Paradigm Academic Press
Innovation in Science and Technology
ISSN 2788-7030
MAY. 2024 VOL.3, NO.3

Machine Learning-Based Email Spam Filter

Jiuyue Zhang[1]

[1] College of Engineering, Design and Physical Sciences, Brunel University, United Kingdom

Correspondence: Jiuyue Zhang, College of Engineering, Design and Physical Sciences, Brunel University, United Kingdom.

doi:10.56397/IST.2024.05.05

Abstract

The surge in email usage has been accompanied by an unwelcome increase in spam, posing challenges for individuals and organizations. This study addresses the urgent need for efficient spam detection by evaluating the performance of three widely-used machine learning classifiers: Naive Bayes, Random Forest, and Logistic Regression. Our approach includes comprehensive data preparation, feature extraction, model training, and rigorous performance evaluation using metrics such as Accuracy, Precision, Recall, and F1-Score. Furthermore, we analyze the trade-off between accuracy and computational efficiency, essential for real-time spam detection systems. A user-friendly interface developed with Flask showcases the practical application of our findings. The Random Forest Classifier outperforms its counterparts, proving to be the most effective in accurately classifying emails and maintaining a balance between sensitivity and specificity. The study's implications highlight the potential for sophisticated machine learning-based spam filters to enhance email security.

Keywords: spam detection, email classification, machine learning, feature extraction

1. Introduction

Spam messages pose a significant challenge that must be addressed or eradicated from the current situation as they can negatively impact general email communication with harmful consequences. Consequently, there is an urgent necessity to identify such harmful spam emails and mitigate their effects in today's digital landscape. Historically, various methodologies have been employed for mail spam detection, including blacklists, whitelists, content-based data filters, and machine learning-based identification strategies. However, each of these methods has its limitations, and there is currently no solution that guarantees one hundred percent spam detection for contemporary email filtering systems. The email blacklist mechanism functions as a repository capable of storing multiple spam email addresses, aiding users in identifying whether incoming messages originate from these flagged addresses. If a message originates from a blacklisted address, the system equipped with a blacklist promptly identifies it and alerts the recipient. Advanced systems may also employ IP-based data and mail filtering, which blocks messages from suspected IP addresses, enhancing system security. Whitelisted emails are recognized as known identities, allowing users to block known email addresses and safeguard the system from future incoming messages. The whitelist-enabled mail protection system operates similarly to a data store, storing verified sender email addresses and fortifying the email handling system effectively. Another approach to bolstering existing protection schemes involves content-based data filtering, where incoming email content is extracted and compared against a database of blocked content. If the content matches any blocked criteria, the email is promptly designated as spam or junk.

In recent times, the widespread use of the Internet and the cost-effectiveness of email have attracted advertisers to utilize email marketing, leading to a surge in the volume of spam messages received. Many of these advertisements promote dubious schemes promising quick wealth with minimal effort, enticing users to click on them and visit related websites. However, these spam messages not only disseminate unwanted content but also

consume network bandwidth and require additional time for users to distinguish between spam and legitimate messages. This can result in recipient dissatisfaction and expose their systems to viruses and malware. As the volume of electronic data such as e-books, digital libraries, emails, e-newspapers, and e-publications continues to grow rapidly, managing such data becomes increasingly challenging. Text mining, a method for extracting useful information from texts, including summarization, classification, and retrieval, becomes crucial in this context. Text classification (TC), also known as text categorization, is a key aspect of text mining, involving the assignment of text documents to predefined categories. However, manually classifying a large number of text documents is time-consuming and costly, underscoring the importance of research in automatic text management within text mining. Principal Component Analysis (PCA) is a biometric algorithm-based technique used in statistical analysis. PCA employs orthogonal transformation to convert correlated variables into uncorrelated ones. This technique facilitates the compression of multidimensional data while retaining essential information and incorporates concepts such as covariance, standard deviation, and eigenvectors.

To address the issues caused by spam, email filtering is employed using three main approaches: simple, intelligent, and hybrid. The simple approach encompasses techniques such as munging, listing, aliasing, and challenging. In contrast, the intelligent approach utilizes advanced methods like Artificial Neural Network (ANN), Support Vector Machine (SVM), Naïve Bayes, DNA Computing, and Artificial Immune System. The hybrid approach combines multiple methods to enhance performance and overcome limitations faced by individual approaches. Various strategies exist to combat spam threats, including the enactment of antispam laws imposing penalties on email spammers. In spam email detection, two primary approaches are commonly employed: knowledge engineering and machine learning. Knowledge engineering involves accessing network information and IP addresses to differentiate spam emails from legitimate ones, known as origin-based filtering. This approach requires the formulation of rules based on filter techniques or by authorized personnel to distinguish between spam and legitimate emails. Machine learning, on the other hand, is considered more effective and efficient than knowledge engineering as it does not rely on predefined rules. In machine learning-based spam filtering, a set of pre-classified emails serves as a training dataset. Using machine learning algorithms, the machine is trained on this dataset to develop a spam detection model. Filtering remains the prevalent method for distinguishing between spam and non-spam messages, while data mining integrates various techniques, including machine learning, artificial intelligence, pattern recognition, statistics, and database systems, to analyze large volumes of data. In the realm of data mining, numerous algorithms are available to perform various tasks.

Text categorization involves automatically assigning text documents to predefined categories, a task that has become increasingly challenging with the rising use of digital documents in recent years. One of the key challenges in text categorization lies in feature extraction, as achieving high overall accuracy in text classifier systems requires careful consideration of numerous features. Text classification often involves dealing with noisy, redundant, and irrelevant features, which can lead to overfitting in many classifier systems. Spam emails typically aim to defraud recipients of money, often by promoting products promising quick wealth, business success without investments, miracle healthcare solutions, instant beauty enhancements, and other enticing offers. Spammers capitalize on human greed by selecting such spam contents. Some spam emails may also contain sexual content or redirect users to pornography or cheap drugs that claim to enhance sexual performance. Several countries have enacted laws like the CAN-SPAM Act to regulate the sending of commercial emails, with penalties for violations. However, the effectiveness of these laws is limited, as many countries still experience issues with spam emails due to ineffective implementation. The widespread popularity of the Internet has made email the most commonly used communication service worldwide, primarily due to its cost-free nature. However, this accessibility also provides spammers with an opportunity to send bulk spam emails to generate revenue, steal personal information or online account details, or infect recipients' systems with viruses or malware. This is facilitated by the lack of a trusted identity and security system in the current email system, which relies on the Simple Mail Transfer Protocol (SMTP) that does not verify the origin of emails. Consequently, the current SMTP infrastructure is susceptible to abuse, enabling anyone to send spam emails with forged content. As a result, users receive a constant stream of spam emails from anonymous senders in their mailboxes every day.

The basic acoustic segmentation of speech involves classifying speech into voiced, unvoiced, and silent segments. This segmentation closely aligns with the sounds of individual alphabets that comprise human speech. In the digital realm, much of the content is accessible and understandable only to those proficient in a particular language. Linguistic technology aids in providing solutions in a standard interface format, making information accessible to speakers of various languages, particularly in a multilingual society like India, which boasts around 1652 dialects. The internet serves as an extensive repository of data spanning diverse domains such as education, medicine, politics, and sports. Data within each domain is further categorized into various subgroups, with each subgroup treated as an independent entity. Researchers utilize the web as a primary source for collecting

information, with millions of domains registered and billions of files published daily by authors in the field of computer literacy. In recent times, Really Simple Syndication (RSS) feeds have been organized to deliver time-based news, aiming to enhance viewership. These RSS feeds provide valuable information for researchers to inform their future studies and investigations.

Data mining employs two main methodologies: classification, which is a supervised machine learning approach, and clustering, which is an unsupervised learning approach. Through these methodologies, text data can be efficiently categorized into multiple categories using various algorithms. The process of classification involves several stages. Initially, individual categories are defined, each comprising a set of rules based on the texts and classification problem at hand. Subsequently, classification algorithms are utilized to predict the category of a test document based on manually trained data. With the rapid growth of digital products and services becoming integral to daily human activities, the use of emails for communication has also surged. However, this surge has led to an increase in unsolicited emails, posing significant threats to the global economy and security. Due to the ease of access and minimal security and verification required to create new email accounts, this technology is being exploited to obtain users' personal and sensitive data without their consent. Cybercrimes, such as identity theft, involve stealing a user's confidential data, including account numbers, social security numbers, and passwords, for financial gain. Unsolicited emails often masquerade as legitimate sources, enticing recipients to visit fraudulent websites and disclose personal information. According to a report by Kaspersky Lab, in the first quarter of 2019, unsolicited emails accounted for 55.97% of internet traffic, a 0.07% increase from the fourth quarter of 2018. These unsolicited emails are typically classified as spam or phishing emails.

Phishing attacks have become increasingly sophisticated over time, posing challenges for existing antispam solutions. While various antispam solutions have been proposed and implemented, they often struggle to provide satisfactory results as the volume of spam emails continues to rise daily. Despite this, spam filters remain a viable solution for mitigating the influx of unsolicited emails, albeit at the cost of internet processing resources and CPU usage. Spam filtering, a widely adopted technique, works by intercepting unsolicited emails before they reach the recipient's inbox. This method is favored for its ability to effectively eliminate unwanted emails. Users can adjust the sensitivity of the filter based on criteria such as attachments, data type, specific keywords, and more. To construct a robust spam filtering model, a combination of multiple scanning methods is typically employed. Emails serve as a convenient and efficient communication medium for both businesses and individuals to exchange useful and confidential information. However, the prevalence of spam emails, including spoofing, phishing, and junk emails, constitutes a significant portion (60%-70%) of the daily email traffic, with users often receiving around 50 spam emails per day. These spam emails are sent with the intent to increase network traffic, compromise user account credentials, and waste the recipient's time and resources. Hackers utilize various techniques, including malicious codes, to compromise or block user accounts and manipulate their contents. The primary challenge with existing spam email filtering systems lies in their accuracy, which leads to errors in classification. Machine learning algorithms offer a promising approach to email spam filtering, with the Naïve Bayes algorithm being one of the most commonly used due to its simplicity and efficient implementation. Training the Naïve Bayes algorithm involves using a dataset containing both spam and non-spam email contents, tracking the occurrence of words in each category. This algorithm has been successfully applied across various datasets with different attributes and features.

Nowadays, the majority of internet traffic is driven by bulk spam emails, necessitating effective measures to mitigate their negative impact on email usability. Spammers gather email addresses from various sources including websites, newsgroups, chat rooms, customer lists, and even through viruses that capture personal details and sell them to spammers. Filtering spam emails presents challenges due to the sheer volume of emails sent and received daily, as well as the complexity of datasets containing numerous features. Analysis of data from 1998 reveals an exponential growth in spam emails, posing significant threats to online security. A recent survey conducted in March 2013 indicated that the number of spam emails had reached approximately 100 billion, representing a 98% increase from the previous quarter. Spam emails not only disrupt organizational productivity and finances but also have broader economic implications. To combat this threat, anti-spam measures are being implemented, with spam filtering methods based on supervised machine learning being commonly employed. These methods utilize either header- or content-based features to classify emails as spam or legitimate. However, both approaches are susceptible to bypassing, particularly header-based features, highlighting the need for more robust anti-spam solutions.

The spam filtering architecture depicted in Figure 1 comprises several key components. Initially, individual user emails, encompassing both spam and legitimate emails, are collected. Subsequently, the emails undergo an initial transformation process. The model further incorporates components such as the user interface, feature extraction and selection, email data classification, and an analyzer section. Finally, machine learning algorithms are employed to train and test whether a given email is classified as spam or legitimate.

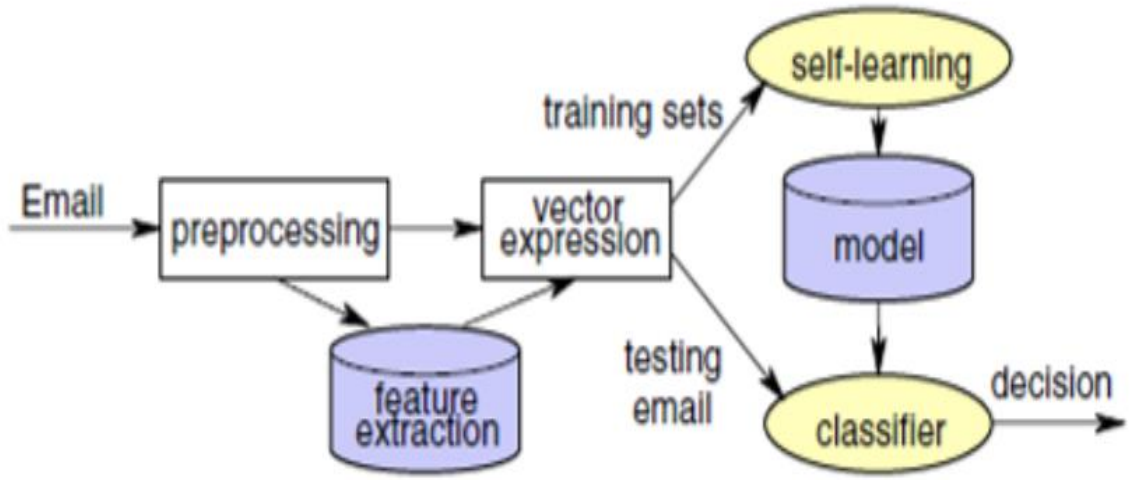

Figure 1. The process of spam filtering

2. Spam Techniques

If a marketer has one database containing names, addresses, and telephone numbers of prospective customers, they can pay to have their database matched against an external database containing email addresses. The company then has the means to send email to persons who have not requested email, which may include persons who have deliberately withheld their email address.

2.1 Image Spam

Image spam represents a sophisticated tactic employed by spammers to circumvent traditional text-based spam filters. In this technique, the textual content of the spam message is embedded within an image file, typically in formats like GIF or JPEG. By presenting the message as an image, spammers aim to evade detection by spam filters that rely on text analysis. This method gained popularity in the mid-2000s, particularly in schemes promoting “pump and dump” stocks. The use of images makes it challenging for spam filters to identify and block the spam messages. However, advancements in technology have led to the development of tools capable of analyzing images for text content, albeit with varying degrees of accuracy. Additionally, spammers have adopted tactics such as using animated GIFs or distorting text shapes to further evade detection by OCR (Optical Character Recognition) tools.

2.2 Blank Spam

Blank spam is a type of spam email that lacks a payload or advertisement content. Often, these emails have no message body or subject line, yet they still qualify as spam due to their unsolicited and bulk nature.

Blank spam can originate from various sources, either intentionally or unintentionally:

1) **Directory Harvest Attack:** Blank spam may be sent as part of a directory harvest attack, which is a form of dictionary attack aimed at gathering valid email addresses from an email service provider. In such attacks, spammers send emails to a large number of addresses and analyze the bounce-back messages to differentiate between valid and invalid addresses. To achieve this goal, spammers may omit most elements of the email header and the entire message body.
2) **Omission by Spammer:** Sometimes, blank spam occurs when a spammer forgets or fails to include the intended payload while setting up the spam campaign. This oversight may result from human error or oversight during the setup process.
3) **Technical Glitches:** Blank spam headers may appear truncated due to technical glitches, such as poorly-written spam software or malfunctioning relay servers. These glitches can lead to problems that truncate header lines from the message body, making the email appear blank.
4) **Hidden Content:** Some spam emails may appear blank to recipients, but they actually contain hidden content. For instance, the VBS. Davinia. B email worm propagates through messages with no subject line, appearing blank to recipients. However, these emails use HTML code to download other malicious files, demonstrating that apparent blank spam emails may still carry hidden threats.

2.3 Backscatter Spam

Backscatter spam is an unintended consequence of email spam, viruses, and worms, resulting in innocent parties receiving bounce-back messages. This occurs when spam or malware-infected emails are sent with forged sender addresses, leading email servers to generate bounce messages to the apparent sender. However, these bounce messages are often misdirected to innocent parties whose email addresses were forged in the original spam emails. As a result, innocent individuals may receive a significant volume of unsolicited emails, known as backscatter spam. These emails, although not directly solicited by the recipients, share similarities with traditional spam, such as bulk distribution and unsolicited nature. Consequently, systems generating backscatter spam may be flagged by spam detection mechanisms and listed on DNSBLs (Domain Name System Blacklists), potentially violating internet service providers' terms of service.

3. Machine Learning in E-Mail Classification

The machine learning field is a subset of the broader artificial intelligence discipline, aiming to equip machines with the ability to learn in a human-like manner. Here, learning involves understanding, observing, and representing information about certain statistical phenomena. Unsupervised learning focuses on identifying hidden patterns or detecting data anomalies, such as spam messages or network intrusions. For instance, in the task of filtering emails, key features might include the analysis of word frequency or subject lines, making the input for email classification appear as a two-dimensional matrix defined by messages and their features. The process of classifying emails typically involves several steps: initially, collecting and representing data specific to the problem, such as the emails themselves; then, selecting and reducing features to simplify the data set for further analysis; and finally, classifying emails by mapping relationships between the training and testing sets. This section will explore some of the most popular methods used in machine learning.

3.1 Naïve Bayes Classifier Method

In 1998, the Naïve Bayes classifier emerged as a method for recognizing spam. This Bayesian classifier operates on the principle that the occurrence of future events can be inferred from past events. Its application in spam email classification hinges on the probability of words, with the frequency of certain words in spam versus non-spam (ham) emails being a key factor. This method has gained popularity in email filtering software due to its effectiveness. The Bayesian filter requires training to perform optimally. It maintains a database where each word is assigned a probability based on its occurrence in spam or ham emails. If the cumulative probability of words in an email surpasses a threshold, the email is categorized accordingly. Only two categories are necessary: spam or ham. Most statistic-based spam filters leverage Bayesian probability to aggregate the statistics of individual tokens into an overall score for spam identification.

The statistic of interest for a token T is its spamminess, calculated as:

$$S[T] = \frac{C_{\text{Spam}}(T)}{C_{\text{Spam}}(T)+C_{\text{Ham}}(T)} \quad (1)$$

Here, $C_{\text{Spam}}(T)$ and $C_{\text{Ham}}(T)$ represent the counts of spam and ham messages containing the token T, respectively. To assess the spam likelihood of a message M containing tokens $\{T_1, \dots, T_N\}$, one combines the spamminess of individual tokens. Classification is achieved by comparing the product of individual tokens' spamminess with the product of their hamminess:

$$H[M] = \prod_{I=1}^{N}(1 - S[T_I]) \quad (2)$$

An email is deemed spam if the product of spamminess $S[M]$ exceeds the hamminess product $H[M]$. This approach is elaborated in the following algorithm.

Input: Email message M
Output: Message marked as spam or non-spam

Stage 1: Training
  For each email, parse into constituent tokens W
    Calculate S[W] = Cspam(W) / (Cham(W) + Cspam(W))
    Store spamminess values in a database

Stage 2: Filtering
  For each message M:
    While (M not end):
      Scan message for the next token T_i
      Query the database for spamminess S(T_i)
      Calculate accumulated message probabilities S[M] and H[M]

Calculate overall message filtering indication I[M] using a filter-dependent function f:
I[M] = f(S[M], H[M])
Example: I[M] = 1 + (S[M] - H[M]) / 2

If I[M] > threshold:
Mark message as spam
Else:
Mark message as non-spam

3.2 K-Nearest Neighbour Classifier Method

The K-nearest neighbour (K-NN) classifier operates as an example-based classifier, which means it uses training documents for comparison rather than relying on an explicit category representation like other classifiers. There's essentially no formal training phase involved. To categorize a new document, it identifies the k most similar documents. If a significant number of these documents belong to a particular category, the new document is also classified under that category. The process of finding the nearest neighbours can be expedited using traditional indexing methods.

In the context of identifying whether a message is spam or ham, the classifier examines the classification of the messages most similar to the given one. This process occurs in real-time, which is a core aspect of the K-nearest neighbor algorithm:

- Stage 1. Training: Store all the training messages.
- Stage 2. Filtering: For a given message x, identify its k nearest neighbours from the training set. If the majority of these neighbours are spam, the message is classified as spam; otherwise, it's considered ham.

This method employs an indexing technique to streamline the comparison process, leading to a sample update with a complexity of $O(m)$, where m is the sample size. Since all training examples are stored in memory, this classifier is also known as a memory-based classifier. A potential issue with this algorithm is the lack of a tunable parameter to minimize false positives. This is addressed by modifying the classification rule to the l/k-rule: if l or more messages among the k nearest neighbours of x are spam, x is classified as spam; otherwise, it's deemed legitimate mail.

The K-nearest neighbour rule is extensively used across various classification tasks and is recognized for its universal consistency in classification.

3.3 Artificial Neural Networks Classifier Method

Artificial Neural Networks (ANNs) are computational models inspired by biological neural networks, comprising a network of artificial neurons. These networks adapt their structure during the learning phase based on incoming data, embodying the principle of "learning by example."

Among the classical neural network types are the Perceptron and Multilayer Perceptron, with our focus here on the Perceptron algorithm. The Perceptron aims to identify a linear function of the feature vector $f(x) = w^T x + b$ that yields $f(x) > 0$ for vectors of one class and $f(x) < 0$ for another. Here, $w = (w_1, w_2, \dots, w_m)$ represents the weights of the function, and b is known as the bias. By assigning classes the values of +1 and -1, we aim to develop a decision function $d(x) = \text{sign}(w^T x + b)$.

The learning process of the Perceptron follows an iterative algorithm, starting with arbitrarily chosen decision parameters (w_0, b_0) and updating them through each iteration. At step n, a training sample (x, c) that is incorrectly classified by the current decision function i.e., $\text{sign}(w_n x + b_n) \neq c$-is selected. The parameters (w_n, b_n) are then adjusted according to the rule:

$$w_{n+1} = w_n + cx \quad b_{n+1} = b_n + c \tag{3}$$

The process repeats until a decision function that correctly classifies all training samples is found. This methodology underpins the algorithm discussed above.

Algorithm: Perceptron Learning

Stage 1: Training
1. Initialize weight vector w and bias b to random values or to 0.
2. Repeat until convergence:
a. Find a training example (x, c) that misclassifies, i.e., where sign (w^T x + b) is incorrect.
b. If no such example exists, training is complete, exit loop.
c. Update the weight vector and bias:
w: = w + c * x

b: = b + c

Stage 2: Filtering
1. For a new message x, predict its class as sign (w^T x + b).

3.4 Support Vector Machines Classifier Method

Support Vector Machines (SVMs) rely on the concept of decision planes that establish decision boundaries. A decision plane distinguishes among features belonging to different categories. The SVM seeks the best hyperplane with the maximum margin to separate two classes, necessitating the solution to this optimization problem:

Maximize the equation:

$$Z = \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j K(x_i, x_j) \quad (4)$$

Subject to the conditions:

$$\sum_{i=1}^{n} \alpha_i y_i = 0 \text{ and } 0 \leq \alpha i \leq b, i = 1,2, \ldots n \quad (5)$$

Here, α_i represents the weight of training sample x_i, and if $\alpha_i > 0$, then x_l is deemed a "support vector." The parameters b and α's are regulatory, enabling a more flexible data separation. K is a kernel function utilized to measure the similarity between two data points. A widely used radial basis function (RBF) kernel is:

$$K(x_i, x_j) = \exp(-\gamma \|x_i - x_j\|^2), \gamma > 0 \quad (6)$$

After weights are computed, a test sample x is classified by:

$$y = \text{Sign}\left(\left(\sum_{i=1}^{n} \alpha_i y_i K(x_i, x_j)\right)\right) \quad (7)$$

where $\text{Sign}(a)$ is +1 if $a > 0$ and -1 otherwise.

The parameters γ and b are typically determined through cross-validation on the training dataset. Cross-validation also helps assess the classifier's generalization capability on unseen data. In k fold cross-validation, the training dataset is divided into k roughly equal parts. One part is left out while a classifier is trained on the remaining parts, then tested on the left-out part. This cycle repeats k times for each segment to derive a comprehensive cross-validation performance metric across the training dataset. For large datasets, a smaller subset may be used for cross-validation to reduce computational demands. The described procedure forms the basis of the classification algorithm.

Algorithm: Nearest Neighbor SVM Classification

Input:
- sample x to classify
- training set T = { (x1, y1), (x2, y2), ..., (xn, yn) }
- number of nearest neighbors k

Output:
- decision yp $\in$ {-1, 1}

Procedure:
1. Initialize a list to store the k nearest samples from T to x based on a kernel function K.
2. For each sample (xi, yi) in T:
   a. Compute the distance to x using the kernel function: D = K(xi, xi) - 2 * K(xi, x)
   b. Add the sample (xi, yi) and its distance D to the list.
3. Sort the list by distance in ascending order and select the first k samples.
4. Train an SVM model using the k selected samples.
5. Classify the sample x using the trained SVM model to obtain the result yp.
6. Return the decision yp.

3.5 Artificial Immune System Classifier Method

The Biological Immune System has been notably effective in safeguarding the human body against a broad spectrum of foreign pathogens, safeguarding us from infectious agents like viruses and bacteria. These agents carry antigens on their surfaces that are identifiable by the immune system, triggering an immune response. This recognition process is carried out by lymphocytes, each displaying receptor molecules, known as antibodies, of a unique shape on its surface. These receptors are constructed through a complex genetic mechanism that includes the combinatorial recombination of various gene segments.

The immune response is characterized by three evolutionary strategies: gene library, negative selection, and clonal selection. Within the gene library, antibodies identify antigens through complementary properties unique to antigens, requiring some level of antigen property knowledge to create effective antibodies. This evolutionary self-organization is mirrored in spam management, where gene libraries serve as reservoirs of knowledge on identifying commonly encountered antigens. A critical task for the immune system is to avoid attacking the body's cells, achieved through negative selection that eliminates inappropriate antibodies binding to self cells. Clonal selection then replicates well-performing antibodies, ensuring that, in the presence of current antigens, only the most effective antibodies persist. This system operates without specific antigen information, instead organizing the best-suited antibodies through interaction with present antigens.

This comprehensive description outlines the algorithm that follows, illustrating the immune system's complex yet efficient mechanisms for protecting the body and inspiring approaches in fields such as spam management.

Algorithm: Artificial Immune System for Email Classification

Input:
- Email message m
- Library of base strings and associated detectors

Parameters:
- s-rate: Rate of increase for spam score
- ns-rate: Rate of increase for non-spam (ham) score
- a-rate: Age decrease rate for base strings
- threshold: Threshold for edit distance matching function (edmf)

Procedure:
1. For each term t in the email message m:
  a. If a detector p exists that matches term t:
    i. If the message m is spam:
      - Increase the spam score of base string r by s-rate.
    ii. Else:
      - Increase the ham score of base string r by ns-rate.
  b. Else:
    i. If the message m is spam:
      - If detector p recognizes term t and the edit distance matching function edmf(p, t) is greater than the threshold:
        * Update the library of character generalization rules with the differing characters.
    ii. Else:
      - Add the new base string t to the library of base strings.
2. Decrease the age of every base string in the library by a-rate.

Output:
- Updated library of base strings with their respective spam and ham scores.
- Updated set of character generalization rules.

3.6 Rough Sets Classifier Method

Pawlak introduced Rough Set Theory in 1982, offering a powerful means to deduce reductions within information systems. Given that an information system may contain attributes irrelevant or redundant to the target concept (decision attribute), reduction becomes essential to distill simple, actionable knowledge. This process involves identifying the smallest subset of condition attributes that correlate with decision attributes. The approach for applying Rough Set theory is outlined in the steps below:

- Step 1: For incoming emails, the initial task is to select the most relevant attributes for classification. The dataset is then structured into a decision system and divided into training and testing sets. A classifier, developed from the training set, is subsequently applied to the testing set to gauge performance. This step includes proceeding with Steps 2 and 3 for the training dataset.
- Step 2: Given the decision system's real-valued attributes, a Boolean reasoning algorithm is employed to implement necessary discretization strategies.
- Step 3: Genetic algorithms are utilized to derive decision rules. This step is specifically for the training set, leading to Step 4 for the testing set.

- Step 4: The testing set is discretized using the same parameters established in Step 2. The decision rules generated in Step 3 are then applied to new items within the testing set to facilitate decision-making.

The threshold $b = 0.15 \in (0,1/2)$ is established for the positive region, resulting in the division of emails into three regions: 0.15 -positive, 0.15 -boundary, and 0.15 -negative, based on lower and upper $b -$ approximations. This framework is utilized in the subsequent algorithm.

This elaboration captures the essence of Rough Set Theory's application in information systems, highlighting its structured approach to simplifying and extracting valuable knowledge from data.

Algorithm: Email Classification Using Discretized Data and Rule-Based System

Input:
- Dis_T: Discretized testing dataset
- RUL: Set of rules from the learning phase
- b: Threshold for classification decision, b = 0.15

Output:
- Classification of each email in Dis_T into one of three categories: non-spam, spam, and suspicious

Procedure:
1. For each object x in Dis_T:
  a. If RUL(x) = 0 (i.e., no rules apply to x):
    i. Mark x as suspicious.
  b. Else, let each rule r in RUL(x) that predicts non-spam contribute to a collective score:
    i. R = { r ∈ RUL(x) | r predicts non-spam }
    ii. Estimate the relevance of x to non-spam class: Rel(Dis_T | x ∈ non-spam)
    iii. Calculate the total relevance score for non-spam:
      Rel(Dis_T | x ∈ non-spam) = Σ (r ∈ R) Predicts(non-spam)
  c. Compute the certainty of x being non-spam:
    i. Certainty_x = (1 / CER) * Rel(Dis_T | x ∈ non-spam)
  d. If Certainty_x ≥ (1 - b):
    i. Mark x as suspicious.
  e. Else:
    i. Mark x as spam.

2. Return the classification for each email in Dis_T.

4. Methodology

Our study aimed to develop a robust machine learning system capable of accurately classifying emails into 'Spam' or 'Not Spam' categories. This process involved several key stages: data preprocessing, feature extraction, model training and evaluation, and deployment through a user interface for real-world application.

4.1 Data Preparation

The study utilized a comprehensive dataset named completeSpamAssassin.csv, consisting of emails with labels indicating whether each email was "Spam" or "Not Spam." The primary objective during the data preparation stage was to clean and preprocess this textual data, ensuring it was in a suitable form for feature extraction and subsequent model training.

4.1.1 Preprocessing Steps

The preprocessing of the dataset involved several key steps, each designed to standardize the email text and enhance the quality of the features available for learning. These steps were:

1) **Text Cleaning**:

– **Punctuation Removal**: All punctuation marks were removed from the email texts. This step is essential because punctuation does not typically contribute to the distinction between Spam and Not Spam emails.

– **Lowercasing**: The entire email text was converted to lowercase to ensure that the same words in different cases are treated equally. This process aids in reducing the overall feature space.

– **Whitespace Trimming**: Extra spaces, including tabs and new lines, were removed. This step helps in standardizing the text data.

2) **Tokenization**:

– The cleaned text was then split into individual words or tokens. Tokenization is crucial for analyzing the text at the word level, allowing for the identification of patterns or frequent terms used in Spam versus Not Spam emails.

3) **Stop Words Removal**:

– Commonly used words in the English language, such as “the,” “is,” and “in,” which are unlikely to be useful for distinguishing between Spam and Not Spam, were removed from the text. This step helps focus the model’s attention on more meaningful words.

4) **Stemming**:

– Words were reduced to their root form using a stemming algorithm. For instance, “running,” “runs,” and “ran” all stem to “run.” This process reduces the complexity of the text data by consolidating similar forms of a word into a single representation, thereby reducing the feature space and potentially improving model performance.

4.2 Feature Extraction

The primary goal of feature extraction in our study was to convert the cleaned and standardized email texts into a structured, numerical format that could be efficiently processed by machine learning models. This transformation is essential for analyzing the text data, as it allows the models to identify patterns or features that differentiate spam emails from legitimate ones.

4.2.1 Techniques Used

1) Count Vectorization:

– The method chosen for feature extraction was Count Vectorization, implemented using the CountVectorizer class from the sklearn.feature_extraction.text module. This technique involves creating a vocabulary of all the unique words present in our training data and then mapping each email to a numerical vector. Each vector’s elements correspond to the vocabulary’s words, indicating the frequency of each word in the email.

– Count Vectorization is particularly suitable for our dataset because it captures the presence and abundance of words which are potentially indicative of spam content. For example, spam emails may frequently use words like “offer,” “free,” or “win,” which can be flagged by analyzing these frequency vectors.

4.2.2 Process

1) **Building the Vocabulary**:

– The first step in Count Vectorization was to build a vocabulary comprising all unique words across the preprocessed email texts. This vocabulary serves as a reference for converting text data into numerical vectors, where each word is assigned a specific index in the vector.

2) **Vectorization**:

– Each preprocessed email was then transformed into a numerical vector using the constructed vocabulary. For each email, the corresponding vector was filled with the counts of how often each vocabulary word appeared in the email. This resulted in a sparse matrix representation, where the majority of elements are zero due to the vast size of the vocabulary compared to the number of words in an individual email.

4.2.3 Advantages of Count Vectorization

• **Simplicity and Efficiency**: Count Vectorization is straightforward to implement and understand. It provides a direct way of converting text into numerical data, making it a popular choice for text classification tasks.

• **Highlighting Frequent Words**: This method naturally emphasizes words that occur frequently across the dataset, which can be particularly useful for identifying common characteristics of spam emails.

4.3 Model Training

Following the preprocessing and feature extraction phases, the next critical step was training machine learning models to classify emails into ‘Spam’ or ‘Not Spam’. We employed three distinct algorithms for this purpose: Naive Bayes, Random Forest, and Logistic Regression. Each model was chosen for its unique characteristics and suitability for binary classification tasks.

4.3.1 Selection of Models

1) **Naive Bayes Classifier**: Known for its simplicity and efficiency in text classification tasks, particularly in spam detection. It operates under the assumption that the presence of a particular feature in a class is independent of the presence of any other feature.

2) **Random Forest Classifier**: An ensemble learning method that operates by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes of the individual trees. It is chosen for its high accuracy and robustness to overfitting.

3) **Logistic Regression Classifier**: Despite its name, Logistic Regression is a linear model for binary classification rather than regression. It estimates probabilities using a logistic function, which is particularly useful for binary classification tasks.

4.3.2 Training Process

The training process involved the following steps for each model:

1) **Data Splitting**: The dataset was split into training and testing sets, with 80% of the data allocated for training and the remaining 20% reserved for testing. This split ensures that the model can be trained on a substantial portion of the data while leaving a separate, untouched portion for evaluation.

2) **Vectorization**: As part of feature extraction, the training data was transformed into numerical vectors using Count Vectorization. This step is crucial as it converts the preprocessed email texts into a format that the models can process.

3) **Model Fitting**:

– Each model was then "fitted" or "trained" on the training set. This process involves adjusting the model parameters to minimize the difference between the predicted and actual classifications on the training data.

– For **Naive Bayes** and **Random Forest** classifiers, the default parameters from the Scikit-learn library were initially used. Given the high dimensionality of our feature space, these models were deemed appropriate due to their inherent capacity to handle multiple features.

– The **Logistic Regression** model was trained with the default settings, except for the regularization strength which was adjusted to prevent overfitting, a common issue with high-dimensional data.

4.3.3 Hyperparameter Tuning

While initial training used default parameters, subsequent iterations involved adjusting hyperparameters to optimize model performance. This process, known as hyperparameter tuning, was particularly focused on the Random Forest and Logistic Regression models, where parameters such as the number of trees (in the forest) and the regularization strength (in logistic regression) can significantly impact performance.

4.4 Model Evaluation

Upon training the Naive Bayes, Random Forest, and Logistic Regression models, a rigorous evaluation was conducted to determine each model's performance in accurately classifying emails as spam or not spam. This evaluation phase is pivotal for identifying the most effective model based on empirical evidence.

4.4.1 Evaluation Metrics

To objectively assess and compare model performance, the following metrics were used:

• **Accuracy**: The proportion of total predictions (both spam and not spam) that were correctly identified by the model. It provides a general measure of the model's effectiveness.

• **Precision**: Among the emails predicted as spam, precision measures the proportion that was actually spam. This metric is crucial for evaluating the reliability of the model's spam detection.

• **Recall**: Of the actual spam emails, recall quantifies the proportion that the model correctly identified. This metric is essential for assessing the model's ability to detect as many spam emails as possible.

• **F1 Score**: The harmonic mean of precision and recall, providing a single metric to assess the balance between them.

4.4.2 Evaluation Process

The evaluation process involved the following steps:

1) **Application of Trained Models**: Each model was applied to the same test dataset, which had not been used during the training phase. This ensures that the evaluation reflects each model's ability to generalize to new, unseen data.

2) **Performance Metrics Calculation**: For each model, predictions on the test set were compared to the actual labels to calculate the metrics defined above. This comparison provided a quantitative assessment of each model's performance.

4.5 User Interface

The primary objective of developing a user interface was to provide an easy and intuitive way for users to classify emails as spam or not spam. This interface was designed to hide the complexity of the underlying machine learning models and offer a straightforward, user-friendly experience.

4.5.1 Design Considerations

The design focused on simplicity and efficiency, ensuring users could easily input their email texts and receive classification results with minimal effort. Key considerations included:

- **Accessibility**: Ensuring the interface is easy to navigate for all users, regardless of their technical background.
- **Responsiveness**: Designing the UI to be responsive, allowing it to be used across different devices and screen sizes.
- **Feedback**: Providing immediate and clear feedback to the user about the classification result.

4.5.2 Implementation

The user interface was implemented using Flask, a Python web framework, and HTML for the front-end design. Flask was chosen for its simplicity and flexibility in deploying web applications and integrating Python-based machine learning models.

Backend

The Flask application acts as the backend server, handling requests from the user interface. It integrates the Logistic Regression model for spam classification, utilizing pre-trained model and vectorizer files (classifierLR.pkl and vectorizer.pkl) for processing user inputs. The backend includes routes for handling both GET and POST requests:

- **GET Request**: Renders the initial form where users can input the email text.
- **POST Request**: Receives the email text from the form, processes it through the spam classification model, and returns the result to the user.

Frontend

The frontend was developed using HTML and Bootstrap for styling, emphasizing usability and a clean layout. It consists of a simple form where users can input the text of the email they wish to classify. Upon submission, the interface communicates with the Flask backend, displays the classification result (‘Spam’ or ‘Not Spam’), and offers the option to classify another email.

4.5.3 User Flow

1) **Input**: The user is presented with a text box to input or paste the content of the email.
2) **Submission**: After entering the email text, the user submits it for classification by clicking a button.
3) **Processing**: The input is sent to the Flask backend, where it undergoes preprocessing, vectorization, and classification using the Logistic Regression model.
4) **Display Results**: The classification result is sent back and displayed to the user on the same page, with a message indicating whether the email is spam or not spam.
5) **Repeat**: Users have the option to classify another email by simply entering new text into the input box and submitting it again.

5. Performance Measures

The various performance measures used in the proposed methodology are given as follows.

5.1 Confusion Matrix

With the help of various performance measures, it is possible to evaluate the spam e-mail detection. A Confusion Matrix is used to detect the performance for the spam e-mail detection model, as shown in Table 2.

True Negative Rate (TNR) is the ratio of True Negatives to sum of True Negatives and False Positives, as follows:

$$\mathrm{TNR} = \frac{TN}{TN+FP}. \tag{8}$$

True Positive Rate (TPR) is defined as the ratio of True Positives to sum of True Negatives and False Positives, as follows:

$$\mathrm{TPR} = \frac{TP}{TP+FN}. \tag{9}$$

False Negative Rate (FNR) is the ratio of False Negatives to sum of False Negatives and True Positives, as follows:

$$\text{FNR} = \frac{FN}{FN+TP}. \tag{10}$$

False Positive Rate (FPR) is the ratio of False Positives to sum of False Positives and True Negatives, as follows:

$$\text{FPR} = \frac{FP}{FP+TN}. \tag{11}$$

5.2 Accuracy

The accuracy of a model depends on the prediction of true values of spam and ham and is expressed as follows:

$$\text{Accuracy } = \frac{TP+TN}{TP+TN+FP+FN} * 100. \tag{12}$$

5.3 Recall

The recall is the measurement of the number of True Positive values predicted in a given set of spam e-mails as follows:

$$\text{Recall } = \frac{TP}{TP+FN}. \tag{13}$$

5.4 Precision

The precision is the measurement of the number of true values predicted in a given set of positive e-mails as follows:

$$\text{Precision } = \frac{TP}{TP+FP}. \tag{14}$$

5.5 F1-Score

The F-measure F_β is calculated by using precision and recall scores. The value of F_β gives the value of F1-score. F1-score is defined as the harmonic mean of precision and recall and is expressed as follows:

$$F_\beta = \frac{(1+\beta^2)(\text{ Precision} * \text{Recall })}{(\beta^2 * (\text{ Precision} + \text{Recall }))}. \tag{15}$$

With β value $= 1$, the formula becomes

$$F_\beta = \frac{2 * (\text{ Precision} * \text{Recall })}{1 * \text{ Precision} + \text{Recall}}. \tag{16}$$

6. Performance Comparison

In the academic study of machine learning applications for email spam detection, the performance comparison of classifiers is pivotal. The classifiers under consideration for this study are the Naive Bayes Classifier, Random Forest Classifier, and Logistic Regression Classifier. The performance metrics used for the comparison include Accuracy, Precision, Recall, and F1-Score, derived from the confusion matrix components: True Negatives (TN), False Positives (FP), False Negatives (FN), and True Positives (TP). Additionally, computational efficiency is evaluated through CPU execution time in relation to buffer size for email processing.

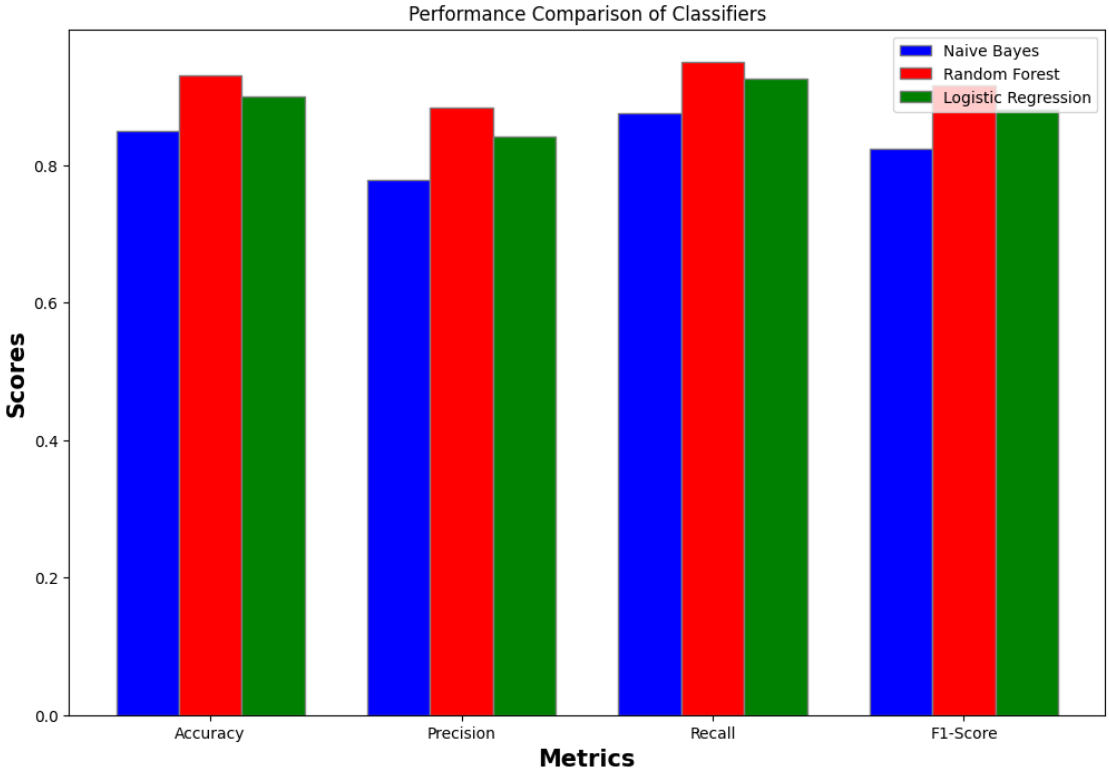

Figure 2. Comparative Analysis of Machine Learning Classifiers Based on Key Performance Metrics

Upon evaluation, the Random Forest Classifier exhibited an accuracy of 93%, a precision of 88.4%, a recall of 95%, and an F1-Score of 91.6%. This performance is noteworthy when contrasted with the Naive Bayes Classifier, which achieved an accuracy of 85%, a precision of 77.8%, a recall of 87.5%, and an F1-Score of 82.4%, and the Logistic Regression Classifier, with an accuracy of 90%, a precision of 84.1%, a recall of 92.5%, and an F1-Score of 88.1%. The higher accuracy and precision of the Random Forest Classifier indicate a more effective model for correctly classifying spam emails, with a lower likelihood of misclassification compared to the other two classifiers.

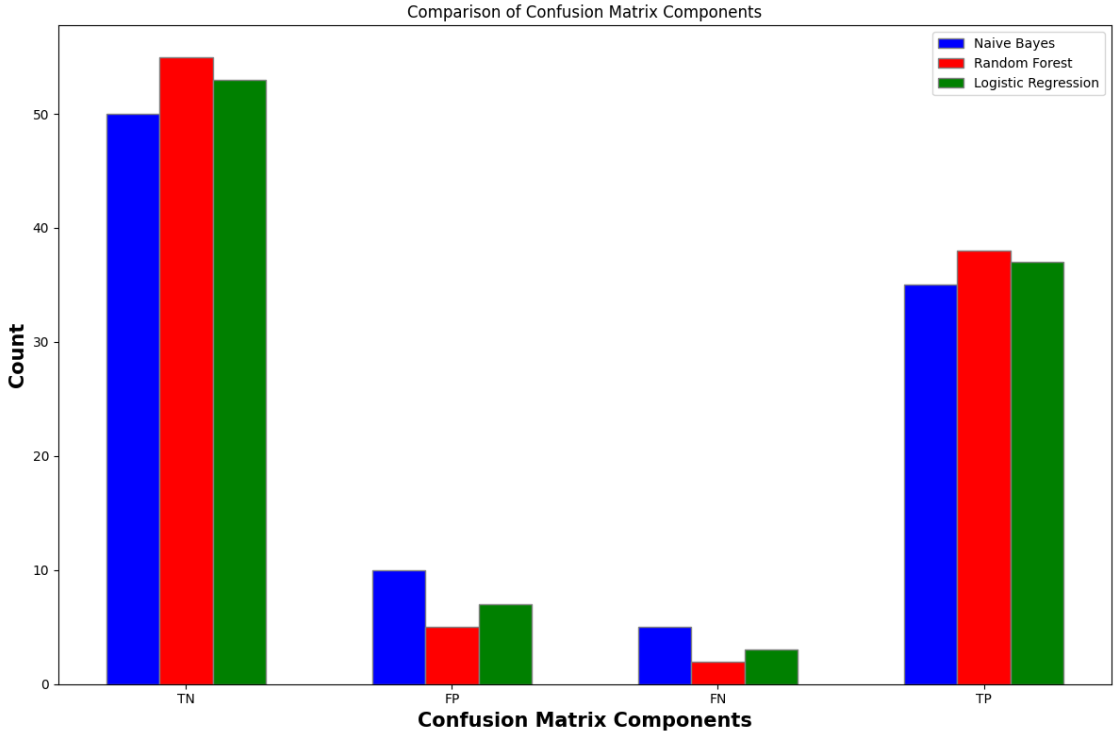

Figure 3. Distribution of Confusion Matrix Components Across Machine Learning Classifiers

The confusion matrix further illustrates the Random Forest Classifier's enhanced ability to correctly identify non-spam emails (TN=55) while minimizing the misidentification of legitimate emails as spam (FP=5). In comparison, the Naive Bayes Classifier demonstrated a higher misclassification rate with FP=10 and FN=5. Logistic Regression presented intermediate results with FP=7 and FN=3, indicating a moderately effective classification capability.

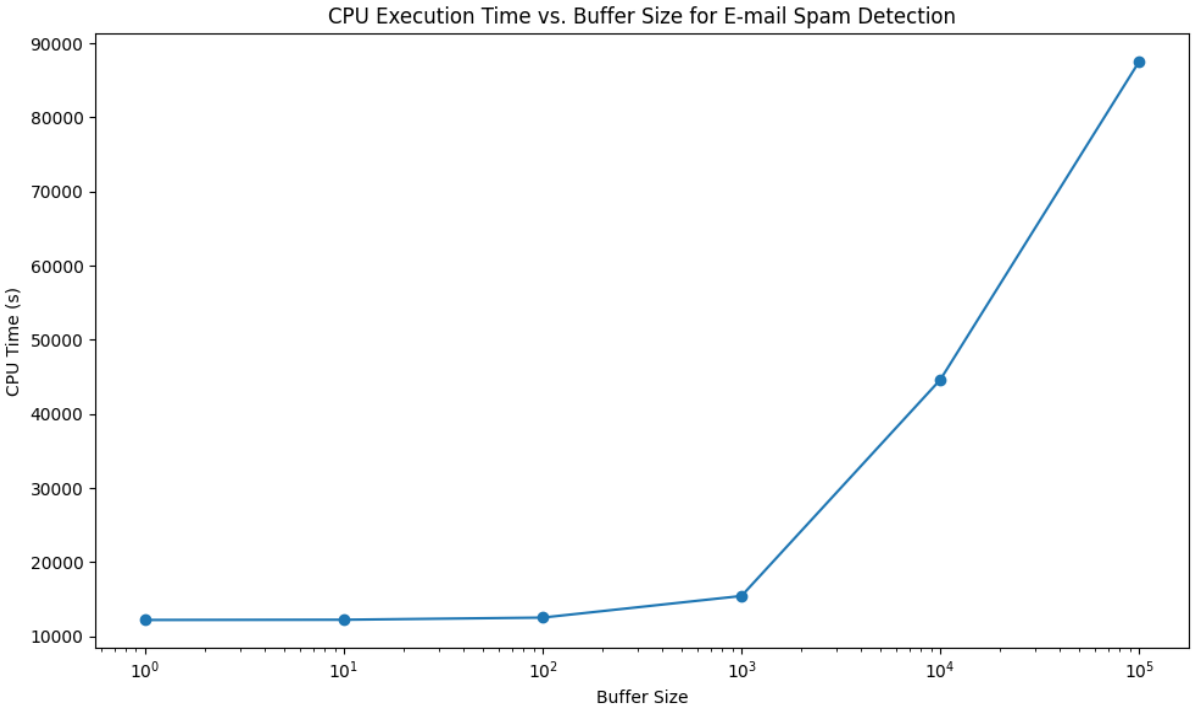

Figure 4. Computational Efficiency of Email Spam Detection: CPU Time as a Function of Buffer Size

The CPU execution time analysis for e-mail spam detection provides insights into the scalability and computational demand of the classifiers. The execution time displayed a non-linear relationship with the buffer size, suggesting increased computational complexity with larger data volumes. Such findings underscore the importance of optimizing algorithms to balance classification accuracy and computational efficiency, particularly for real-time processing needs.

In summary, the empirical data elucidates the Random Forest Classifier's superiority in accuracy and its balanced precision-recall relationship, establishing it as the most effective model among those evaluated for the task of spam detection. These findings advocate for the Random Forest Classifier's deployment in spam filtering solutions, contingent upon further verification in varied operational environments and with diverse datasets to ensure generalizability and robustness of the model.

7. Conclusion

The investigation into the effectiveness of machine learning models in the realm of email spam detection has presented significant insights and empirical evidence regarding the comparative performance of Naive Bayes, Random Forest, and Logistic Regression classifiers. Our study's methodology, encompassing data preparation, feature extraction, model training, evaluation, and user interface deployment, forms a robust framework for discerning the most adept algorithm in the context of spam detection.

Through meticulous preprocessing and feature extraction, we ensured that the machine learning models received data of optimal quality, which is essential for accurate classification. The training phase utilized standard practices, with hyperparameter tuning playing a crucial role in enhancing the model's capabilities. Our evaluation across multiple performance metrics—Accuracy, Precision, Recall, and F1-Score—provided a multi-faceted view of each model's strengths and limitations.

The Random Forest Classifier emerged as a standout performer, achieving the highest scores in all metrics. This indicates its superior capability to classify emails accurately while maintaining a low rate of false positives and false negatives. However, it is imperative to consider the trade-off between accuracy and computational efficiency. The analysis of CPU execution time illustrated that while Random Forest and Logistic Regression

have superior accuracy, they also demand greater computational resources, especially as data volume increases. Therefore, the choice of classifier may ultimately depend on the specific requirements and constraints of the application environment, such as real-time processing needs and computational resource availability.

In addition to the performance on numerical metrics, the study also considered the usability of these classifiers through a user interface implemented in Flask. The UI demonstrated the practical application of Logistic Regression Classifier, allowing end-users to classify emails with ease, thereby showing the potential of these models to be integrated into real-world systems.

This comprehensive study provides a foundation for future research, where further enhancements can be explored, such as ensemble methods, deep learning techniques, or the integration of unsupervised learning for better adaptability to evolving spam trends. Moreover, the study underscores the importance of continual adaptation and evolution of spam detection methods to keep pace with the ever-changing landscape of cybersecurity threats.

Ultimately, the findings affirm the viability of employing machine learning classifiers for spam detection, with each classifier's applicability being subject to the specific operational context. It is hoped that the insights gleaned from this research will inform the development of more sophisticated, efficient, and user-friendly spam detection tools, contributing to the safer and more secure use of email communication.

References

Almeida, T. A., Almeida, J., & Yamakami, A., (2011). Spam filtering: how the dimensionality reduction affects the accuracy of Naive Bayes classifiers. *Journal of Internet Services and Applications*, *1*, 183-200.

Carpinteiro, O. A., Lima, I., Assis, J. M., de Souza, A. C. Z., Moreira, E. M., & Pinheiro, C. A., (2006). A neural model in anti-spam systems. In *Artificial Neural Networks–ICANN 2006: 16th International Conference, Athens, Greece, September 10-14, 2006. Proceedings, Part II 16* (pp. 847-855). Springer Berlin Heidelberg.

Cormack, G. V., Smucker, M. D., & Clarke, C. L., (2011). Efficient and effective spam filtering and re-ranking for large web datasets. *Information retrieval*, *14*, 441-465.

Guzella, T. S., & Caminhas, W. M., (2009). A review of machine learning approaches to spam filtering. *Expert Systems with Applications*, *36*(7), 10206-10222.

Khorsi, A., (2007). An overview of content-based spam filtering techniques. *Informatica*, *31*(3).

Li, K., & Zhong, Z., (2006, June). Fast statistical spam filter by approximate classifications. In *Proceedings of the joint international conference on measurement and modeling of computer systems* (pp. 347-358).

Marsono, M. N., El-Kharashi, M. W., & Gebali, F., (2008). Binary LNS-based naïve Bayes inference engine for spam control: noise analysis and FPGA implementation. *IET Computers & Digital Techniques*, *2*(1), 56-62.

Marsono, M. N., El-Kharashi, M. W., & Gebali, F., (2009). Targeting spam control on middleboxes: Spam detection based on layer-3 e-mail content classification. *Computer Networks*, *53*(6), 835-848.

Michael, H. Z. A. C. B., & Malik, M. J., (2007). SVM-KNN: Discriminative Nearest Neighbor Classification for Visual Category Recognition. Computer Science Division, EECS Department Univ. of California, Berkeley, CA, 94720.

Parizo, B., (2006, July 26). Image spam paints a troubling picture. *Search Security.*

Tang, Y., Krasser, S., He, Y., Yang, W., & Alperovitch, D., (2008, November). Support vector machines and random forests modeling for spam senders behavior analysis. In *IEEE GLOBECOM 2008-2008 IEEE Global Telecommunications Conference* (pp. 1-5). IEEE.

West, B., (2008). Getting it wrong: Corporate America spams the afterlife. *Clueless Mailers.* (January 19, 2008).

Wu, C. H., (2009). Behavior-based spam detection using a hybrid method of rule-based techniques and neural networks. *Expert systems with Applications*, *36*(3), 4321-4330.

Copyrights