iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 3 **Issue:** I **Month of publication:** January 2015

DOI:

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

ISSN: 2321-9653
International Journal for Research in Applied Science & Engineering Technology (IJRASET)

Comparison and Analysis of Spam Detection Algorithms

Sakshi Hooda[1], Aakanksha[2], Varsha Kansal[3], Swati Kadian[4]
[1]Assistant. Professor, [2-4]Student, Department of Computer Science & Technology
Maharaja Surajmal Institute of Technology
Janakpuri, New Delhi, India

Abstract— In this e-world, most of the transactions and business is taking place through e-mails. Nowadays, email becomes a powerful tool for communication as it saves a lot of time and cost. But, due to social networks and advertisers, most of the emails contain unwanted information called spam. Even though lot of algorithms has been developed for email spam classification, still none of the algorithms produces 100% accuracy in classifying spam emails. Current server-side anti-spam filters are made up of several modules aimed at detecting different features of spam e-mails. In particular, text categorization techniques have been investigated by researchers for the design of modules for the analysis of the semantic content of e-mails, due to their potentially higher generalization capability with respect to manually derived classification rules used in current server-side filters. Our research paper consists of comprehensive study of spam detection algorithms under the category of content based filtering. The implemented results have been benchmarked to analyze how accurately they have been classified into their original categories of spam and ham. Further, a new dynamic aspect has been added which includes run-time implementation of Naive Bayes and J48 Tree algorithm on the data which we fed from the mail server dynamically for more efficient results.

Keywords— Spam, Naive-Bayes, KNN, Black list, White list, J48 Decision Tree, Naïve Bayes Multinomial.

I. INTRODUCTION

Due to the intensive use of internet, email has become one of the fastest and most economical mode of communication. This enables internet user to easily transfer information from anywhere in the world in a fraction of second. However, the increase of email users have resulted in the dramatic increase of spam emails during the past few years. E-mail spam, also known as junk e-mail or unsolicited bulk e-mail (UBE), is a subset of spam that delivers nearly identical messages to numerous recipients by e-mail. Definitions of spam usually include the aspects that e-mail is unsolicited and sent in bulk. E-mail spam has steadily grown since the early 1990s. Botnets, networks of virus-infected computers, are used to send about 80% of spam. Spammers collect e-mail addresses from chat rooms, websites, customer lists, newsgroups, and viruses which harvest users' address books, and are sold to other spammers. Since the cost of the spam is borne mostly by the recipient, many individual and business people send bulk messages in the form of spam. The voluminous of spam emails a strain the Information Technology based organizations and creates billions of dollars lose in terms of productivity. In recent years, spam emails lands up into a serious security threat, and act as a prime medium for phishing of sensitive information. Addition to this, it also spread malicious software to various users. Therefore, email classification becomes an important area to automatically classify original emails from spam emails. Automatic email spam classification contains more challenges because of unstructured information, more number of features and large number of documents. As the usage increases all of these features may adversely affect performance in terms of quality and speed. Many recent algorithms use only relevant features for classification. Even though more number of classification techniques has been developed for spam classification, still 100% accuracy of predicting the spam email is questionable. So Identification of best spam algorithm itself became a tedious task because of features and drawbacks of every algorithm against each other.

Daily Spam emails sent: 12.4billion
Daily Spam received per person: 6
Annual Spam received per person: 2,200
Spam cost to all non-corporate: $255 million Internet users
Spam cost to all U.S Corporation in 2002: $8.9 billion
Email address changes due to spam: 16%
Annual Spam in 1,000 employee's company: 2.1 million
Users who reply to Spam email: 28%

Fig 1: Statistics of spam mails

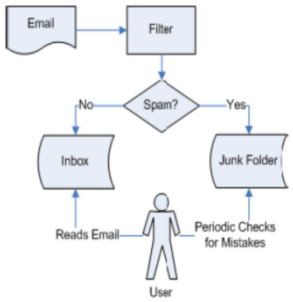


Fig 2: Interpretation of SPAM Filter

The basic format of Electronic-mail generally consists of the following sections:
Header section includes the sender email address, the receiver email address, the subject of the email.
The Content of the email includes the main body consisting of text, images and other multimedia data

In content based spam filtering, the main focus is on classifying the email as spam or as ham, based on the data that is present in the body or the content of the mail. However, the header section is ignored in the case of content based spam filtering. There are number of techniques such as Bayesian Filtering, Gary Robinson technique, KNN classifier, Multilayer Perceptron. Combining function based on Fisher-Robinson Inverse Chi-Square Function is available which can be used for content based filtering. This research work comprises of the analytical study of various spam detection algorithms based on content filtering such as Fisher-Robinson Inverse Chi Square function, Bayesian classifiers, AdaBoost algorithm and KNN algorithms. The algorithms have been implemented; the results were studied to draw a relative comparison on the effectiveness of a technique to identify the most accurate one. Each technique is demonstrated in the following sections with their implemented result. The paper is concluded with the benchmarking of the techniques.

II. RELEVANT WORK

A. Content Based Filtering

In content Based Filtering, the spam techniques are applied on the name of the sender, header, content of mail or the content of the attachment. Content Based Filtering uses the concept of rules to classify mails as spam or ham. These rules may be applied on “To:”, “From:” or “Subject:” field of the header or the body of the mail.
Different rules may vary from checking the font size of the text to checking whether the mail arrived from an address in the person’s address book or searching the subject line for words like ‘free’, ‘sale’ and so on.

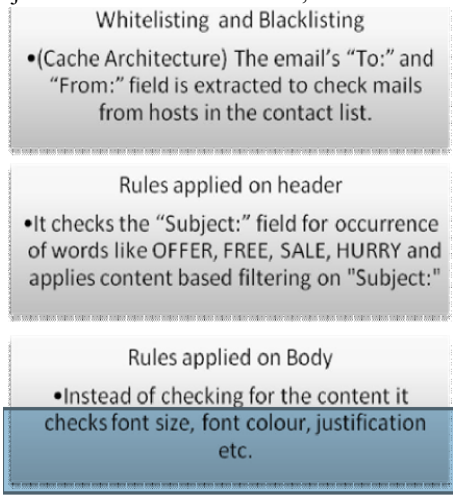


Fig: Flowchart of Content Based Filtering

After creating both the lists, when an email arrives, the 'To' and 'From' field is extracted from its subject to check if it is in the Black List or the white List. The main rule applied here is that if the sender is from the Black List, then it will be considered as a spam mail. The concept is illustrated in the fig below:

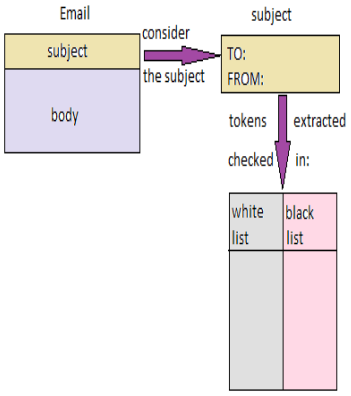


Fig: How it works

Some of the rules applied on header are as under:

Mailing: This rule is applied to the 'From' and 'To' section of the email, i.e. if the sender or receiver address corresponds to an address in the Black List or White List, corresponding action can be taken.
Pattern: When the header (mailing list or subject) depicts some pattern like ankit*@gmail.com then the mail is categorized as spam.
Content Based Filtering on subject: Here the normal content based filtering is applied to the subject line itself to find whether it contains words classified as spam or ham.

Rules applied on the body:
Font Size:

Generally, spam mails consist of large fonts. So body is checked for words with higher font size, if the frequency is higher than a preset threshold, email can be declared as a spam.

Font Color:

Spam mails usually comprise of large variation in the color of the text to attract the receiver. Again, the body can be scanned for words with different colors and frequency can be checked to be classified as spam or ham.

Some more rules that can be applied on the header are as follows:

- If the body consists entirely of images, generally a spam message.
- 'From:' containing empty name, can be classified as a spam mail,
- The 'From:' field of the subject starts with many numbers, is generally assumed to be a spam message with machine created email address.
- From: has no local-part before @ sign.
- Message-ID contains multiple '@' characters.
- The subject contains gaps between text.

B. KNN Algorithm

K nearest neighbour is a simple algorithm that stores all available cases and classifies new cases based on a similarity measure (e.g., distance functions).KNN has been used in statistical estimation and pattern recognition already in the beginning of 1970's as a non-parametric technique.

C. Algorithm

A case is classified by a majority vote of its neighbours, with the case being assigned to the class most common amongst its K nearest neighbours measured by a distance function. If K=1, then the case is simply assigned to the class of its nearest neighbour.

Distance functions

Euclidean $\sqrt{\sum_{i=1}^{k}(x_i - y_i)^2}$

Manhattan $\sum_{i=1}^{k}|x_i - y_i|$

Minkowski $\left(\sum_{i=1}^{k}(|x_i - y_i|)^q\right)^{1/q}$

It should also be noted that all three distance measures are only valid for continuous variables. In the instance of categorical variables the Hamming Distance must be used. It also brings up the issue of standardization of the numerical variables between 0 and 1, when there is a mixture of numerical and categorical variables in the dataset.

Hamming Distance

$$D_H = \sum_{i=1}^{k}|x_i - y_i|$$

$$x = y \Rightarrow D = 0$$

$$x \neq y \Rightarrow D = 1$$

X	Y	Distance
Male	Male	0
Male	Female	1

Choosing the optimal value for K is best done by first inspecting the data. In general, a large value of K is more precise as it reduces the overall noise but there is no guarantee. Cross-validation is another way to retrospectively determine a good value of K. Historically, the optimal K for most of the datasets is between 3-10. That produces much better results than 1NN.

III. IMPLEMENTED WORK

A. *Naïve Bayes*

The Bayesian approach is fundamentally an important DM technique. The Bayes classifier can provably achieve the optimal result when the probability distribution is given. Bayesian method is based on the probability theory. A Bayesian filter learns a spam classifier from a set of manually classified examples of spam and legitimate (or *ham*) messages i.e. Training collection. This training collection is taken as the input for the learning process; this consists of the following steps:

1) *Pre-processing*: The pre-processing is the deletion of irrelevant elements (e.g. HTML), and selection of the segments suitable for processing (e.g. headers, body).
2) *Tokenization:* This is the process of dividing the message into semantically coherent segments (e.g. words, other character strings).
3) *Representation:* The representation is the conversion of a message into an attribute-value pairs' vector [10], where the attributes are the previously defined tokens, and their values can be binary, (relative) frequencies, etc.
4) *Selection:* The selection process includes the Statistical deletion of less predictive attributes (using e.g. quality metrics like Information Gain).
5) *Learning:* The learning phase automatically building a classification model (the classifier) from the collection of messages. The shape of the classifier depends on the learning algorithm used, ranging from decision trees (C4.5), or classification rules (Ripper), to statistical linear models (Support Vector Machines, Winnow), neural networks, genetic algorithms, etc.

B. *Naïve Bayesian Classifiers*

Naive Bayes can often outperform more with sophisticated classification methods. The following example shows the Naïve

Bayes classifier demonstration. Here, the objects can be classified as either Black or WHITE. The task is to classify new cases as they arrive (i.e., decide to which class label they belong).

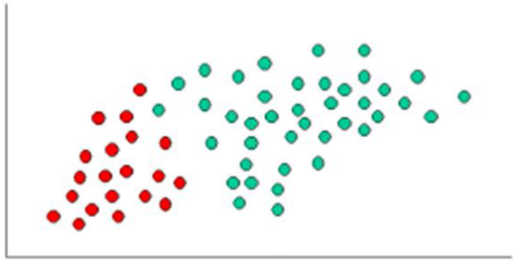


Fig 3: Objects are classified to GREEN or RED

The calculation of the priors is (i.e. the probability of the object among all objects) based on the previous knowledge. Therefore:
Prior probability for GREEN α No. Of green objects/Total no. Of objects
Prior probability for RED α No. Of red objects/Total no. Of objects

There is a total of 60 objects, 40 of which are GREEN and 20 RED, the prior probabilities for class membership are:
Prior probability for GREEN α 40/60
Prior probability for RED α 20/60

Having formulated the prior probability, the system is ready to classify a new object (WHITE circle in Figure 10). As the objects are well clustered, assume that the more GREEN (or RED) objects in the vicinity of X, more likely that the new cases belong to that particular colour. Then a circle is drawn around X to measure this likelihood, which encompasses a number (to be chosen a priori) of points irrespective of their class labels. Then the number of points in the circle is calculated.

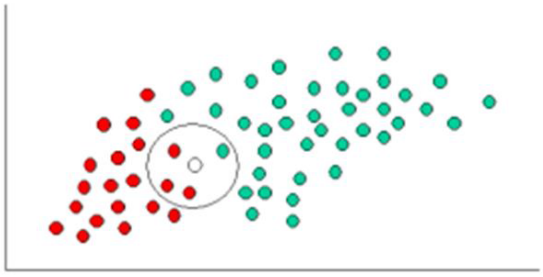


Fig 4: Classify the WHITE circle

Then the likelihood is calculated as follows:
Likelihood of X given GREEN α Total no. Of GREEN in the vicinity of X/Total no. Of GREEN cases
Likelihood of X given RED α Total no. Of RED in the vicinity of X/ Total no. Of RED cases

In Figure 3, it is clear that Likelihood of X given RED is larger than Likelihood of X given GREEN, as the circle encompasses 1 GREEN object and 3 RED ones. Thus:
Probability of X given GREEN α 1/40
Probability of X given RED α 3/40

In the Bayesian analysis, the final classification is produced by combining both sources of information (i.e. the prior and the likelihood) to form a posterior probability using Bayes Rule.

Posterior probability of X being GREEN α Prior probability of GREEN X Likelihood of X given GREEN= 4/6 X 1/40 = 1/60

Posterior probability of X being RED αPrior probability of RED X Likelihood of X given RED=2/6 X 3/40 = 1/40
Finally, classify X as RED since its class membership achieves the largest posterior probability.

C. Naïve Bayes Multinomial

In contrast to the multi-variant Bernoulli event model, the multinomial model captures word frequency information in documents. Consider, for example, the occurrence of numbers in the Reuters newswire articles; our tokenization maps all strings of digits to a common token. Since every news article is dated, and thus has a number, the number token in the multi-variant Bernoulli event model is uninformative. However, news articles about earnings tend to have a lot of numbers compared to general news articles. Thus, capturing frequency information of this token can help classification.
In the multinomial model, a document is an ordered sequence of word events, drawn from the same vocabulary *V*. We assume that the lengths of documents are independent of class. We again make a similar naive Bayes assumption: that the probability of each word event in a document is independent of the word's context and position in the document. Thus, each document *di* is drawn from a multinomial distribution of words with as many independent trials as the length of *di*. This yields the familiar "bag of words" representation for documents. Define *Nit* to be the count of the number of times word *wt* occurs in document *di*.

Term Frequency**:**
An alternative approach to characterize text documents — rather than binary values — is the *term frequency (tf(t, d))*. The term frequency is typically defined as the number of times a given term *t* (i.e., word or token) appears in a document *d* (this approach is sometimes also called *raw frequency*). In practice, the term frequency is often normalized by dividing the raw term frequency by the document length.

normalized term frequency=tf(t,d)nd

tf(t,d): Raw term frequency (the count of term t in document d).
nd: The total number of terms in document d.

The term frequencies can then be used to compute the maximum-likelihood estimate based on the training data to estimate the class-conditional probabilities in the multinomial model:

$P\hat{}(xi|\omega j)=\sum tf(xi,d\in\omega j)+\alpha\sum Nd\in\omega j+\alpha\cdot V$

where

1) xi: A word from the feature vector x of a particular sample.
2) $\sum tf(xi,d\in\omega j)$: The sum of raw term frequencies of word xi from all documents in the training sample that belong to class ωj.
3) $\sum Nd\in\omega j$: The sum of all term frequencies in the training dataset for class ωj.
4) α: An additive smoothing parameter (α=1 for Laplace smoothing).
5) V: The size of the vocabulary (number of different words in the training set).

The class-conditional probability of encountering the text x can be calculated as the product from the likelihoods of the individual words (under the *naive* assumption of conditional independence).

$P(x||\omega j)=P(x1|\omega j)\cdot P(x2|\omega j)\cdot\ldots\cdot P(xn|\omega j)=\prod i=1mP(xi|\omega j)$

1) *Term Frequency-Inverse Document Frequency (Tf-idf):* The term frequency - inverse document frequency (Tf-idf) is another alternative for characterizing text documents. It can be understood as a weighted term frequency, which is especially useful if stop words have not been removed from the text corpus. The Tf-idf approach assumes that the importance of a word is inversely proportional to how often it occurs across all documents. Although Tf-idf is most commonly used to rank documents by relevance in different text mining tasks, such as page ranking by search engines, it can also be applied to text classification via naive Bayes.

Tf-idf=tfn(t,d)·idf(t) (39)

Let tfn(d,f) be the normalized term frequency, and idf, the inverse document frequency, which can be calculated as follows:

idf(t)=log(ndnd(t)), (40)

where
nd: The total number of documents.
nd(t): The number of documents that contain the term t.

2) Technical Specifications

a) java.lang.Object
b) weka.classifiers.AbstractClassifier
c) weka.classifiers.bayes.NaiveBayesMultinomial

3) All Implemented Interfaces

java.io.Serializable,
java.lang.Cloneable, Classifier, CapabilitiesHandler, OptionHandler, RevisionHandler, TechnicalInformationHandler, Weighted InstancesHandler

4) Direct Known Subclasses

NaiveBayesMultinomialUpdateable
Public class NaiveBayesMultinomial
extends AbstractClassifier
implementsWeightedInstancesHandler, TechnicalInformationHandler.

Class for building and using a multinomial Naive Bayes classifier. The core equation for this classifier:
P[Ci|D] = (P[D|Ci] x P[Ci]) / P[D] (Bayes rule)
where Ci is class i and D is a document.

D. J48

C4.5 algorithm generates decision trees which is an extension of Quinlan's ID3 algorithm. Such decision trees are used for classification and hence "statistical classifier is one more name for this C4.5 algorithm.

1) Algorithm: C4.5 builds decision trees from a set of training data in the same way as ID3, using the concept of information entropy. The training data is a set of already classified samples. Each sample consists of a p-dimensional vector, where it represents attributes or features of the sample, as well as the class in which it falls.

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 (difference in entropy). The attribute with the highest normalized information gain is chosen to make the decision. The C4.5 algorithm then recurs on the smaller subsists.

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.

2) Pseudo Code: In pseudo code, the general algorithm for building decision trees is:

a) Check for base cases
b) For each attribute *a*, find the normalized information gain ratio from splitting on *a*
c) Let *a_best* be the attribute with the highest normalized information gain
d) Create a decision *node* that splits on *a_best*
e) Recurse on the sublists obtained by splitting on *a_best*, and add those nodes as children of *node*

3) *Implementation*: J48 is an open source Java implementation of the C4.5 algorithm in the weka data mining tool.

4) *Technical Specifications*

a) java.lang.Object
b) weka.classifiers.AbstractClassifier
c) weka.classifiers.trees.J48

5) *All Implemented Interfaces*

java.io.Serializable,
java.lang.Cloneable, Classifier, Sourcable, AdditionalMeasureProducer, CapabilitiesHandler, Drawable, Matchable, OptionHandler, PartitionGenerator, RevisionHandler, Summarizable,TechnicalInformationHandler,WeightedInstancesHandler.

6) *Syntax*

public class J48
extends AbstractClassifier
implements OptionHandler, Drawable, Matchable, Sourcable, WeightedInstancesHandler, Summarizable,
AdditionalMeasureProducer, TechnicalInformationHandler, PartitionGenerator

IV. BENCHMARKING OF TECHNIQUES

The major techniques illustrated in the previous sections have been implemented and the results are shown in the table. The mails are categorized as:
Spam mails that were incorrectly classified as ham mails.
Spam mails that were correctly classified as spam mails.
Ham mails that were correctly classified as ham mails.
mails that were incorrectly classified as spam mails.

A. Tables

TABLE I. COMPARISON OF IMPLEMENTED SPAM FILTERING ALGORITHMS

Algorithm	***Correctly Classified (No. of Instances)***	***Incorrectly Classified (No. of Instances)***	***Success Percentage (No. of Instances in %)***	***Dataset Size (No.of Instances)***
Naïve Bayes	32	8	80%	40
J48 Decision Tree	40	0	100%	40
Naïve Bayes Multinomial	40	0	100%	40

TABLE II. COMPARISON OF PERFORMANCE

Parameters	***J48 Decision Tree***	***Naïve Bayes Multinomial***
Mean absolute error	0%	0.0027%
Root mean squared error	0%	0.0192%
Relative absolute error	0%	0.7853%
Root relative squared error	0%	4.6943%
Coverage of cases (0.95 level)	100%	100%

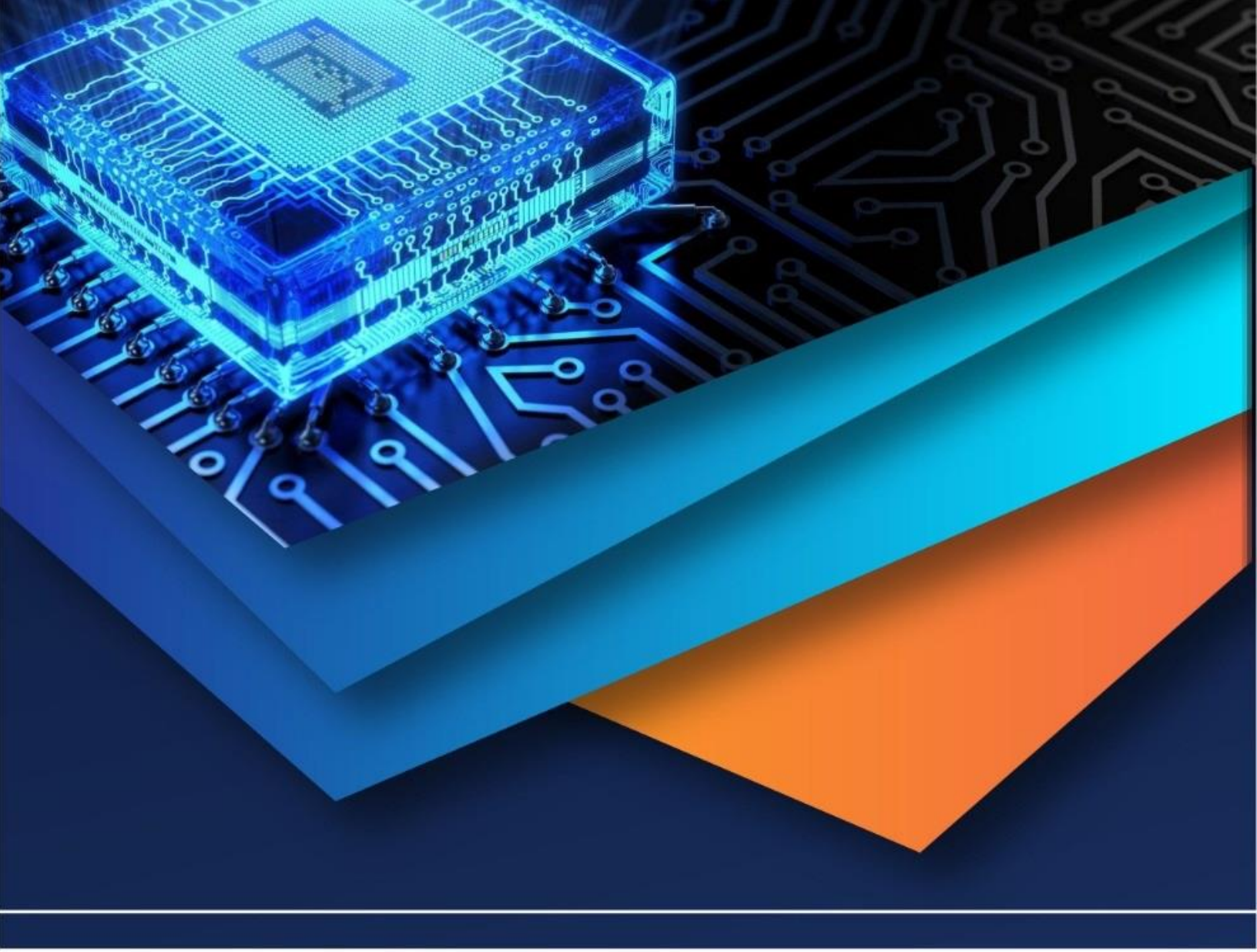
After comparing the results we reached the conclusion that **Naïve Bayes Multinomial & J48 Decision Tree** approach were equally efficient but out of them **J48 Decision Tree** approach was the best due to zero percent error calculated above.

V. CONCLUSION

Various techniques of spam filtering are studied and analyzed. The implemented results are mentioned in the tables shown above. The most efficient technique according to our research is **J48 Decision Tree** if the technique is to be applied on *static* data, but if our run time type data is *dynamic* then **Naïve Bayes** is the best.

REFERENCES

[1] E. Horvitz- A Bayesian Approach to Filtering Junk E-Mail.
[2] Khorsi A.- "An overview of Content-Based Spam Filtering Techniques".
[3] Robinson, G. Gary Robinson's Rants. Available: http://www.garyrobinson.net.
[4] Sudipto Guha, Adam Meyerson, Nina Mishra, Rajeev Motwani and Liadan O'Callaghan, "Clustering Data Streams," IEEE Trans.s on Knowledge & Data Engg., 2003.
[5] Micheline Kamber, "Data Mining Concepts and Techniques", Second Edition.
[6] Roychowdhury, "Personal Email networks: an effective anti-spam tool".
[7] Sahami, Dumais, "A Bayesian Approach to Filtering Junk E-Mail"2008.
[8] Golbeck, "Reputation Network Analysis for Email Filtering".
[9] Sculley D, Wachman G., "Spam Filtering Using Inexact String Matching in Explicit Feature Space with On-Line Linear Classifiers", Text Retrieval Conference, pp. 1, 2006.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY