges labeled a1; :::; am

such that for every i, the edge ai goes to the tree ID3(R ¡ fAg; C; T(ai)).

What remains is to explain how the best predicting attribute is chosen. This is the central principle of ID3 and is based on information theory. The entropy of the class attribute clearly expresses the difficulty of prediction. We know the class of a set of transactions when the class entropy for them equals zero. The idea is therefore to check which attribute reduces the information of the class-attribute to the greatest degree. This results in a greedy algorithm which searches for a small decision tree consistent with the database. The bias favoring short descriptions of a hypothesis is based on Occam's razor. As a result of

this, decision trees are usually relatively small, even for large databases.2 The exact test for determining the best attribute is defined as follows. Let c1; :::; c` be the classattribute values and let T(ci) denote the set of transactions with class ci . Then the information needed

to identify the class of a transaction in T is the entropy, given by

Let T be a set of transactions, C the class attribute and A some non-class attribute. We wish to quantify the information needed to identify the class of a transaction in T given that the value of A has been obtained. Let A obtain values a1; :::; am and let T(aj) be the transactions obtaining value aj for A. Then, the conditional information of T given A, equals:

Extensions of ID3. Since its inception there have been many extensions to the original algorithm, the most well-known being C4.5. We now briefly describe some of these extensions. One of the immediate shortcomings of ID3 is that it only works on discrete data, whereas most databases contain continuous data. A number of methods enable the incorporation of continuous-value attributes, even as the class attribute. Other extensions include handling missing attribute values, alternative measures for selecting attributes and reducing the problems of overfitting by pruning. (The strategy described in footnote 2 also addresses the problem of overfitting.)

The ID3 algorithm chooses the "best" predicting attribute by comparing entropies that are given as real numbers. If at a given point, two entropies are very close together, then the two (different) trees resulting from choosing one attribute or the other are expected to have almost the same predicting

capability. Formally stated, let δ be some small

value. Then, for a pair of attributes $A1$ and $A2$, we

say that $A1$ and $A2$ have *δ-close information gains*

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

3. RELATED WORKS

A wide research has been devoted to the protection of sensitive information when samples are given to third parties for processing or computing [2], [3], [4], [5], [6]. Samples may be leaked or stolen anytime during the storing process or while residing in storage. This paper focuses on preventing such attacks to the samples by third parties.

Contemporary research in privacy preserving data mining mainly falls into one of two categories: 1) perturbation and randomization-based approaches, and 2) secure multiparty computation (SMC)-based approaches [7].SMC approaches employ cryptographic tools for collaborative data mining computation by multiple parties. Samples are distributed among different parties and they take part in the information computation and communication process. SMC research focuses on protocol development [8] for protecting privacy among the involved parties [9] or computation efficiency [10]; however, centralized processing of samples and storage privacy is out of the scope of SMC.

This approach is designed to preserve both the privacy and the utility of the sample data sets used for decision tree data mining. This method applies a series of encrypting functions to sanitize the samples and decrypts them correspondingly for building the decision tree. In addition to protecting the input data of the data mining process, this approach also protects the output data, i.e., the generated decision tree.

4. PROBLEM DEFINITION & PROPOSED METHODOLOGY

In this paper we have proposed a privacy preserving approach that can be applied to decision tree learning. This approach converts the original sample datasets into a group of unreal datasets. The original sample datasets cannot be reconstructed from it. Meanwhile, an accurate decision tree is built, using C4.5 algorithm, from those unreal datasets.

- Unrealized dataset conversion
- Decision Tree Generation
- Distribution
- Comparison

UNREALIZED DATASET CONVERSION:

For conversion of unrealized dataset, we use the algorithm of unrealized training set. Data modification techniques main-tain privacy by modifying attribute values of the sample data sets. For this process K-anonymity approach is used for the modification purpose. In this process datasets are inserted into the data table. Data unrealization algorithm is used for this process. Inserted dataset are unreal dataset.

First we load the universal set and the sample set. This sample set and universal set is implemented by the unrealized training set algorithm. Finally the output of the unrealized data set is training set and perturbation set. T^U, the universal set of data table T, is a set containing all possible datasets in data table T. Let T associates with attributes <Wind, Play> where Wind= {Strong, Weak} and Play= {Yes, No} then T^U = {<Strong, Yes>, <Strong, No>, <Weak, Yes>, <Weak, No>}. T_S is constructed by inserting sample data sets into a data table. T^p is a perturbing set that generates unreal datasets which is used for convert-ing T_S into unrealized training set T'.

Algorithm Unrealize-Training-Set (T_S,T^U, T', T^P)

Input: T_S , a set of input sample data sets

T^U , a universal set

T' , a set of output training data sets

T^P , a perturbing set

Output: T^P , T

1. if TS is empty then return(T', T^p)
2. t ← a data set in T_S
3. if t is not an element of T^P or T^P = t then
4. $T^P \leftarrow T^P + T^U$
5. $T^P \leftarrow T^P - \{t\}$
6. t' ←the most frequent dataset in T
7. return Unrealize-TrainingSet

T_S-{t},T^u ,T'+{t'},T^p -(t'})

SYSTEM DESIGN

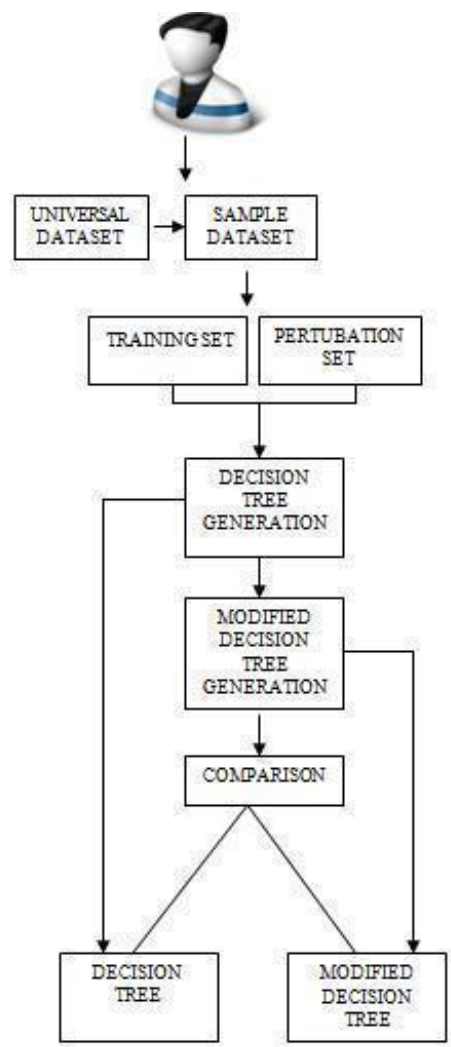


Figure 1: System Architecture of Proposed Methodology

DECISION TREE GENERATION:

The well-known ID3 algorithm builds a decision tree by call-

ing algorithm Choose-Attribute recursively. This algorithm selects a test attribute according to the information content of the training set T_S.

Algorithm Generate-Tree (T_S , attribs , default)

Input: T_S, the set of training data sets

attribs, set of attributes default,

default value for the goal predicate

Output: tree, a decision tree

1. if T_S is empty then return default
2. default ←Majority _ Value(T_S)
3. if $H_{ai}(T_S) = 0$ then return default
4. else if attribs is empty then return default
5. else
6. best ←Choose-Attribute(attribs, T_S)
7. tree← a new decision tree with root attribute best
8. for each value v_i of best do
9. T_{Si}← {datasets in T_S as best = k_i}
10. subtree ←Generate-Tree(T_{Si}, attribs-best, default}
11. connect tree and subtree with a branch labelled k_i
12. return tree

C4.5 algorithm is used for making decision tree process. A decision tree is done by calling algorithm Choose-Attribute recursively. There are 2 types of tree is generated in this pro-cess, first decision tree using the majority values in the gain calculation and modified decision tree using minority values in the gain calculation.

Algorithm Generate-Tree' (size, T', T^P , attribs, default) Input: size, size of qT^U

T', the set of unreal training data sets

T^P , the set of perturbing data sets

attribs, set of attributes default,

default value for the goal predicate

Output: tree, a decision tree

1. if (T'; T^P) is empty then return default
2. default ← Minority _ Value(T', T^p)
3. if H_{ai} ($q[T'+T^{pc}]$)=0 then return default
4. else if attribs is empty then return default
5. else
6. best ← Choose-Attribute'(attribs, size, (T', T^P))
7. tree ← a new decision tree with root attribute best
8. size ← size/number of possible values k_i in best
9. for each value v_i of best do
10. T'$_i$← {data sets in T' as best =k_i}
11. T^P_i←{data sets in T^p as best =k_i}
12. subtree← Generate-Tree(size, T'$_i$, T^P_i, attribs-best, default)
13. connect tree and subtree with a branch labelled k_i
14. return tree

Decision Tree Generation

The well-known ID3 algorithm [18] shown above builds a decision tree by calling algorithm *Choose-Attribute* recursively. This algorithm selects a test attribute (with the smallest entropy) according to the information content of the training set TS. The information entropy functions are given as

$$H_{a_i}(T_S) = -\sum_{e \in K_i} \left(\frac{\left|T_{S_{(a_i=e)}}\right|}{|T_S|} \right) \log_2 \left(\frac{\left|T_{S_{(a_i=e)}}\right|}{|T_S|} \right)$$

$$H_{a_i}(T_S/a_j) = \sum_{f \in K_j} \left(\frac{\left|T_{S_{(a_j=f)}}\right|}{|T_S|} \right) H_{a_i}\left(T_{S_{(a_j=f)}}\right)$$

Where K_i and K_j are the sets of possible values for the decision attribute, a_i, and test attribute, a_j, in TS, respectively, and the algorithm Majority-Value retrieves the most frequent value of the decision attribute of T_S

An algorithm that generates an unrealized training set, T', and a perturbing set, T^P, from the samples in T_S. In this section, we use data tables T' and T^P as a means to calculate the information content and information gain of TS, such that a decision tree of the original data sets can be generated based on T' and T^P

5 Information Entropy Determination

From the algorithm Unrealized-Training-Set, it is obvious that the size of T_S is the same as the size of T'. Furthermore, all data sets in (T' + T^P) are based on the data sets in T^U, excepting the ones in T_S, i.e., T_S is the q-absolute complement of (T' + T^P) for some positive integer q. The size of qT^U can be computed from the sizes of T' and T^P, with qT^U = 2* |T'|+|T^P |. Therefore, entropies of the original data sets, T_S, with any decision attribute and any test attribute, can be determined by the unreal training set, T', and perturbing set, T^P .

6. Modified Decision Tree Generation Algorithm

As entropies of the original data sets, T_S, can be determined by the retrievable information—the contents of unrealized training set, T', and perturbing set, T^P —the decision tree of T_S can be generated by the following algorithm.

Similar to the traditional ID3 approach, algorithm Choose-Attribute' selects the test attribute using the ID3 criteria, based on the information entropies, i.e., selecting the attribute with the greatest information gain. Algorithm Minority-Value retrieves the least frequent value of the decision attribute of (T' + T^P), which performs the same function as algorithm Majority- Value of the tradition ID3 approach, that is, receiving the most frequent

International Journal of Advanced and Innovative Research (2278-7844) / #324 / Volume 2 Issue 9

To generate the decision tree with T', T^P and $|qT^U|$ (which equals 2* |T'|+|T^P|), a possible value, kd, of the decision attribute, ad, (which is an element of A—the set of attributes in T) should be arbitrarily chosen, i.e., we call the algorithm *Generate -Tree (*2* |T'|+|T^P|, T_S, T^U, A - ad, kd)*. The resulting decision tree of our new ID3 algorithm with unrealized sample inputs is the same as the tree generated by the traditional ID3 algorithm with the original samples

7. Data Set Reconstruction

Section B introduced a modified decision tree learning algorithm by using the unrealized training set, T', and the perturbing set, T^P. Alternatively, we could have reconstructed the original sample data sets, T_S, from T' and T^P, followed by an application of the conventional ID3 algorithm for generating the decision tree from T. The reconstruction process is dependent upon the full information of T' and T^P (whereas q =2* |T'|+|T^P|/|T^U|); reconstruction of parts of T_S based on parts T' and T^P is not possible.

8. Enhanced Protection with Dummy Values

Dummy values can be added for any attribute such that the domain of the perturbed sample data sets will be expanded while the addition of dummy values will have no impact on T_S. Dummy represents a dummy attribute value that plays no role in the data collection process. In this way we can keep the same resulting decision tree (because the entropy of T_S does not change) while arbitrarily expanding the size of T^U. Meanwhile, all data sets in T' and T^P, including the ones with a dummy attribute value, are needed for determining the entropies of $(T' + T^P)$ during the decision tree generation process.

8.1 C5.0 algorithm

In the proposed algorithm, consider C5.0 Algorithm for data mining. The enhancement and the optimization of the C4.5 emerge as algorithm C5.0, which exhibits the better performance as compared to the other existing mining algorithms. C5.0 algorithm to build either a decision tree or a rule set. A C5.0 model works by splitting the sample based on the field that provides the maximum information gain. Each sub sample defined by the first split is then split again, usually based on a different field, and the process repeats until the sub samples cannot be split any further. Finally, the lowest-level splits are re-examined, and those that do not contribute significantly to the value of the model are removed or pruned. C5.0 can produce two kinds of models. A decision tree is a straightforward description of the splits found by the algorithm. Each terminal (or "leaf") node describes a particular subset of the training data, and each case in the training data belongs to exactly one terminal node in the tree.

In contrast, a rule set is a set of rules that tries to make predictions for individual records. Rule sets are derived from decision trees and, in a way, represent a simplified or distilled version of the information found in the decision tree. Rule sets can often retain most of the important information from a full decision tree but with a less complex model. Because of the way rule sets work, they do not have the same properties as decision trees. The most important difference is that with a rule set, more than one rule may apply for any particular record, or no rules at all may apply. If multiple rules apply, each rule gets a weighted "vote" based on the confidence associated with that rule, and the final prediction is decided by combining the weighted votes of all of the rules that apply to the record in question. If no rule applies, a default prediction is assigned to the record. It was introduced an alternative formalism consisting of a list of rules of the form ―if A and B and C and ... then class X‖, where rules for each class are grouped together. A case is classified by finding the first rule whose conditions are satisfied by the case; if no rule is satisfied, the case is assigned to a default class. Each case belongs to one of a small number of mutually exclusive classes. Properties of every case that may be relevant to its class are provided, although some cases may have unknown or non-applicable values for some attributes. C5.0 can deal with any number of attributes. Rule sets are generally easier to understand than trees since each rule describes a specific context associated with a class. Furthermore, a rule set generated from a tree usually has fewer rules than the tree has leaves, another plus for comprehensibility. Another advantage of rule set classifiers is that they are often more accurate predictors than decision trees.

C5.0 decision tree is constructed using *GainRatio*. *GainRatio* is a measure incorporating entropy. Entropy (*E(S),*) measures how unordered the data set is. It is denoted by the following equation when there are classes $C_1 \ldots C_N$ in data set *S* where *P (Sc)* is the probability of class *C* occurring in the data set *S*:

$$E(S) = -\sum_{c=1}^{N} P(S_c) * \log_2 P(S_c)$$

Information Gain is a measure of the improvement in the amount of order.

$$Gain(S,V) = E(S) - \sum_{V \in Values(V)} (S_v|S) * E(S_v)$$

Gain has a bias towards variables with many values that partition the data set into smaller ordered sets. In order to reduce this bias, the entropy of each variable over its *m* variable values is calculated as *SplitInfo: GainRatio* is calculated by dividing *Gain* by *SplitInfo* so that the bias towards variables with large value sets is dampened

$$Gain(S,V) = \frac{Gain(S,V)}{SplitInfo(S,V)}$$

C5.0 builds a decision tree greedily by splitting the data on the variable that maximizes gain ratio. A final decision tree is changed to a set of rules by converting the paths into conjunctive rules and pruning them to improve classification accuracy.

DISTRIBUTION:

Here we performs the calculation using the even distribution, extremely uneven distribution, and normal distribution. And also we perform the process of accuracy and accuracy calcu-lation. Dummy values can be added for any attribute such that the domain of the perturbed sample data sets will be expanded while the addition of dummy values will have no impact on Training Set.

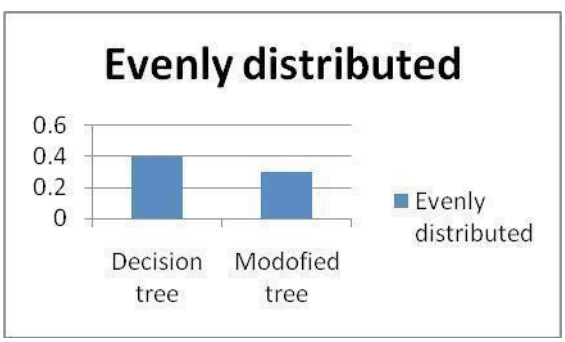
COMPARISON GRAPH:

We generate the accuracy graph and the time complexity graph.

:

The storage requirement increases while the required stor-age may be doubled if dummy attribute values technique is applied to double the sample domain. The best case happen when samples are evenly distributed.

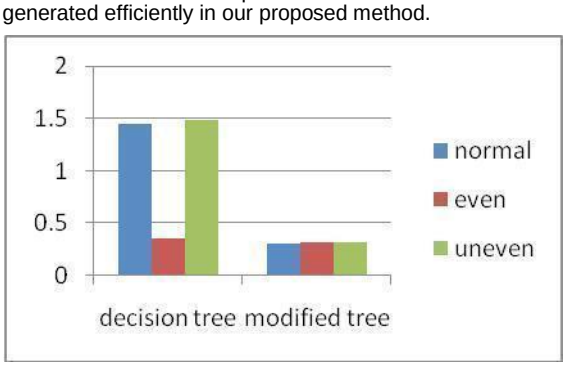
Samples with even distribution are taken. In even distribution, all datasets have the same counts. Decision tree is generated with increased storage required in existing method while it is not with our proposed method.



Time complexity Graph:

The worst case happens when the samples are in uneven distribution. Based on the randomly picked tests, Time complexity of storage for our approach is less than five times (without using dummy values) and eight times (with dummy values) than that of the original samples.

Normally distributed, evenly distributed and extremely uneven distributed samples are taken. Decision tree is generated efficiently in our proposed method.



5. CONCLUSION

We introduced a new privacy preserving approach that converts the sample data sets, training set, into some unreal data sets, such that any original data set is not able to reconstruct, if an unauthorized party where to steal some portion of data set. Privacy preservation via data set complementation fails if all training data sets are leaked because the data set reconstruction algorithm is generic. Therefore, further research is required to overcome this preserving approach with the C4.5 decision tree learning algorithm and discrete-valued at-tributes only.

1. S. Ajmani, R. Morris, and B. Liskov, ―A Trusted Third-Party Computation Service,‖ Technical Report MIT-LCS-TR-847, MIT, 2001.
2. S.L. Wang and A. Jafari, ―Hiding Sensitive Predictive Association Rules,‖ Proc. IEEE Int'l Conf. Systems, Man and Cybernetics, pp. 164-169, 2005.
3. R. Agrawal and R. Srikant, ―Privacy Preserving Data Mining,‖Proc. ACM SIGMOD Conf. Management of Data (SIGMOD '00),pp. 439-450, May 2000.
4. Q. Ma and P. Deng, ―Secure Multi-Party Protocols for Privacy Preserving Data Mining,‖ Proc. Third Int'l Conf. Wireless Algorithms, Systems, and Applications (WASA '08), pp. 526-537, 2008.
5. J. Gitanjali, J. Indumathi, N.C. Iyengar, and N. Sriman, ―A Pristine Clean Cabalistic Fortuity Strategize Based Approach for Incremental Data Stream Privacy Preserving Data Mining,‖ Proc. IEEE Second Int'l Advance Computing Conf. (IACC), pp. 410-415, 2010.
6. N. Lomas, ―Data on 84,000 United Kingdom Prisoners is Lost,‖ Retrieved Sept. 12, 2008, http://news.cnet.com/8301-1009_3-10024550-83.html, Aug. 2008.
7. BBC News Brown Apologises for Records Loss. Retrieved Sept.12, 2008, http://news.bbc.co.uk/2/hi/uk_news/politics/7104945.stm, Nov. 2007.
8. D. Kaplan, Hackers Steal 22,000 Social Security Numbers from Univ.of Missouri Database, Retrieved Sept. 2008, http://www.scmagazineus.com/Hackers-steal-22000-Social-Security-numbers-from-Univ.-of-Missouri-database/article/34964/, May 2007.
9. D. Goodin, ―Hackers Infiltrate TD Ameritrade client Database,‖ Retrieved Sept. 2008, http://www.channelregister.co.uk/2007/09/15/ameritrade_database_burgled/, Sept. 2007.
10. L. Liu, M. Kantarcioglu, and B. Thuraisingham, ―Privacy Preserving Decision Tree Mining from Perturbed Data,‖ Proc. 42nd Hawaii Int'l Conf. System Sciences (HICSS '09), 2009.
11. Y. Zhu, L. Huang, W. Yang, D. Li, Y. Luo, and F. Dong, ―Three New Approaches to Privacy-Preserving Add to Multiply Protocol and Its Application,‖ Proc. Second Int'l Workshop Knowledge Discovery and Data Mining, (WKDD '09), pp. 554-558, 2009.
12. J. Vaidya and C. Clifton, ―Privacy Preserving Association Rule Mining in Vertically Partitioned Data,‖ Proc Eighth ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining (KDD '02), pp. 23-26, July 2002.
13. M. Shaneck and Y. Kim, ―Efficient Cryptographic Primitives for Private Data Mining,‖ Proc. 43rd Hawaii Int'l Conf. System

Author:

Dr L PRASANNA KUMAR
ASSOCIATE PROFESSOR,
DEPT OF CSE,
DADI INSTITUTE OF
ENGINEERING & TECHNOLOGY

V.Sita Rama Prasad II-
Mtech(PURSUING)
DADI INSTITUTE OF
ENGINEERING & TECHNOLOGY
ANAKAPALLE,VISAKHAPATNAM
DISTRICT,ANDHRAPRADESH