Automatic mining of Threatening e-mail using Ad Infinitum algorithm

Appavu alias Balamurugan[1], Rajaram [2], Muthupandian [1] and Athiappan[1]

[1]Department of Information Technology,
Thiagarajar College of Engineering,
Madurai-625 015, India.

[2]Department of Computer Science and Engineering,
Thiagarajar College of Engineering,
Madurai- 625 015, India.

E-mail :{ sbit, rrajaram@tce.edu}

Abstract

We have proposed a new classifier named Ad Infinitum to identify e-mails containing terrorist threats. This Ad Infinitum is an enhancement of decision tree induction algorithm which will generate reliable rules to discriminate emails from potentially dangerous to those that are safe, with the help of agent. In the preprocessing stage, we have used deception theory which states that deceptive writing is characterized by reduced frequency of first person pronouns, exclusive words, elevated frequency of negative emotion words and action verbs. We have demonstrated the effectiveness of the proposed approach on a real world email corpus.

Keywords: Data mining, Threatening e-mail detection, Decision Tree, Ad Infinitum algorithm, Classification, Concept mining.

1. Introduction

The tragedy of September 11 is immeasurable and it has caused a permanent effect in United States and rest of the World. In order to avoid such disaster in future, an effective security system need to be established. Moreover it was identified that the terrorists have used the e-mail as the medium for transferring information among them. In order to prevent such disaster in future there is a need to identify the possible meaning of the information which is exchanged among the terrorists.

Data mining has been the science of extracting useful information from large data sets or databases. Data mining has recently become one of the most attractive message analyzing tools, also it creates considerable attention towards database practitioners and researchers because of its applicability in many areas such as decision support, market strategy, financial forecasts, etc. Combining techniques from the fields like statistics, machine learning, databases, etc, data mining helps in extracting useful and invaluable information from database and it is a powerful tool that enables criminal investigators who may lack extensive training as data analyst to explore large database quickly and efficiently.

E-mail has become one of today's standard means of communication. E-mail data is also growing rapidly, creating needs for automated analysis. So to detect crime, a spectrum of techniques should be applied to discover and identify patterns and make predictions. Data mining has emerged to address problems of understanding ever-growing volumes of information for structured data, finding patterns within data that are used to discover useful knowledge.

As individuals increase their usage of electronic communication, there has been research in the area of detecting deception in these new forms of communication. Models of deception assume that deception leaves a footprint. Work done by various researchers suggests that deceptive writing is characterized by reduced frequency of first-person pronouns, exclusive words, elevated frequency of negative emotion words and action verbs. We apply this model of deception to the e-mail dataset and preprocess the e-mail body. To train the system we used Ad Infinitum classifier that categorizes the email as either threatening or normal.

Classification is a primary data mining task aimed at learning a function that classifies a database record into one of the several predefined classes (e.g. classification of e-mails into normal versus threatening) based on the value of the instances. Common classification algorithms like Back Propagation, Naive Bayes, Decision Tree and Support Vector Machine are designed to optimize the predictive performance of the Induced model. Other aspects of knowledge discovery such as identification of relevant features and inconsistent data are given only secondary consideration by most existing algorithms. Consequently classification models induced from real world data does not deal with inconsistent data and are statistically insignificant. The Ad Infinitum algorithm presented in this paper is aimed at solving these problems.

In this paper we have introduced a novel approach for building simple and reasonably accurate classifier termed Ad Infinitum for classification and detection of threatening e-mails from the given user Inbox. We have categorized the e-mail in to two types such as threatening and normal. Threatening e-mail provides information about the future criminal activities which creates serious consequences in future. And these types of malicious e-mails (i.e. e-mail- related to terrorism, fraud, etc.) can be identified through our proposed system, by which the security enforcing methods can be strengthened. Also we can prevent the occurrences of future attacks.

1.1 Motivation

The importance of National security has increased significantly due to the sequential bomb blast, hijack of planes and gets intensified since the terrorist attack on 11 September 2001 on the world trade centre which killed more than 3000 innocent people. The Central Intelligence Agency (CIA), Federal Bureau of Investigation (FBI) and other countries national security agencies are actively collecting key information from domestic and foreign intelligence to prevent future attacks. These efforts have in turn motivated us to collect data and undertake this paper work as a challenge.

1.2 The contribution of this paper

• In supervised learning for e-mail classification we introduced new classification algorithm named Ad-Infinitum which is particularly suitable for classifying text documents. It is fast, easy to tune, and can handle large feature sets. We compare the learning behavior of Ad Infinitum with well established algorithms such as Decision Tree (DTs), Support Vector Machines (SVM) and Naive Bayes (NB) and show that Ad Infinitum outperforms well.

• We have implemented the feature selectors, TFV and IG and show that IG outperforms the widely used and computationally more expensive TFV.

• We conduct a large scale algorithm performance evaluation on the benchmark spam filtering corpora LingSpam and PU1. We compare Ad Infinitum, DT, SVM and NB, using the same version of the corpora, the same pre-processing and optimising the parameters of all algorithms.

• We show empirically that Ad Infinitum can be successfully used to learn from a small number of labeled examples in the domain of threatening e-mail detection.

1.3 Organization of the Paper

The paper is organized as follows: Section 2 defines problem statement and related works in this area. Section 3 describes the proposed work and implementation. Section 4 illustrates the experimental setup .Section 5 discusses about the experimental result and discussion. Section 6 discusses the performance evaluation; Section 7 concludes the paper and points out some potential future work.

2. Problem Statement and Related Work

E-mail classification (e.g. Threatening e-mail detection) is a supervised learning problem. It can be formally stated as follows. Given a training set of labeled e-mail documents $D_{train}= \{ (d_1, C_1), \ldots, (d_n, C_n) \}$,where d_i is an e-mail document from a document set D and C_i is the label chosen from a predefined set of categories C, the goal is to induce a hypothesis (classifier) $h : D \rightarrow C$ that can correctly classify new, unseen e-mail documents D_{test} , $D_{test} \not\subset D_{train}$. Here C contains two labels: Threatening and Normal (legitimate).

It is hard to remember what our lives were like without e-mail. Though e-mail was originally developed for sending simple text messages, it has become more robust in the last few years. So, it is one possible source of data from which potential problem can be detected. Thus the problem is to find a system that identifies the deception in communication through e-mails by which the security enforcing methods can be strengthened. Also we can prevent the occurrence of future attacks.

E-mail classification has been an active area of research. Cohen [7] developed a propositional learning algorithm RIPPER to induce keyword-spotting rules for filing e-mails into folders. The multi-class problem was transformed into several binary problems by considering one folder versus all the others. The comparison with the traditional information retrieval method Rocchio indicated similar accuracies. Cohen argued that keyword spotting rules are more useful as they are easier to understand, modify and can be used in conjunction with user-constructed rules.

Bayesian approaches are the most widely used in text categorization and e-mail classification. They allow quick training and classification and can be easily extended to incremental learning. While rule-based approaches make binary decisions, probabilistic techniques provide a degree of confidence of the classification which is an advantage, especially for cost-sensitive evaluation. Sahami et al. [24] applied NB for spam e-mail filtering using bag of words representation of the e-mail corpora and binary encoding. The performance improved by the incorporation of hand-crafted phrases (e.g. ‘‘FREE!’’, ‘‘be over 21’’) and domain-specific features such as the domain type of the sender and the percentage of non-alphabetical characters in the subject. Rennie’s iFile [22] uses NB to file e-mails into folders and suggest the three most suitable folders for each message. The system applies stemming, removes stop words and uses document frequency threshold as feature selector. SpamCop [17] is a system for spam e-mail filtering also based on NB. Both stemming and a dynamically created stop word list are used. The authors investigated the effect of the training data size, different ratios of spam and non-spam e-mails, use of trigrams instead of words and also showed that SpamCop outperforms Ripper. Provost’s experiments [18] also confirmed that NB outperforms Ripper in terms of classification accuracy on both filing e-mail into folders and spam filtering.

MailCat [28] uses a Nearest-neighbor (k-NN) technique and tf-idf representation to file e-mails into folders. k-NN supports incremental learning but requires significant time for classification of new e-mails. Androutsopoulos et al. [5] found that Naive Bayes and a k-NN technique called TiMBL clearly outperform the keyword- based spam filter of Outlook 2000 on the LingSpam corpora. Ensembles of classifiers were also used for spam filtering. Sakkis et al. [25] combined a NB and k-NN by stacking and found that the ensemble achieved better performance. Carreras et al. [6] showed that boosted trees outperformed decision trees, NB and k-NN. Rios and Zha [23] applied RF for spam detection on time indexed data using a combination of text and meta data features. For low false positive spam rates, RF was shown to be comparable with SVM in classification accuracy.

One of the first Decision tree algorithms, ID3 [20], has applied the Chi-square statistic to the null hypothesis about the irrelevance of a test attributes. However, most other methods of decision tree learning like CART and C4.5 [19], have adopted the post pruning approach for the sake of exploring a large set of potentially valid patterns. [26] developed incremental versions of decision tree induction algorithm (system ID4) where uncertain instances are labeled by an expert. [30] Proposed ID5: Optimization of Tree generated by ID4 (systems ID5 or ITI)). [13] developed a method based on singular value decomposition to detect unusual and deceptive communication in e-mail. The problem with this approach is that it does not deal with incomplete data in an efficient and elegant way and cannot incorporate new data incrementally without having to reprocess the entire matrix.

A list of existing approaches for classification and detection of threatening e-mails is only limited. [3] applied a decision tree (ID3) algorithm for threatening e-mail detection. The problem with this approach is that it cannot give reliable rules which lead to misclassification of correct record**.** [1] Presented Association rule mining for suspicious e-mail detection and problem with approach is that computational time is relatively slow. [2] compared to a cross experiment between four classification methods including Decision Tree, Naive Bayes,SVM and NN for the classification of e-mail in to threatening or normal and decision tree performed better than other classifiers.

We extend previous research on supervised e-mail classification by: (1) Applying Ad Infinitum algorithm for detecting threatening e-mails, comparing Ad Infinitum with a number of state-of-the-art classifiers and showing that it is the better choice in terms of accuracy, running and classification time and simplicity to tune. (2) Introducing a new feature selector that is accurate and computationally efficient. (3)Comparing the performance of a large number of algorithms on the benchmark corpora for threatening e-mail detection using the same version of the data and the same pre-processing.

3. Proposed work and Implementation

3.1 Classifier construction framework

Classification is the process of finding a set of models that describe and distinguish data, classes and concepts, for the purpose of being able to use the model to predict the class of objects whose class label is unknown. In this paper, we develop Ad Infinitum algorithm which aims to identify the threatening e-mail from e-mail corpus. In contrast to previous decision tree algorithms, our proposed algorithm provides optimal and reliable decision for situations which cannot be handled by majority voting concept in decision tree induction algorithm.

3.2 E-mail Corpora used in the experiment

We used two publicly available [14] corpora: LingSpam, PU1, as shown in Table 2. LingSpam [5] was created by mixing spam e-mails received by a user with legitimate e-mails sent to the Linguist mailing list. PU1 [4] consists of e-mails received by a user over three years. A representative dataset was constructed taking into account the e-mails received by regular correspondents and removing duplicate e-mails received on the same day. The corpus was encrypted for privacy reasons by replacing each token with a number.

Table 1: A sample e-mail data set used in the experiment

E-mail	Message
1	The IISC Scientist Conference hall should be blast on tomorrow at 10.00 a.m
2	Today there will be bomb blast in parliament house and the US consulates in India at 11.46 a.m.Stop it, if you could. Cut relations with the U.S.A.Long Live Osama Finladen Asadullah Alkalfi.
3	Indian airlines Boeing-727 from Chennai to Delhi will be hijacked on tomorrow
4	Beware, Tomorrow, there will be a bomb blast at CM' office.
5	There is going to be terrorist attack in Chennai airport

Table 2: Spam filtering Corpora –Statistics

Corpus	# E-mails	# Spam E-mails	# Legitimate E-mails
PU1	1099	481	618
LingSpam	2893	481	2412

The problem we faced when trying to test out new ideas dealing with e-mail systems was an inherent limitation of the available threatening e-mails because we only have access to our own data, our results and experiments, no doubt reflect some bias. In order to avoid the problem, we have created two corpora: TCEThreatening 1, TCEThreatening 2 which is a mixture of our own threatening e-mail data set with bench mark spam filtering corpora Ling Spam and PU1 as shown in Table 3.

Table 3: Email Corpus used in the experiment

Corpus	# E-mails	# Threatening E-mails	# Spam E-mails	# Legitimate E-mails
TCEThreatening 1	2099	500	481	1118
TCEThreatening 2	3893	500	481	2912

3.3 Pre-processing of the corpora

The first step in the process of constructing a classifier is the transformation of the e-mails into a format suitable for the classification algorithms. We have developed a generic architecture for e-mail categorization called automated feature selection. It supports the bag of words representation which is most commonly used in e-mail categorization. All unique terms (tokens, e.g. words, special symbols, numbers etc.) in the entire training corpus are identified and each of them is treated as a single feature. A feature selection mechanism is applied to choose the most important terms and reduce dimensionality. Each document is then represented by a vector that contains a normalized weighting for every selected term which represents the importance of that term in the document.

3.4 Term extraction in TCEThreatening 1 and TCEThreatening 2

Here the Body and Subject were used, and the following headers were parsed and tokenized: Sender, Recipient and Subject. Attachments are considered as a part of the body and are processed in their original format (binary, text, and html). All these fields were treated equally and a single bag of words was created for each e-mail.

3.5 Features, weighting and normalization.

Feature selection is an important step in e-mail categorization as e-mail documents have a large number of terms. Removing the less informative and noisy terms reduces the computational cost and improves the classification performance. The features are ranked according to the feature selection mechanism and those with value higher than a threshold are selected. In this paper, we implemented two feature selectors: Information Gain (IG) and Term Frequency Vector (TFV). IG is the most popular and one of the most successful feature selection techniques used in text categorization. Given a set of possible categories C= {c1… ck}, the IG of a feature f is defined as

$$I(G)=\sum^{k}_{i=1} P(c_i) \log P(c_i) + P(f) \sum^{k}_{i=1} P(c_i|f) \log P(c_i|f) + P(f)^{-1} \sum^{k}_{i=1} P(c_i|f)^{-1} \log P(c_i|f)^{-1} \quad (1)$$

It measures the known amount about the presence or absence of a document and helps us to predict the category. The importance of the feature is measured globally as the computation is done for each feature across all categories.IG has quadratic time complexity.

Like IG, TFV is category dependent. For each term f, we compute the term frequency (tf) in each category and then calculate the variance as

$$TFV(f) = \sum^{k}_{i=1} [tf(f, c_1) - mean\_tf(f)]^2 \quad (2)$$

The normalizing factor from the standard variance formula is ignored, as our goal is to rank features and select the ones with the highest score. Features with high variance across categories are considered informative and are selected. For example, terms that occur predominantly in some of the categories will have high variance and terms that occur in all categories will have low variance. TFV can be seen as an improvement of the document frequency, which is the simplest method for feature reduction. For each term in the training corpora, document frequency counts the number of documents in which the term occurs and selects features with frequency above a predefined threshold. Yang and Pedersen [34] compared several feature selectors and found that document frequency is comparable to the best performing techniques (IG and for feature reduction up to 90%). They concluded that document frequency is not just an adhoc approach but a reliable measure for selecting informative features. However, document frequency is category independent and will simply select terms with high DF no matter about their distribution in the categories. TFV addresses this problem by not selecting terms with high document frequency if they appear frequently in each category, i.e. are not discriminating. Both TFV and document frequency is highly scalable as they have linear complexity. Our approach incorporates the three most popular feature weighting mechanisms: (1) binary, (2) term frequency and (3) term frequency-inverse document frequency (tf-idf). In the first method weights are either 0 or 1 denoting absence or presence of the term in the e-mail. In the term frequency method the weights correspond to the number of times the feature occurs in the document. Term frequency weights are more informative than the binary weights. The third method assigns higher weights to features that occur frequently, but also balances this by reducing the weight if a feature appears in many documents.

The time taken to build the classifier is shown in Table 4. It is an important consideration as classifiers must be kept up to date and this requires re-training. The fastest algorithm was NB (although it was the less accurate), the slowest was SVM, AD-INFINITUM was fast enough –it built a classifier for 4.715 s on average.

Table 4: Time(s) to build the classifier for IG (results for TFV are similar)

	AD-INFINITUM	DT	SVM	NB
TCETHREATEN 1	3.25	3.55	37.83	0.41
TCETHREATEN 2	6.18	6.58	70.17	0.75
Average	4.715	5.07	54.00	0.58

3.6 Proposed Ad Infinitum Classifier

The basic algorithm for decision tree induction is a greedy algorithm that constructs decision trees in a top-down recursive divide-and-conquer manner. The algorithm, summarized given below, is a version of ID3, a well-known decision tree induction algorithm.

Table 5: Algorithm for Decision tree induction

```
Algorithm:
            Generate a decision tree from the given training data.
  Input:    The training samples, test samples, represented by discrete-valued attributes;
            the set of   candidate attributes, attribute-list.
  Output:   A decision tree and set of rules.
```

```
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 $a_i$ of test-attribute //partition the samples
9)  grow a branch from node N for the condition test-attribute= $a_i$ ;
10) let $s_i$ be the set of samples in samples for which test-attribute= $a_i$;// a partition
11) if $s_i$ is empty then
12) attach a leaf labeled with the most common class in samples;
13) else attach the node returned by Generate Extended_Decision_tree ($s_i$ ,attribute-
    list-test-attribute);
```

The basic strategy is as follows.

- The tree starts as a single node representing the training samples (step 1).
- If the samples are all of the same class, then the node becomes a leaf and is labeled with that class (steps 2 and 3).
- Otherwise, the algorithm uses an entropy-based measure known as information gain as a heuristic for selecting the attribute that will best separate the samples into individual classes (step6). This attribute becomes the “test” or “decision” attribute at the node (step7). In this version of the algorithm, all attributes are categorical, that is, discrete-valued. Continuous-valued attributes must be discretized.
- A branch is created for each known value of the test attribute and the samples are partitioned accordingly (steps 8-10).
- The algorithm uses the same process recursively to form a decision tree for the samples at each partition. Once an attribute has occurred at a node, it need not be considered in any of the node’s descendents (step 13).
- The recursive partitioning stops only when any one of the following conditions is true:
    1. All samples for a given node belong to the same class (steps 2 and 3), or
    2. There are no remaining attributes on which the samples may be further partitioned (step 4). In this case, **majority voting** is employed (step 5). This involves converting the given node into leaf and labeling it with the

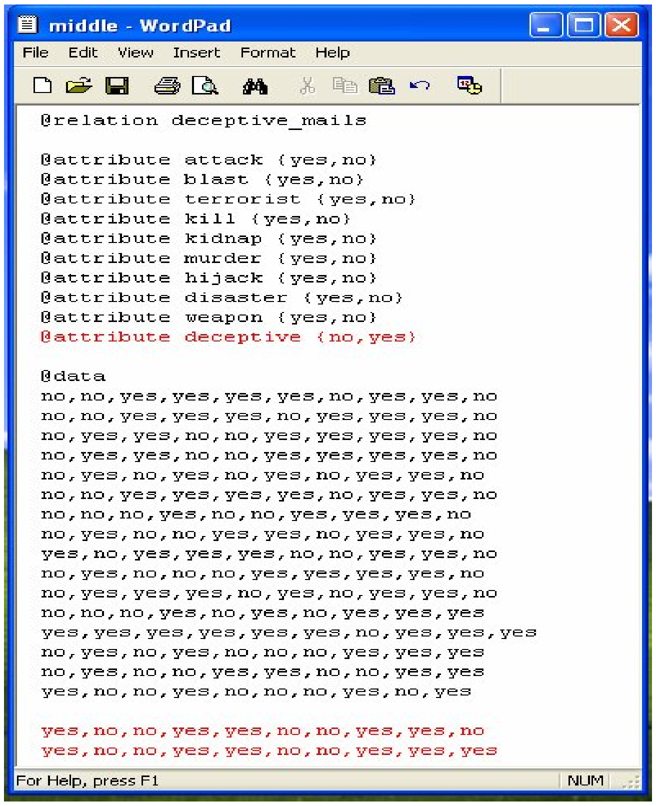
class in majority among samples. Alternatively, the class distribution of the node samples may be stored.

3. There are no samples for the branch test=attribute=a_i (step 11). In this case, a leaf is created with the majority class in samples (step 12).

3.6.1 Unhandled Exception of the Decision tree induction algorithm

We have used TCEThreatening 1 and TCEThreateing 2 corpora for the decision tree induction algorithm to compute and inferred that different set of rules were developed for same training dataset. In order to prove this, we have used the open source software WEKA to examine the training dataset. WEKA uses attribute-file-format (*.arff), so we have converted our dataset into attribute file format, which is given in the Table 6 and 7. In these tables we have given the training dataset and the attribute values are specified in attribute file format. In these tables the class attribute and its values are specified in red color. From the training dataset we expect that the same set of rules would be generated, since both training dataset are same. The records highlighted with the red color are important because the decision tree induction algorithm does not handle these instances correctly and generates incorrect rules. In the forthcoming discussion we will confirm that the rules generated by decision tree induction algorithm are not reliable and hence error occurs in the classification of the training dataset as well as test set. For the software WEKA, the attributes and their possible values are given in the general format as @attribute attribute_name {possible values for the attribute} and hence the two inputs are same.

Table 6: A Sample Training Data set 1



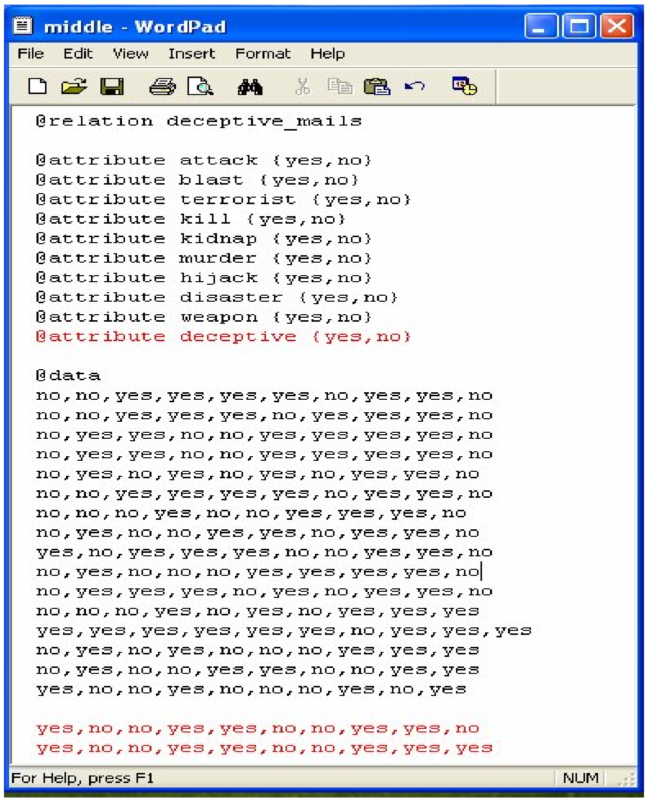
```
@relation deceptive_mails

@attribute attack {yes,no}
@attribute blast {yes,no}
@attribute terrorist {yes,no}
@attribute kill {yes,no}
@attribute kidnap {yes,no}
@attribute murder {yes,no}
@attribute hijack {yes,no}
@attribute disaster {yes,no}
@attribute weapon {yes,no}
@attribute deceptive {no,yes}

@data
no,no,yes,yes,yes,yes,no,yes,yes,no
no,no,yes,yes,yes,no,yes,yes,yes,no
no,yes,yes,no,no,yes,yes,yes,yes,no
no,yes,yes,no,no,yes,yes,yes,yes,no
no,yes,no,yes,no,yes,no,yes,yes,no
no,no,yes,yes,yes,yes,no,yes,yes,no
no,no,no,yes,no,no,yes,yes,yes,no
no,yes,no,no,yes,yes,no,yes,yes,no
yes,no,yes,yes,yes,no,no,yes,yes,no
no,yes,no,no,no,yes,yes,yes,yes,no
no,yes,yes,yes,no,yes,no,yes,yes,no
no,no,no,yes,no,yes,no,yes,yes,yes
yes,yes,yes,yes,yes,yes,no,yes,yes,yes
no,yes,no,yes,no,no,no,yes,yes,yes
no,yes,no,no,yes,yes,no,no,yes,yes
yes,no,no,yes,no,no,no,yes,no,yes

yes,no,no,yes,yes,no,no,yes,yes,no
yes,no,no,yes,yes,no,no,yes,yes,yes
```

Table 7: A Sample Training Data set 2



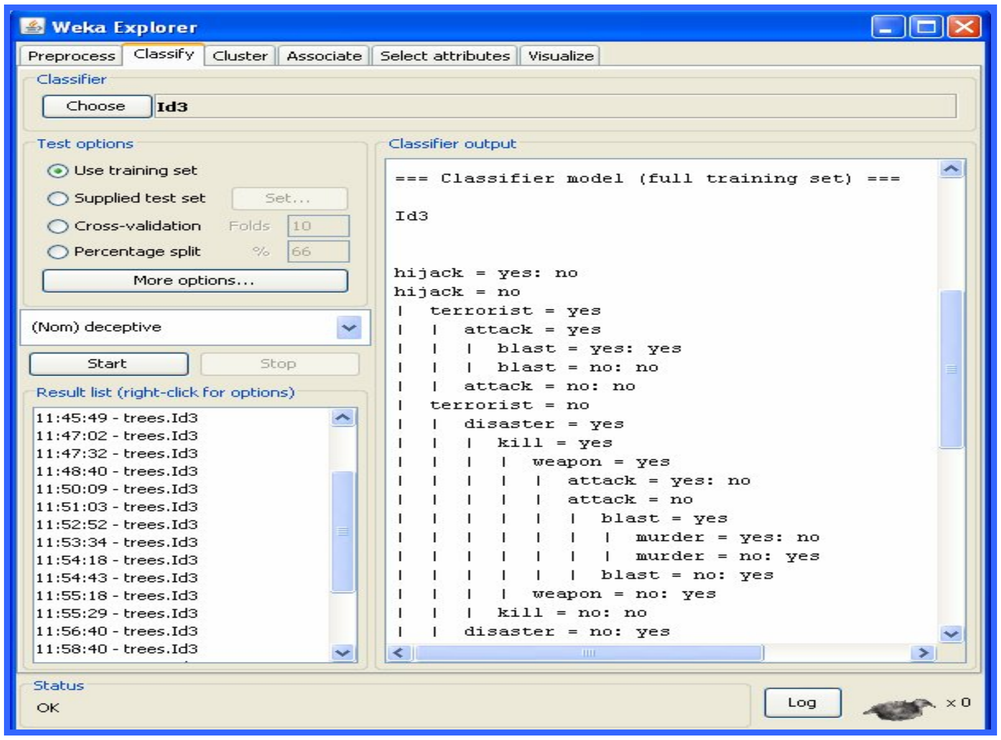
```
@relation deceptive_mails

@attribute attack {yes,no}
@attribute blast {yes,no}
@attribute terrorist {yes,no}
@attribute kill {yes,no}
@attribute kidnap {yes,no}
@attribute murder {yes,no}
@attribute hijack {yes,no}
@attribute disaster {yes,no}
@attribute weapon {yes,no}
@attribute deceptive {yes,no}

@data
no,no,yes,yes,yes,yes,no,yes,yes,no
no,no,yes,yes,yes,no,yes,yes,yes,no
no,yes,yes,no,no,yes,yes,yes,yes,no
no,yes,yes,no,no,yes,yes,yes,yes,no
no,yes,no,yes,no,yes,no,yes,yes,no
no,no,yes,yes,yes,yes,no,yes,yes,no
no,no,no,yes,no,no,yes,yes,yes,no
no,yes,no,no,yes,yes,no,yes,yes,no
yes,no,yes,yes,yes,no,no,yes,yes,no
no,yes,no,no,no,yes,yes,yes,yes,no
no,yes,yes,yes,no,yes,no,yes,yes,no
no,no,no,yes,no,yes,no,yes,yes,yes
yes,yes,yes,yes,yes,yes,no,yes,yes,yes
no,yes,no,yes,no,no,no,yes,yes,yes
no,yes,no,no,yes,yes,no,no,yes,yes
yes,no,no,yes,no,no,no,yes,no,yes

yes,no,no,yes,yes,no,no,yes,yes,no
yes,no,no,yes,yes,no,no,yes,yes,yes
```

The table 6 & 7 were given as input to WEKA and the classification is done on the training dataset using the decision tree induction algorithm. Training data is chosen to be the test option. According to this option, the generated rules are used to classify the given dataset. For the training data in Table 6, WEKA's tree structure is provided in Fig.1. For the given data the decision tree induction algorithm starts its recursive function of identifying the attribute with the highest information gain. On the first recursion, the attribute named "hijack" gets highest information gain that it divides the dataset into two half, which is the core idea behind the algorithm. The recursive function identifies the attributes at each iteration and the rules are generated. The tree structure is given in the Fig.2 and a node with the red color is important because this is the single rule that differs from the training data in table 6. This rule is generated because this situation satisfies the condition in steps 4 and 5 of the decision tree induction algorithm (**majority voting**).



Classifier: Id3

Test options: [✓ Use training set]

Folds: 10

%: 66

(Nom) deceptive

```
11:45:49 - trees.Id3
11:47:02 - trees.Id3
11:47:32 - trees.Id3
11:48:40 - trees.Id3
11:50:09 - trees.Id3
11:51:03 - trees.Id3
11:52:52 - trees.Id3
11:53:34 - trees.Id3
11:54:18 - trees.Id3
11:54:43 - trees.Id3
11:55:18 - trees.Id3
11:55:29 - trees.Id3
11:56:40 - trees.Id3
11:58:40 - trees.Id3
```

Classifier output

```
=== Classifier model (full training set) ===

Id3


hijack = yes: no
hijack = no
|   terrorist = yes
|   |   attack = yes
|   |   |   blast = yes: yes
|   |   |   blast = no: no
|   |   attack = no: no
|   terrorist = no
|   |   disaster = yes
|   |   |   kill = yes
|   |   |   |   weapon = yes
|   |   |   |   |   attack = yes: no
|   |   |   |   |   attack = no
|   |   |   |   |   |   blast = yes
|   |   |   |   |   |   |   murder = yes: no
|   |   |   |   |   |   |   murder = no: yes
|   |   |   |   |   |   blast = no: yes
|   |   |   |   weapon = no: yes
|   |   |   kill = no: no
|   |   disaster = no: yes
```

Status

OK

Figure (1): WEKA's tree structure for training dataset in Table 6

The majority voting is the stopping condition for the recursive partitioning and the possible data at that position is given in the Table 8.

Table 8: A sample partitioned data from training tuples in Table 6

Attack	Blast	Murder	Threatening
no	yes	Yes	no
yes	no	No	no
yes	no	No	Yes

From this Table 8 "attack" gets the highest information gain and hence it splits the table into two, one is when attack = yes, then class threaten = no and the other condition is when attack = no, then the recursive function is applied on the Table 8.

Table 9: Example for situation where majority voting not possible

Blast	Murder	Threatening
no	No	no
no	No	yes

Finally the entire tree structure is generated and the possible set of rules generated are as follows.

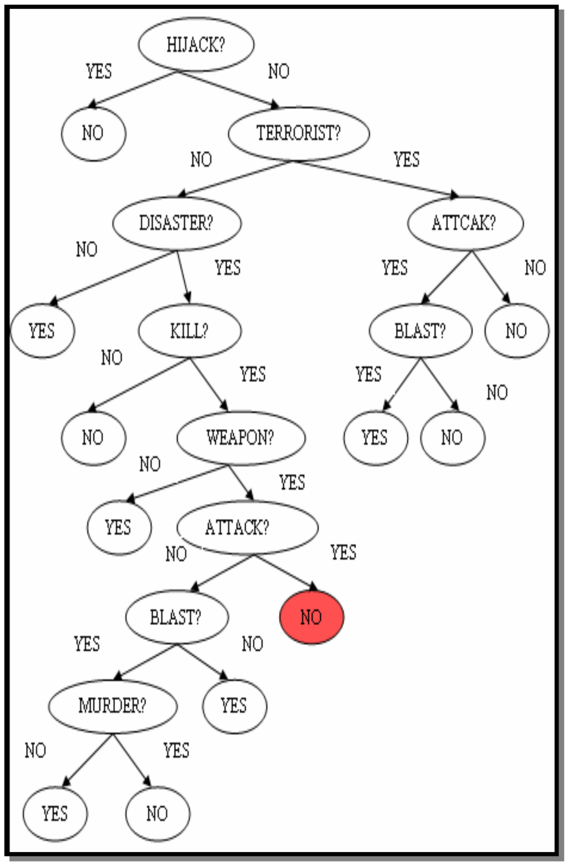


Figure (2): Generated Tree for Table 6

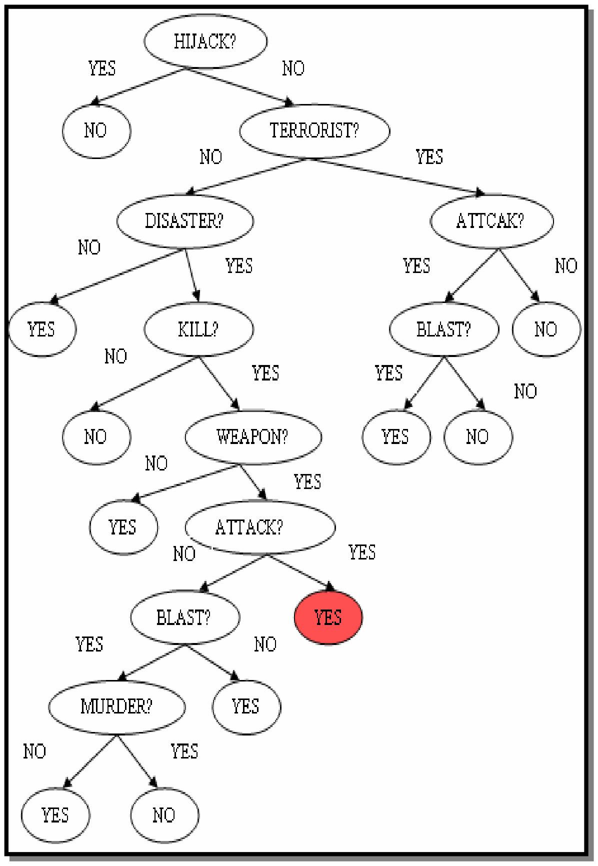


Figure (3): Generated Tree for Table 7

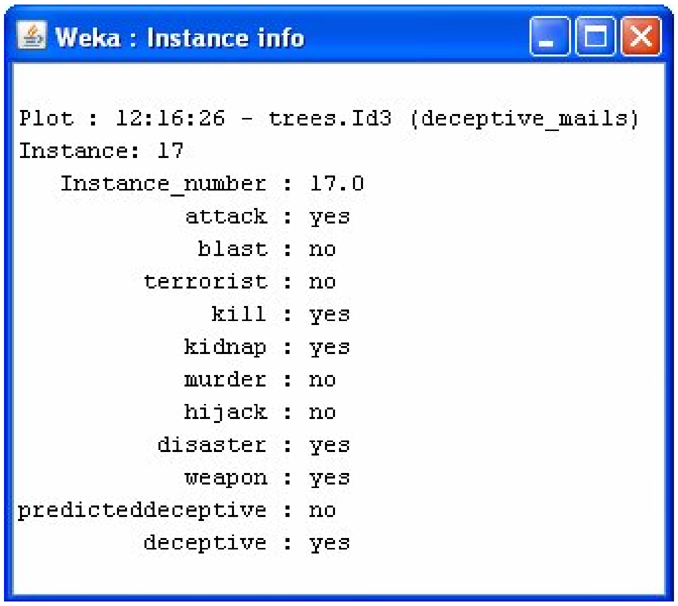
Generated rules for Table 6

1. If hijack=yes then threatening=no
2. If hijack=no & terrorist=yes & attack=yes & blast=yes then threatening=yes
3. If hijack=no & terrorist=yes & attack=yes & blast=no then threatening=no
4. If hijack=no & terrorist=yes & attack=no then threatening=no

5. **If hijack=no & terrorist=no & disaster=yes & kill=yes & weapon=yes & attack=yes then threatening= no**
6. If hijack=no & terrorist=no & disaster=yes & kill=yes & weapon=yes & attack=no & blast=yes & murder=yes then threatening=no
7. If hijack=no & terrorist=no & disaster=yes & kill=yes & weapon=yes & attack=no & blast=yes & murder=no then threatening=yes
8. If hijack=no & terrorist=no & disaster=yes & kill=yes & weapon=yes & attack=no & blast=no then threatening=yes
9. If hijack=no & terrorist=no & disaster=yes & kill=yes & weapon=no then threatening=yes
10. If hijack=no & terrorist=no & disaster=yes & kill=no then threatening=no
11. If hijack=no & terrorist=no & disaster=no then threatening=yes

3.6.2 Identification of the Inconsistent data from Training tuples in Table 6

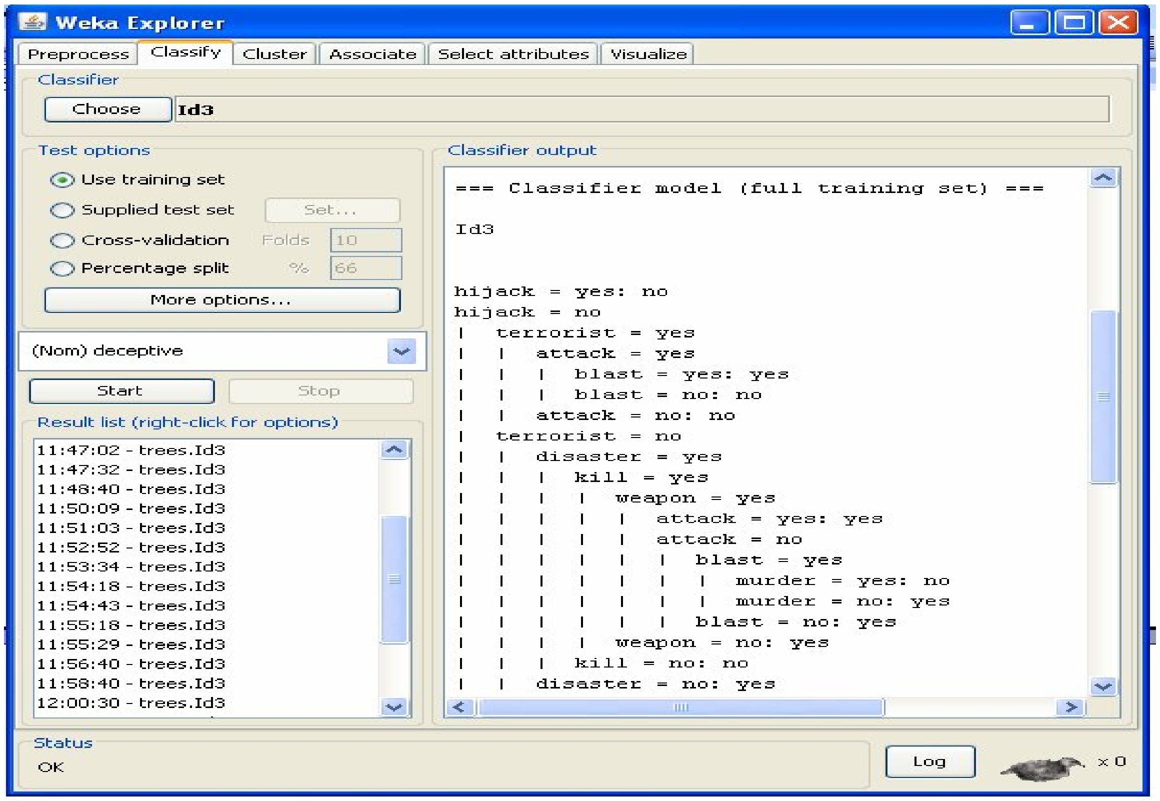
Since the generated rules compare the training data given and classifies according to the class label value for the particular rule, we consider that the rules are highly reliable and the algorithm classifies correctly. During the comparison, rule no.5 predicts that the instance 17 should have class label “Threatening= no” but the record has the class label value as “Threatening= yes” and thus the algorithm identifies the inconsistent instance.



```
Plot : 12:16:26 - trees.Id3 (deceptive_mails)
Instance: 17
   Instance_number : 17.0
            attack : yes
             blast : no
         terrorist : no
              kill : yes
            kidnap : yes
            murder : no
            hijack : no
          disaster : yes
            weapon : yes
predicteddeceptive : no
         deceptive : yes
```

Figure (4): Instance information for Table 6

Now we are examining the Table 7 and expecting that the rules generated should be the same and the same 17th instance should be classified as inconsistent.



Classifier: Id3

Test options: [✓ Use training set] Supplied test set; Cross-validation Folds 10; Percentage split % 66

(Nom) deceptive

```
11:47:02 - trees.Id3
11:47:32 - trees.Id3
11:48:40 - trees.Id3
11:50:09 - trees.Id3
11:51:03 - trees.Id3
11:52:52 - trees.Id3
11:53:34 - trees.Id3
11:54:18 - trees.Id3
11:54:43 - trees.Id3
11:55:18 - trees.Id3
11:55:29 - trees.Id3
11:56:40 - trees.Id3
11:58:40 - trees.Id3
12:00:30 - trees.Id3
```

```
=== Classifier model (full training set) ===

Id3


hijack = yes: no
hijack = no
|   terrorist = yes
|   |   attack = yes
|   |   |   blast = yes: yes
|   |   |   blast = no: no
|   |   attack = no: no
|   terrorist = no
|   |   disaster = yes
|   |   |   kill = yes
|   |   |   |   weapon = yes
|   |   |   |   |   attack = yes: yes
|   |   |   |   |   attack = no
|   |   |   |   |   |   blast = yes
|   |   |   |   |   |   |   murder = yes: no
|   |   |   |   |   |   |   murder = no: yes
|   |   |   |   |   |   blast = no: yes
|   |   |   |   weapon = no: yes
|   |   |   kill = no: no
|   |   disaster = no: yes
```

Status: OK

Figure (5): WEKA's tree structure for training dataset in Table 7

But one of the generated rule gives contradictory result and that rule classifies the 17th instance correctly and classifies the 16th instance as inconsistent record. The generation of contradictory rule is because the majority voting is the stopping condition for the recursive partitioning and the possible data at that position is given in the Table 10.

Table 10: A sample partitioned training data set for Table 7

Attack	Blast	Murder	Threatening
yes	yes	yes	yes
no	yes	yes	no
no	yes	no	yes

From this Table 10 "attack" gets the highest information gain and hence it splits it into two, one is when attack = yes, then class Threaten = no and the other condition is when attack = no, then the recursive function is applied on the table.

Table 11: Example for situation where majority voting not possible

Blast	Murder	Threatening
yes	yes	no
yes	no	yes

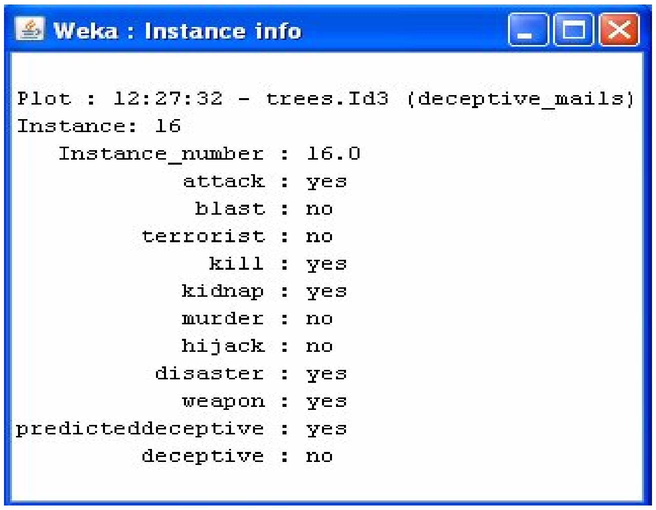
Finally the entire tree structure is generated and the possible set of rules generated are as follows

Generated rules for Table 7

1. If hijack=yes then threatening=no
2. If hijack=no & terrorist=yes & attack=yes & blast=yes then threatening=yes
3. If hijack=no & terrorist=yes & attack=yes & blast=no then threatening=no
4. If hijack=no & terrorist=yes & attack=no then threatening=no
5. **If hijack=no & terrorist=no & disaster=yes & kill=yes & weapon=yes & attack=yes then threatening= yes**
6. If hijack=no & terrorist=no & disaster=yes & kill=yes & weapon=yes & attack=no & blast=yes & murder=yes then threatening=no
7. If hijack=no & terrorist=no & disaster=yes & kill=yes & weapon=yes & attack=no & blast=yes & murder=no then threatening=yes
8. If hijack=no & terrorist=no & disaster=yes & kill=yes & weapon=yes & attack=no & blast=no then threatening=yes
9. If hijack=no & terrorist=no & disaster=yes & kill=yes & weapon=no then threatening=yes
10. If hijack=no & terrorist=no & disaster=yes & kill=no then threatening=no
11. If hijack=no & terrorist=no & disaster=no then threatening=yes

3.6.3 Identification of the Inconsistent data from training tuples in Table 7

Since the generated rules compare the training data given and classifies according to the class label value for the particular rule, we consider that the rules are highly reliable and the algorithm does correct classification. During the comparison, rule no.5 predicts the 16th instance should have class label "Threatening= yes" but the record has the class label values as "Threatening= no" and thus the algorithm identifies the inconsistent instance.


Weka : Instance info

```
Plot : 12:27:32 - trees.Id3 (deceptive_mails)
Instance: 16
   Instance_number : 16.0
            attack : yes
             blast : no
         terrorist : no
              kill : yes
            kidnap : yes
            murder : no
            hijack : no
          disaster : yes
            weapon : yes
predicteddeceptive : yes
         deceptive : no
```

Figure (6): Instance Information for Table 7

Thus for the same set of data different classification rules are generated and the instances in Table 6 which are correctly classified are classified as inconsistent records when Table

7 is given as the input. This type of misconception occurs when recursive partitioning faces a data set which has equal number of records but the concept of majority voting works correctly when number of records is unequal. For example, consider the Table 12.

Table 12: A Sample training data set

Weapon	Threatening(Class label)
Yes	Yes
Yes	Yes
No	Yes
Yes	No

When table 12 is to be partitioned, the **majority voting** can be applied. According to majority voting, the distinct count of the attribute is found and highest distinct value is taken into account. Here the value of weapon = yes occurs three times and hence it is taken as the value for the attribute "weapon" and then its corresponding count on class label values is taken into account where the value "weapon = yes" occurs 3 times and "threatening = yes" occurs 2 times and hence the rules generated for the dataset will be

If "weapon = yes" then "threatening = yes"

Now consider the Table 13 & 14 which indicates the problem with majority voting and these kind of conditions are not handled in the decision tree induction algorithm. For example

Table 13: A sample training dataset

weapon	Threatening(Class label)
yes	Yes
yes	no
no	yes
no	no

Table 14: A sample training dataset

weapon	Threatening(Class label)
yes	yes
yes	yes
yes	no
yes	no

In the Table 13 the count of weapon = yes is 2 and weapon = no is 2, so majority voting cannot be applied on the attribute "weapon" and hence the rules cannot be generated.

If "weapon =?" then "threatening=?"

In table 14 the count of weapon = yes is 4 and hence majority voting can be applied to attribute "weapon" but deciding of the class label "threatening" cannot be determined because the count of "threatening = yes" is 2 and "threatening = no" is 2 and hence the rules cannot be generated

If "weapon = yes" then "threatening =?"

Since the Table 13 and 14 faces the same problem, WEKA developers cannot provide any specific solution to this problem, so they have used deciding factor in the attribute

value declaration. When table 13 is given as input to WEKA, the tool faces the same problem and so it has referenced the value given in the attribute declaration.
For the Table 13, when attribute declaration is given as

@attribute weapon {yes, no}	@attribute weapon {no, yes}
@attribute threatening {yes, no}	@attribute threatening {no, yes}
Then, the rule generated is	Then, the rule generated is
If "weapon =yes" then "threatening =yes"	**If "weapon =no" then "threatening =no"**

Similarly for the Table 14 we can obtain the required result in WEKA by just changing the attribute declaration. Thus the decision tree induction algorithm does not handle this problem.

3.6.4 Ad Infinitum algorithm

For this problem, the possible solution is that the user should assign the value of the class label under this situation. The class value given by the user would be the best solution and the machine learning algorithms cannot give optimal solution. Hence Ad infinitum would be the extension of the decision tree induction algorithm by giving best solution not giving contradictory rules.

Table 15: The Ad infinitum Tree – updating terminating condition of decision tree induction algorithm

```
1) if attribute-list is empty then
2) if training data rules is null
3) return N as a leaf node labeled with the most common class in samples // Majority
   voting
     • This rule traversal from the root to the leaf is alpha rule.
     • If (record satisfy alpha rule)
          ❖ Correctly classified as (as normal Decision Tree Induction).
     • Else
          ❖ Incorrectly classified (as normal Decision Tree Induction).
4) else
5) return N as a leaf node labeled with the attribute corresponding to class label
   value from the user guided class value // Agent(user)
     • This rule traversal from the root to the leaf is beta rule.// An unique rule
     • If (record satisfy beta rule)
          ❖ Correctly classified.(This type of records cannot be handled by
            Decision Tree Induction algorithm)
     • Else
          ❖ Incorrectly classified.
```

Table 16: The proposed Ad Infinitum algorithm

```
Algorithm:
            Ad Infinitum. Generate a decision tree from the given training data.
  Input:    The training samples, test samples, represented by discrete-valued
            attributes; the set of candidate attributes, attribute-list.
  Output:   A decision tree and set of rules.
  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) if training data rules is null
    6)  return N as a leaf node labeled with the most common class in samples //
        Majority voting
    7)  else
    8)  return N as a leaf node labeled with the attribute corresponding to
        class label value from the user guided class value // Agent(user)
    9)  select test-attribute, the attribute among attribute-list with the highest
        information gain;
    10) label node N with test-attribute;
    11) for each known value a_i of test-attribute //partition the samples
    12)  grow a branch from node N for the condition test-attribute= a_i ;
    13)  let s_i be the set of samples in samples for which test-attribute= a_i;//a
         partition
    14)  if  s_i is empty then
    15)  attach a leaf labeled with the most common class in samples;
    16)  else attach the node returned by Generate extended_decision_tree (s_i
         ,attribute-list-test-attribute);
```

In Ad infinitum algorithm, any dataset considered for classification would fall under any one of the categories:-

a. Consistent data
b. Inconsistent data identified by alpha rules
c. Inconsistent data identified by beta rules

But in the Decision tree induction algorithm any dataset would fall under any one of the following categories:-

1. Consistent data
2. Inconsistent data

3.6.5 Alpha and Beta rules

The inconsistent data are those which are not classified in the given training dataset. Hence efficiency of the algorithm decreases due to the inability to classify the given data.

The inconsistent instances are identified by alpha rules. The noisy data occurs in the training data due to human error, that is, the user gives irrelevant value in the dataset. Hence the algorithm should be developed to handle these noisy data too.

Table 17: A Sample training data set

Weapon	Threatening(Class label)
yes	yes
yes	yes
no	yes
yes	no

When a rule is formed by a set of instances in which the value of the class label counts more than 50% than its counterpart, then that rule is acceptable and those instances which contradicts this rule will be recorded as noisy data. The inconsistent records considered as noise are identified by beta rules. This idea is clearly depicted in the Table 17 in which the attribute weapon has 2 “yes” values with the class label “yes” and a record with the class label “no”. The record which is classified as inconsistent by the decision tree induction algorithm (i.e. the third and fourth record) should be removed before the training dataset is considered for classification as noisy data. This automatically increases the efficiency of the algorithm drastically.

4. Experimental Setup

We used the TCEThreatening 1 and TCEThreatening2 corpus, with the standard bag of words representation and IG for feature selection. Each email was broken up into two sections: - The words found in the subject header and the words found in the main body of the message. A summary of the feature sets (views) used in the experiment is given in Table 18. The feature selection was applied individually to each view of the data using IG. Upon inspection of the word lists and their IG values, it was decided that the top 100 words was a suitable cut-off. This depicts a drastic dimensionality reduction.

Table 18: Feature sets (views used)

View	Description
Body	All words that appear in the body of an e-mail
Subject	All words that appear in subject of an e-mail
All	All words that appear in the body and subject of an e-mail

Table 19: Top 20 features selected by IG for the feature set: Body

S.No	Body
1	Bomb
2	Attack
3	Blast
4	Terrorist
5	Kill
6	Kidnap
7	Murder
8	Hijack
9	Disaster
10	Danger
11	Weapon
12	Destroy
13	Explode
14	Capture
15	Demolish
16	Assassinate
17	Slaughter
18	Damage
19	Assail
20	Shoot

Table 20 contains the accuracy results obtained using 10-fold cross validation. Except with NB with subject, all other feature sets, with all classifiers, obtained very high accuracy, with Ad Infinitum being the best classifier. The best performing feature sets were All and Body.

Table 20: Accuracy (%) using various feature sets in the supervised experiment

Classifier	Subject	Body	All
Ad Infinitum	92.8	98.6	99.7
DT	90	97.4	99.4
SVM	92.6	95.6	99.2
NB	74.9	95.4	95.8

5. Experimental results and discussion

The application of data mining to the task of automatic threatening e-mail detection is done and experiments were carried out on an email corpus. In order to conduct an experiment, different sets of emails (TCETHN1 and TCETHN2) are used. The system was trained with the use of Ad Infinitum algorithm. When the training process was finished, the best quality rules were taken as the final classification rules.

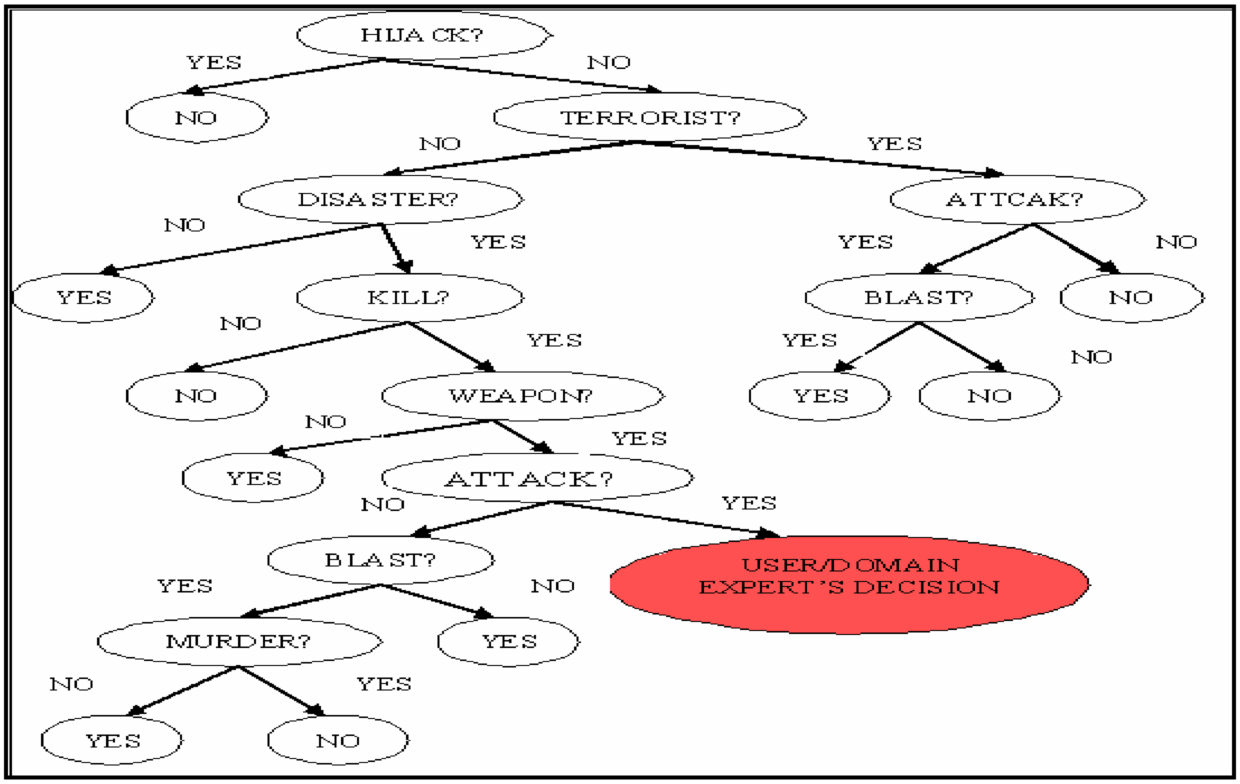


Figure (7): Proposed Ad Infinitum Tree

Generated rules:

1. If hijack=yes then threatening=no
2. If hijack=no & terrorist=yes & attack=yes & blast=yes then threatening=yes
3. If hijack=no & terrorist=yes & attack=yes & blast=no then threatening=no
4. If hijack=no & terrorist=yes & attack=no then threatening=no
5. **If hijack=no & terrorist=no & disaster=yes & kill=yes & weapon=yes & attack=yes then threatening="User/Domain expert's decision"**
6. If hijack=no & terrorist=no & disaster=yes & kill=yes & weapon=yes & attack=no & blast=yes & murder=yes then threatening=no
7. If hijack=no & terrorist=no & disaster=yes & kill=yes & weapon=yes & attack=no & blast=yes & murder=no then threatening=yes
8. If hijack=no & terrorist=no & disaster=yes & kill=yes & weapon=yes & attack=no & blast=no then threatening=yes
9. If hijack=no & terrorist=no & disaster=yes & kill=yes & weapon=no then threatening =yes
10. If hijack=no & terrorist=no & disaster=yes & kill=no then threatening=no
11. If hijack=no & terrorist=no & disaster=no then threatening=yes

To evaluate performance we calculate accuracy (A), recall (R), precision (P) and F1 measure. Accuracy is the most commonly used measure in machine learning. Precision, recall and their combination, the F1 measure, are the most popular criteria used in text

categorization. In the multi class task of general mail classification, macro-averaging [27] was used – precision, recall and the F1 measure were first calculated for each class and the results were then averaged.In the threatening e-mail detection experiment we calculated accuracy, Threatening recall (TR), Threatening precision (TP) and Threatening F1 measure (TF1). TR is the proportion of threatening e-mails in the test set that are classified as threatening, TP is the proportion of e-mails in the test data classified as threatening that are truly threatening. Discarding a legitimate e-mail is of greatest concern to most users than classifying a threatening message as legitimate. This means that high TP is particularly important.
The classification results of TCETHN2 and TCETHN1 are given in Table 21.By comparing Accuracy and TF1 scores, we can see that overall Ad-Infinitum is the best classifier, obtaining the most consistent results on the two corpora for both IG and TFV.When IG was used as feature selector ,Ad Infinitum achieved the best accuracy for both TCETHN2 and TCETHRN1.When TFV was used as feature selector, the best accuracy for both corpora was achieved by Ad Infinitum and DT.The second best classifier is Decision Tree. The worst classifier is NB failing behind the winner in terms of TF1.We should keep in mind that for threatening e-mail detection TP is more important than TR.

Table 21: Performance of AD-INFINITUM, DT, SVM and NB on threatening e-mail detection. (E-mail corpus in Table 3 is taken)

	A	TP	TR	TF1
TCETHREATENING 2				
AD-INFINITUM-IG	98.75	97.88	96.72	97.29
DT-IG	97.28	94.44	94.04	94.24
SVM-IG	95.65	97.78	80.89	88.54
NB-IG	94.93	91.64	82.20	86.66
AD-INFINITUM-TFV	99.20	97.10	96.72	96.91
DT-TFV	98.80	96.30	96.10	96.20
SVM-TFV	98.50	94.10	92.80	93.45
NB-TFV	94.41	90.54	79.89	84.88
TCE THREATENING 1				
AD-INFINITUM-IG	95.50	94.15	93.82	93.98
DT-IG	92.45	90.40	92.42	91.40
SVM-IG	92.10	94.05	87.40	90.60
NB-IG	86.35	93.83	74.20	82.87
AD-INFINITUM-TFV	99.10	96.75	96.45	96.59
DT-TFV	98.30	96.25	96.05	96.15
SVM-TFV	98.10	94.50	91.90	93.13
NB-TFV	89.35	96.20	78.40	86.39

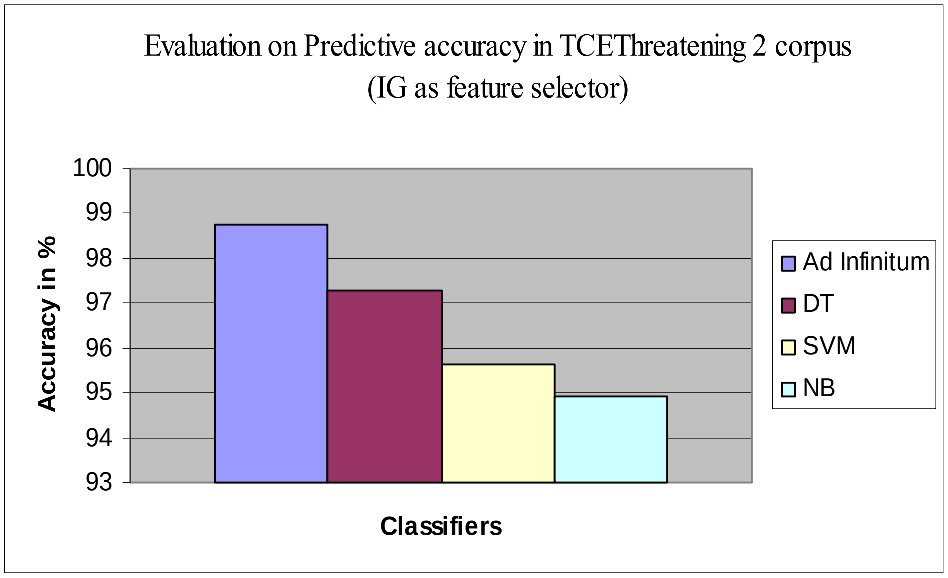


Figure (8): Classifiers performance on TCE THREATENING 2 Corpus

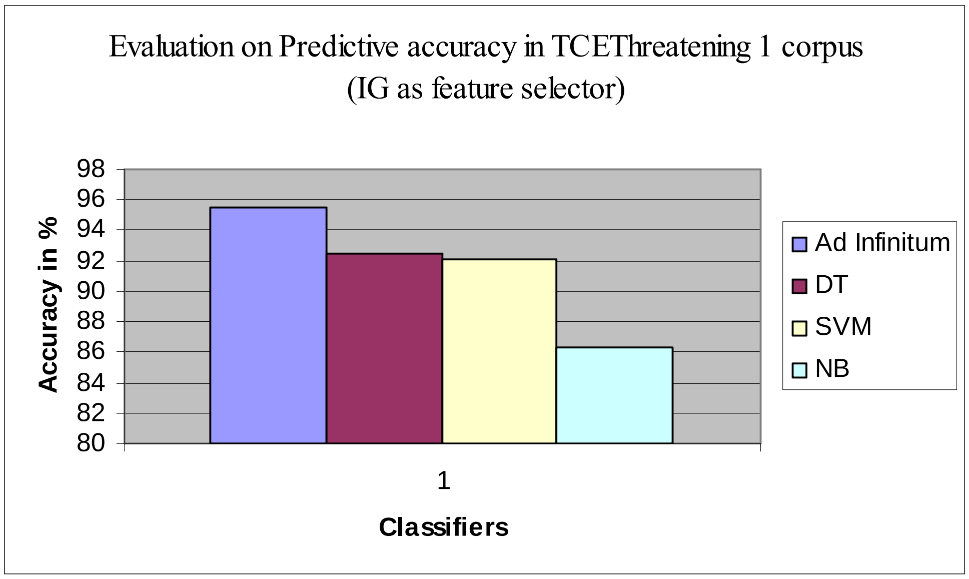


Figure (9): Classifiers performance on TCETHREATENING 1 Corpus

Again we tested the portability across corpora using TCETHREATENING1 and TCETHREATENING2, see Table 22. We trained a Ad Infinitum classifier on TCETHREATENING2 and test it on TCETHREATENING1 (and vice versa), using the features selected from the training corpora. Typical confusion matrices are given in Table

23. The good news is that in both cases threatening e-mails are relatively well recognized (TR=86.0 and 85.2) which can be explained with the common characteristics of threatening e-mails across the two corpora.

Table 22: Portability across corpora using Ad Infinitum (IG as feature selector, similar results for TFV)

	Training set	Testing set	A	TR	TP	TF1
(a)	TCETHN 2	TCETHN 1	58.59	86.0	34.99	49.73
(b)	TCETHN 1	TCETHN 2	12.41	85.2	11.33	20.0

Table 23: Confusion matrices

#Assigned as	(a) Threatening	Normal	(b)Threatening	Normal
Threatening	430	70	426	74
Normal	799	800	3335	58

6. Conclusion and future work

E-mail is an important vehicle for communication .It is one possible source of data from which the potential problem can be detected. E-mail classification is highly important for the domain such as detection of threatening e-mail which does not contain any historical data. From the experimentation done by us, we infer that arbitrary decision making of the decision tree induction algorithm leads to misclassification of correct instances which is overcome by the proposed algorithm. The proposed work will be helpful for identifying the threatening e-mail and also assist the investigators to get the information in time to take effective actions to reduce the criminal activities. A problem we faced when trying to test out new ideas dealing with e-mail systems was an inherent limitation of the available data, because we only have access to our own data, our results and experiments no doubt reflect some bias. Much of the work published in the e-mail classification domain also suffers from the fact that it tries to reach general conclusion using very small data sets collected on a local scale.

In this paper we consider supervised e-mail classification for the task of threaten e-mail detection and investigate the performance of four algorithms: Ad Infinitum, DT, SVM and NB. Our findings can be summarized as follows:

• In supervised learning setting, we have shown that Ad Infinitum is a promising approach for automatic detection of threatening e-mail. It outperforms in terms of classification performance well-established algorithms such as DT(s), SVM and NB being also more complex than Ad Infinitum. Ad Infinitum is easy to tune, and runs very efficiently on large datasets with high number of features, which makes it very attractive for text categorization.

• We have implemented a feature selector IG and found that it performs better than the popular and computationally more expensive TFV.

• We compared the performance of a number of algorithms on the TCETHREATENING 1 and TCETHREATENING 2 e-mail corpora which is a mixture of our own e-mail data set with benchmark spam filtering corpora LingSpam and PU1.

References

[1] S. Appavu alias Balamurugan and R. Rajaram, "Association rule mining for Suspicious Email Detection: A Data mining approach", *Proceedings of the IEEE International Conference on Intelligence and Security Informatics, New Jersey, USA,* pp. 316 – 323, 2007.

[2] S. Appavu alias Balamurugan and R.Rajaram, "Data mining Techniques for Suspicious Email Detection: A Comparative Study", *Proceedings of the European Conference on Data mining,* Portugal, 2007.

[3] S.Appavu alias Balamurugan and R.Rajaram, "Suspicious Email Detection via Decision Tree: A Data mining Approach", *Journal of Computing and Information Technology –CIT 15, PP.161-169, 2007.*

[4] I.Androutsopoulos, J. Koutsias, V.Chandrinos and C. Spyropoulos, "An experimental comparison of naive Bayesian and keyword-based anti-spam filtering with personal e-mail messages", *in: Proc. ACM SIGIR 2000,* pp. 160–167.

[5] I.Androutsopoulos , G.Palioras ,V. Karkaletsis , G. Sakkis , C. Spyropoulos and P. Stamatopoulos , "Learning to filter spam e-mail: A comparison of a Naive Bayesian and memory-based approach", *in: Proc. 4th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD) 2000,* pp. 1–13.

[6] X.Carreras and L. Marquez, **"**Boosting trees for anti-spam email filtering", *in: Proc. 4th International Conference on Recent Advances in Natural Language Processing,* 2001.

[7] W. Cohen, "Learning rules that classify e-mail", *in: Proc. AAAI Symposium on Machine Learning in Information Access,* 1996, pp. 18–25.

[8] M.Dash and H.Liu, "Feature Selection for Classification", *Intelligent Data Analysis, vol.1, no.3, 1997.*

[9] N.Ducheneaut and L.Watts, "In search of coherence: a review of e-mail research", *Human-Computer Interaction 20* (2004) 11–48.

[10] P.Graham, "Better Bayesian filtering", 2003, <http://www.paulgraham.com/better.html>.

[11] Ian H.Witten and Eibe Frank, "Data Mining, Practical Machine Learning Tools and Techniques".

[12] Jiawei Han, Micheline Kamber, "Data Mining Concepts and Techniques", *Morgan Kaufmann Publishers.*

[13] P.S.Keila and D.B.Skillicorn, "Detecting unusual and deceptive communication in e-mail", *Technical reports, June 2005.*

[14] LingSpam and PU1 datasets, <http//www.aueb.gr/users/ion/publications.html>.

[15] C.Manning and H. Schutze, "Foundations of Statistical Natural Language Processing"*, MIT Press,* 1999.

[16] Message Labs, 2005, <http://www.messagelabs.com/Threat_Watch>.

[17] P.Pantel and D. Lin, "SpamCop: A spam classification and organization program", *in: Proc. AAAI Workshop on Learning for Text Categorization* 1998.

[18] J.Provost, "Naive-Bayes vs. rule learning in classification of e-mail", *University of Texas at Austin,* 1999
.
[19] R.Quinlan, "C4.5: Programs for Machine Learning", Morgan Kaufman, 1993.

[20] J. R. Quinlan," Induction of decision trees. Machine Learning", 1:1–106, 1986.

[21] J.R.Quinlan, (1983). "Learning efficient classification procedures and their application to chess end games (pp. 463-482). In Michalski, Carbonell & Mitchell (Eds.), *Machine learning: An artificial intelligence approach*. San Mateo, CA: Morgan Kaufmann.

[22] J.Rennie, "An application of machine learning to e-mail filtering"*, in: Proc. KDD-2000 Text Mining Workshop,* 2000.

[23] G.Rios and H.Zha, "Exploring support vector machines and random forests for spam detection", *in: Proc. First International Conference on Email and Anti Spam (CEAS),* 2004.

[24] M. Sahami, S.Dumais, D. Heckerman and E. Horvitz "A Bayesian approach to filtering junk e-mail", *in: Proc. AAAI Workshop on Learning for Text Categorization,* 1998.

[25] G.Sakkis, I.Androutsopoulos, G.Palioras, V.Karkaletsis, C. Spyropoulos and P.Stamatopoulos, "Stacking classifiers for anti-spam filtering of e-mail", *in: Proc. 6th Conference on Empirical Methods in Natural Language Processing,* 2001, pp. 44–50.

[26] J.C.Schlimmer, and D. Fisher (1986a), "A case study of incremental concept induction", *Proceedings of the Fifth National Conference on Artificial Intelligence* (pp. 496-501). Philadelpha, PA: Morgan Kaufmann.

[27] F.Sebastiani, "Machine learning in automated text categorization", ACM Computing Surveys 34 (2002) 1–47.

[28] R.Segal, M. Kephart and MailCat, "An intelligent assistant for organizing e-mail", *in: Proc. Third International Conference on Autonomous Agents,* 1999.

[29] P.N.Tan, M. Steinbach and V.Kumar, "Introduction to Data Mining", Addison Wesley, 2005.

[30] P.E.Utgoff, (1988), "ID5: An incremental ID3", *Proceedings of the Fifth International Conference on Machine Learning* (pp. 107-120). Ann Arbor, MI: Morgan Kaufman.

[31] V.Vapnik, "Statistical Learning Theory", Wiley, 1998.

[32] Weka, http//www.cs.waikato.ac.nz/ml/weka.

[33] S.Whittaker, V.Bellotti, and P. Moody, "Introduction to this special issue on revisiting and reinventing e-mail", *Human-Computer Interaction 20* (2005) 1–9.

[34] Y.Yang and J. Pedersen, "A comparative study on feature selection in text categorization", *in: Proc. 4th International Conference on Machine Learning,* 1997.

S.Appavu alias Balamurugan was born in Tamil nadu, India, in 1979.He received B.E. degree in Electronics and Communication Engineering from the Madurai Kamaraj University, Madurai, in 2001.He also received the M.E in Computer Science from the University of Madras in 2003.He is currently pursuing Ph.D in Information and Communication Engineering from Anna university, Chennai, India. Now he is a lecturer in the Department of Information Technology, Thiagarajar College of Engineering, Madurai, India. His main interests are Data mining, Classification, and Performance issues in Classification algorithms.

Dr. R.Rajaram, Dean of CSE/IT, Thiagarajar College of Engineering, has BE degree in Electrical and Electronics Engineering from Madras University in 1966.He secured the M Tech degree in Electrical Power Systems Engineering in 1971 from IIT Kharagpur, and the Ph.D. degree on Energy Optimization from Madurai Kamaraj University in 1979. He and his research scholars have published/presented more that 45 research papers in Journals and Conferences. His current areas of interest are Mobile Agents, Cryptography and Data mining. He has published more than 13 text books on Computer languages and Basic Communications.He has served the Makerere University at Uganda during 1977-1978 and University of Mosul during 1980-1981. He secured two best technical paper awards from the Institution of Engineers India and one from Indian Society for Technical Education.

M.MuthuPandian is currently doing final year B.Tech Information Technology at Thiagarajar College of Engineering, Madurai, India. His research interests include study on various classifications and clustering algorithms of Data mining and improving its performance by handling various demerits. He is a member of Computer Society of India (CSI) and Indian Society for Technical Education (ISTE).

G.Athiappan is currently doing final year B.Tech Information Technology at Thiagarajar College of Engineering, Madurai, India. His research interests include Data mining and Information Security. He is a member of Computer Society of India (CSI) and Indian Society for Technical Education (ISTE).