Mesopotamian journal of Cybersecurity
Vol. 5, No.2, **pp.** 657–670
DOI: https://doi.org/10.58496/MJCS/2025/039; ISSN: 2958-6542
https://mesopotamian.press/journals/index.php/cybersecurity

Research Article

A Stacked Ensemble Classifier for Email Spam Detection via an Evolutionary Algorithm

Salam Al-augby[1], *, Hasanen Alyasiri[1], Fahad Ghalib Abdulkadhim[1], Zahraa Ch. Oleiwi [2]

[1] *Faculty of Computer Science and Mathematics, University of Kufa, Najaf, Iraq*

[2] *Faculty of Computer Science and Information Technology, University of Al-Qadisiyah, Qadisiyah, Iraq*

ARTICLEINFO

Article history

Received 17 Dec 2024

Revised: 25 May 2025

Accepted 31 May 2025

Published 6 Jul 2025

Keywords

Spam detection

Ensemble learning

Evolutionary Algorithm

Data Analysis

ABSTRACT

Email communication is a crucial aspect of modern interactions. With the growing volume of spam emails, there is a pressing need for more effective antispam filters to detect unwanted messages. Existing spam detection techniques often fall short, prompting researchers to leverage machine learning and artificial intelligence to enhance online security. This study introduces an advanced spam detection technique using an ensemble learning approach. First, key features are extracted from both spam and non-spam emails via the term frequency-inverse document frequency (TF-IDF) method. Several classification algorithms, including cubist, naïve Bayes, support vector machine, rpart, and ctree, are applied to classify emails on the basis of the extracted features. A stacking model that uses an evolutionary algorithm is implemented to further increase the detection accuracy. The effectiveness of the proposed methodology is evaluated on widely adopted spam datasets. The results demonstrate its robustness, achieving an accuracy of 98.39% on the Enron dataset and 98% on the SpamAssassin dataset, highlighting its efficiency and significance in spam email detection.

1. INTRODUCTION

As electronic communication has proliferated, email has emerged as one of the most prevalent modes of communication worldwide. Recent email data indicate that over 347.3 billion emails are sent and received daily in 2023, with projections exceeding 400 billion by 2027 [1]. Spam, in the case of online communication, can be defined as unwanted persistent messaging sent out in mass for commercial or other undesirable reasons. This can occur through multiple methods, including email, social media, instant messaging, and even phone calls [2]. In 2023, 45.6% of all emails were classified as spam [3]. Spam emails waste time and resources, potentially leading to malware dissemination, phishing efforts, and theft of valuable information, thus presenting a significant issue to individuals and companies alike. Consequently, spam email identification is an essential function in email security, as it protects internet users from such attacks. Machine learning (ML) can assist in preventing or alleviating numerous dangers, including spam. Supervised machine learning-based spam detectors have demonstrated efficacy by producing improved accuracy results with reduced variability, hence ensuring greater consistency for this approach [4], [5]. Malware detectors generated from a single ML algorithm have been clearly studied and have achieved excellent results. However, compared with single classifiers, ensemble learning has achieved superior results [6], [7]. The process of combining a set of models (i.e., classifiers) and forming a single strong model is referred to as an ensemble. The primary objective is to utilize the strengths of each algorithm within the ensemble to achieve a resilient classifier.

Ensemble learning can be classified into three categories: stacking, boosting, and bagging. Bagging is a simple yet powerful ensemble method. This method involves dividing the training dataset into several samples through random selection of data with replacement. Each sample is subsequently utilized to train the respective base model. In classification tasks, the outputs of all models are aggregated, resulting in a solitary conclusion via a voting mechanism. In the regression analysis, the results of all the models are consolidated to derive the final conclusion. The model demonstrates stability by reducing variance and bias in the data distribution [6]. Boosting is an ensemble technique that transforms a collection of weak classifiers into robust classifiers [8]. The predictors are obtained progressively. The initial predictive model is trained via comprehensive datasets, whereas subsequent models derive insights from the performance of prior learners, and so on. In each iteration, the weight must be increased for every instance misclassified by prior learners. In the classification problem, the ultimate prediction is

*Corresponding author. Email: salam.alaugby@uokufa.edu.iq

ascertained by consolidating the outputs of all the predictors via a weighted majority vote. In regression problems, the ultimate prediction is ascertained by the weighted sum of the outputs from all the predictors. The bagging and boosting ensemble techniques depend on voting, whereas stacking combines lower-level base learners to produce high-level learners. Wolpert reported that ensemble approaches produced greater accuracy than did a singular trained model [7]. In a stacking model, the individual learners are referred to as base models, each trained on the entire training dataset. A meta-model is then employed to combine the outputs of these base models, aiming to determine the optimal combination of predictions for improved overall performance. While prior studies have explored the use of evolutionary algorithms (EAs) to select the most effective base models and their combinations, comprehensive ablation studies remain essential. Such studies are critical for understanding the contributions of individual components within ML strategies [9]. Hence, it is essential to understand the reasons behind and the processes by which ML models make decisions when they are applied in security contexts. Recently, several frameworks for interpreting ML models have been developed, helping to enhance users' comprehension of and confidence in these models [10].

Detection techniques based on ensemble learning are able to satisfy the growing demand for reliable and intelligent solutions. Ensembles help meet various security challenges, such as an insufficient amount of quality training data, a reduction in false alarms (i.e., false positives and false negatives), high-dimensional features, and various applications or parts of the investigated system [11], [12]. EAs are a class of optimization and search algorithms inspired by natural selection. They solve complex problems by iteratively improving a population of candidate solutions [13]. Both ML and EA have been used to address various security problems, including networks [14], botnet [15], phishing attacks [16], IoT threats [17], VANET [18], and insider threats [19]. EAs exhibit several advantageous characteristics, including generating interpretable outcomes, producing lightweight models, and utilizing fewer features than traditional ML algorithms do [13]. These attributes are particularly valuable for security teams seeking efficient and transparent spam detection solutions [20]. Furthermore, we use DistilBERT, a resource-efficient pre-trained language model, to perform comparative analysis in email classification. A distilled version of the larger BERT model called DistilBERT comes with a more compact and faster architecture and yet achieves strong results across several natural language tasks [21].

This study introduces a tree-based method, evtree, which leverages the EA paradigm to construct an ensemble-based spam detection model. While ensemble learning and evolutionary approaches have been explored in prior research, evtree offers a refined mechanism for autonomously selecting and integrating multiple base classifiers to increase performance. The key contributions of this study are as follows:

1. We assess the effectiveness of five independent supervised classification algorithms in identifying spam emails, providing insights into their individual performance. In addition, to provide a comparison with deep learning methods, we use the DistilBERT model as a benchmarking reference.
2. We propose an evolutionary ensemble learning framework (evtree) tailored for spam detection, enabling the adaptive formulation of ensemble models.
3. Comparative evaluations demonstrate that the evtree-based ensemble consistently outperforms its corresponding single-base classifiers across supervised and ensemble settings (i.e., averaging and majority voting). However, while performance improvements are observed, they align with incremental advancements reported in prior literature on evolutionary-based ensemble learning.
4. To validate our methodology, experiments were conducted on two widely recognized spam benchmark datasets. The results confirm the robustness and effectiveness of the evtree approach in spam classification. Furthermore, an ablation study was carried out by systematically excluding each of the base models used in the ensemble to evaluate their individual contributions to the overall performance of evtree.

Section 2 provides an overview of existing spam detection approaches in related works. Section 3 outlines the Enron dataset and its preprocessing for feature extraction. Sections 4 and 5 present the nonensemble and ensemble learning paradigms, demonstrating the combination of base classifiers via evtree. The detailed results of the analysis of various algorithms and the probable explanations for our findings are presented in Section 6. The final component comprises the conclusion and prospective endeavors.

2. RELATED WORK

The problem of email spam detection has been studied intensively, and different approaches have been introduced to enhance the classification performance. Some of the research papers are more specific to particular types of methods, ranging from basic machine learning methods to layered models that include deep learning.

The classification model for spear-phishing emails by Zuhdi et al. [22] does not depend on headers, bodies or attachments but rather on content-based features. In their study, they incorporated a new feature known as the 'Email contain Position', which was used to distinguish spear-phishing emails from the ensemble from the Enron dataset. They used random forest and naïve Bayes classifiers: although both were successful, random forest had a higher area under the curve (AUC) of 99.6% and F1 of 96.8%. On the other hand, the naïve Bayes classifier used in this experiment reflected a moderate level of accuracy, which suggested that decision tree-based models are more effective than others for classifying spear-phishing via content-based features. This demonstrates that feature engineering, alongside selecting specific classifiers, can help increase the accuracy of email classification, which is the objective of the proposed approach using a stacked ensemble with evolutionary algorithms.

In the same manner, Junnarkar et al. [23] developed an ML approach to assess email spam classification and NLP. Their solution included text classification that was previously supplemented with URL-based filtering; they applied metrics that include random forest and naïve Bayes filtration. Their work focused on identifying the need for semantic-based text classification along with URL filtering as an additional layer that can enhance multiple-layer security for spam detection systems. The random forest yielded a high accuracy of 97.83%, whereas naïve Bayes obtained a mean accuracy of 91.11%, revealing the efficiency of augmenting the system with semantic-based text classification for spam detection via a URL filter. A more detailed comparison can be made with one of their feature sets: they employed both the content of the site and the URL for their model, which is similar to the feature mixture used in this work because the greater difference between features and algorithms makes the result more accurate.

Douzi et al. [24] presented a hybrid spam detection model that relies on deep learning together with the paragraph vector-distributed memory (PV-DM) model. As mentioned previously, previous works applied conventional approaches such as bag-of-words (BOW) and recognized some issues in the original model, which failed to express the order or context of the words, reducing the performance of the model when trying to focus on the semantics of the words. To address this issue, Douzi et al. suggested extending the work of the PV-DM with a context-constructing model to generate a context-aware representation of emails and combined the improved approach with BOW. Their hybrid model, which they used in this scenario, yielded good results compared with the baseline BOW and PV-DM models on the Enron dataset, where it yielded 95.88%, whereas on the Ling dataset, it yielded 98.27%. Its model easily outperforms their previous methods in terms of both the accuracy and classification of the Enron and Ling spam sets.

Tida and Hsu [25] presented a spam detection framework named the Universal Spam Detection Model (USDM) derived from transformer technology used by Google. A pretrained BERT model from four datasets was adapted by their system, which included SpamText and Enron. The model reached extraordinary success, with an overall accuracy of 97% and an F1 score of 96%, which illustrates the power of transformer models in immediate spam identification across different datasets.

Guo et al. [26] constructed a spam detector model that integrates the BERT transformer architecture with ML algorithms, namely, logistic regression and support vector machines. The model demonstrated exceptional accuracy, with logistic regression exceeding all compared classifiers and scoring 97.84% on the Enron-Spam dataset. Improving the classification of spam is highly dependent on transformers' ability to gather context-rich information within texts.

Zavrak and Yilmaz [27] used the hierarchical attention hybrid neural network (HAN) algorithm and the FastText (FT) model for email spam detection, combining convolutional neural networks, gated recurrent units and attention mechanisms. Their approach selects which parts of the email text to pay particular attention to to to improve classification accuracy. Their model is based on combining convolution layers for extracting useful features with GRUs for sequence processing and performs well across several datasets, such as TREC 2007, Enron, and SpamAssassin, with a very high AUC of 99.9% being achieved on the TREC 2007 dataset.

In [2], Nicholas and Nirmalrani developed a hybrid model that utilized bioinspired algorithms to optimize deep learning models efficiently. During each epoch, the algorithm refined weight optimization by using the sand cat swarm optimization (SCSO) technique to address issues with high-dimensional data challenges. After data processing, BoW was used for extracting features and optimal feature selection. Using a dataset of email spam from the UCI machine learning repository, the hybrid model achieved 92.50% accuracy.

In conclusion, the studies mentioned above reflect various methodologies and developments in the implementation of ML/DL techniques for spam email classification. To improve the recall, a stacking ensemble method is introduced to combine the classifiers from the multiple base classifiers, considering the requirement of managing the dynamic spam techniques. We validate the effectiveness and generalizability of our approach via several experimental evaluations across different datasets, which are composed of diverse data distributions, along with various further experiments. This model has higher recall and addresses the limitations found in individual models, thereby improving the overall performance. By establishing a new baseline for counterfactual classifiers and considering the ability to retrieve spam emails as part of a metric of classifier performance, we demonstrate the potential of our model to improve spam email classification accuracy.

3. DATASET PREPROCESSING AND FEATURE EXTRACTION

Before this study, two widely used datasets considered benchmarks in spam investigations were adopted: Enron [28] and SpamAssassin [29]. Enron has 33,715 emails, 50.92% of which are spam, whereas 31.35% of SpamAssassin's 5,795 emails are spam. In the preprocessing step, each text column contained the subject and content of that email. A series of text data preprocessing operations are performed on the datasets employed in the Natural Language Toolkit (NLTK). The first operation converts all the text to lowercase and removes special characters and multiple spaces. The next step is to eliminate stop words from the text via a predefined set of stop words from the NLTK library. Adapting data constitutes a vital first step in NLP initiatives [30]. Previous studies have investigated various NLP approaches for feature extraction from emails aimed at improving spam detection through ML techniques. According to [31], the most commonly adopted methods include bag-of-words (BoW), N-grams, term frequency-inverse document frequency (TF-IDF), and word embeddings such as word2vec. TF-IDF has demonstrated strong performance in spam email classification, offering both computational efficiency and meaningful word importance weighting [5]. Therefore, this paper adopts TF-IDF for feature extraction. This technique used for assessing the relevance of a word in a document versus a set of documents or a corpus is the aim of the statistical indicator TF-IDF [32]. It consists of two elements: we measure both term frequency (TF) and inverse document frequency (IDF) [33]. TF indicates how many times a particular word occurs in a document. This increases the importance of words that are used frequently. The rarity of a term within the corpus is indicated by the IDF component (refer to)34]) such that the TF-IDF weight for word i in document j is given by:

$$w_{ij} = tf_{ij} * \log\left(\frac{N}{df_i}\right), \tag{1}$$

where tf_{ij} is the number of times word i appears in document j divided by the total number of words in document j; N is the total number of documents in the corpus; and df_i is the number of documents in the corpus that contain word i.

```
Algorithm 1: TF/IDF Algorithm [35]
Input: $R$ is the set of prediction rules (ranking tables)
       $N$ is the set of documents
Output: F is the output file (table of ranking, $tc$, $tf, M, idf$, and $(tf - idf)$)
1. Begin
2. String Matching (R, N)
3. $F = \varnothing$
4. For each rule r in the prediction rule base R Do
5.     For each example n in $\in N$
6.         If r refers to n (R, N)
7.             Add information to F
8. Return F
9. End
```

Algorithm 1 shows how TF/IDF works. By using Python, TF-IDF values can be calculated via the $TfidfVectorizer()$ method in the sklearn module. TfidfVectorizer is a feature extraction method used to convert a collection of raw documents into a matrix of TF-IDF features. When using the $TfidfVectorizer$, 4500 max TF-IDF features are employed in this particular work. The general steps are as follows:

1- The dataset was downloaded, followed by a thorough review of its contents.
2- A pre-processing step is applied to remove stop words from the dataset via NLTK. This step involves utilizing NLTK's built-in list of common stop words, which includes 40 words, such as "a", "an", "the", and "of", with attention given to some words that may affect the meaning change of the emails. By removing these stop words, we aim to increase the quality of the data by focusing on meaningful words and reducing noise in the dataset. This process is beneficial for researchers and practitioners working with NLP tasks.
3- We call $TfidfVectorizer$ by sting the max feature to 4500.
4- The $tf - idf$ values are obtained by applying the $fit\_transform()$ method to the training and testing datasets.
5- The result in step 4 is used as the input to the proposed classifier.

4. CLASSIFIER MODELS

4.1 Cubist Algorithm

Cubist is a rule-based model that is an extension of Quinlan's M5 model tree. It is a powerful classifier that combines decision trees with linear models. This hybrid approach manages to handle complex datasets by capturing nonlinear relationships and linear trends. Cubist splits the dataset into multiple partitions via decision trees. For each leaf, a linear regression model is fit. To smooth the transitions between these linear models, Cubist uses a linear combination of the current model's prediction, $y_{(k)}$, and the parent model's prediction, $y_{(pd)}$, as follows:

$$y_{par} = a \times y_{par} + (1 - a) \times y_{par}, \tag{2}$$

where a is a weighting factor. These models make predictions on the basis of the rules derived from the decision tree structure.
[36], [37].

4.2 Naïve Bayes (NB)

As a probability classifier that uses Bayes' theorem and assumes feature independence on the basis of class labels, naïve Bayes performs its function. Although this model achieves great success with extensive datasets, it still functions appropriately when the independence assumption is neglected. This model appeals to many users for its straightforward application and its ability to classify texts and identify spam in different machine learning scenarios [38]. However, the naïve Bayes algorithm is relatively impervious to the violation of these assumptions in real-world scenarios. In addition, it is among the best algorithms in data mining. In this case, let C represent the class in which a given observation X belongs. To predict the class of this observation, the highest posterior probability must be calculated according to Bayes' theorem:

$$P(C|X) = \frac{P(C)P(X|C)}{P(X)}, \tag{3}$$

where $P(C)$, $P(X|C)$, and $P(X)$ denote the prior probability, likelihood, and evidence (probability of X), respectively. Under the assumption that the features $X1, X2, X3, \dots, Xn$ are conditionally independent given the class, the equation becomes:

$$P(C|X) = P(C) * \frac{\prod P(Xi|C)}{P(X)}, \tag{4}$$

where $P(C)$ is the prior probability, the product of all likelihoods is $\prod P(X|C)$, and $P(X)$ is the evidence.

4.3 Support Vector Machine (SVM)

Support vector machines (SVMs) provide strong classification and regression tools. In a large-dimensional space, SVMs define a hyperplane to distinguish different classes of data effectively while maximizing the margin between them. This is achieved by utilizing a linear model of the form:

$$y(x) = w^T \phi(x) + b, \tag{5}$$

where $\phi(x)$ denotes a fixed feature-space transformation, w is the weight vector, and b is the bias parameter. By using a range of kernel functions, this strategy addresses nonlinear relationships efficiently, making SVMs particularly useful for intricate datasets. Many SVMs are recognized as having excellent generalization abilities, and they commonly appear in domains such as bioinformatics and image recognition [38].

4.4 Recursive Partitioning Tree (part)

A recursive algorithm in the recursive partitioning tree (rpart) splits the dataset into groups on the basis of feature values. This approach seeks to establish a tree framework that precisely forecasts the target variable by ensuring the equality of the generated subsets [36], [39]. The standard Gini index offers a useful interpretation related to misclassification. Consider selecting an object from one of C classes with probabilities $(p_1, p_2, \dots, p_C)$ and randomly assigning it to a class via the same probability distribution. The probability of misclassification is given by:

$$\sum_i \sum_{j \neq i} p_i p_j = \sum_i \sum_j p_i p_j - \sum_i p_i^2 = 1 - \sum_i p_i^2, \tag{6}$$

This final expression, $1 - \sum_i p_i^2$, is the Gini index for the probability distribution p. Now, let $L(i,j)$ represent the loss incurred when an object is assigned to class j while it actually belongs to class i. The expected cost of misclassification can be expressed as $\sum_i \sum_j L(i,j) p_i p_j$. This motivates the definition of the generalized Gini index of impurity as follows:

$$G(p) = \sum_i \sum_j L(i,j) p_i p_j, \tag{7}$$

This generalized index is promising for applications where misclassification costs vary. However, it suffers from certain drawbacks. First, $G(p)$ is not necessarily a concave function of p, which is a desirable property for impurity measures. Second, G effectively symmetrizes the loss matrix. This is evident from the equivalent form:

$$G(p) = \frac{1}{2} \sum_i \sum_j [L(i,j) + L(j,i)] p_i p_j, \tag{8}$$

In particular, for two-class problems, G essentially ignores the loss matrix, rendering it less useful in scenarios where the misclassification costs between the two classes are significantly different.

4.5 Conditional inference trees (ctree)

This statistical technique named Conditional Inference Trees (ctree) resolves issues with traditional tree models, especially in dealing with bias and overfitting. This approach selects splits within a statistical context for a tree that has both a sound structure and an interpretable form [36], [40].

$$P(Y = y | X = x) = \left(\sum_{i=1}^{n} wi\,(x) \right)^{-1} \sum_{1=1}^{n} wi(x)\, I(Y_i = y), y = 1, \dots, J, \tag{9}$$

where:

- $P\hat{}(Y = y \mid X = x)$ is the estimated conditional probability that the response variable Y equals y given that the covariate X equals x.
- $wi(x)$ is a weight function for the $i - th$ observation, which is dependent on the covariate value x.
- $I(Yi = y)$ is an indicator function that equals 1 if the response variable Y for the $i - th$ observation is equal to y and 0 otherwise.
- n is the total number of observations.
- y represents one of the J classes of the response variable Y.
- J is the total number of classes in the response variable Y.
- x represents the given value of the covariate X.

5. ENSEMBLING CLASSIFIERS USING EVTREE

Evolutionary algorithms offer an effective foundation for the independent creation of computer programs [17]. An evolutionary algorithm paradigm begins with the creation of a set of solutions (i.e., population) that could address the problem. Solutions are subsequently assessed to ascertain the effectiveness of their problem-solving abilities or their closeness to a solution. The evaluation of a person's closeness to resolving the issue is denoted by a 'fitness function'. The simulated evolutionary process seeks to cultivate progressively optimal populations of candidate solutions, utilizing operators driven by biology concepts, including reproduction, which involves crossover, mutation, and selection. The three evolutionary operators are used iteratively until the stopping criteria are met. Crossover entails the 'mating' of solutions, whereby two individuals exchange components to generate children. Mutation allows a component of a solution to undergo spontaneous modification, enabling it to adopt values that may not have been previously enclosed by any candidate solution within the population. Ultimately, selection determines which solutions progress to construct the successive generations of the population. This creates a 'survival of the fittest' framework, in which superior solutions have more chances of progression. By continually following the established cycle, it is predicted that future generations will yield more fit solutions, ultimately resulting in at least one that addresses the issue or is considered acceptable. The fundamental procedures of an EA are delineated in Algorithm 2:

```
Algorithm 2: Evolutionary Algorithm [13]
Input: input dataset
Output: the best solution
 1. Start
 2. Create randomly the initial population
 3. Assess the quality of each solution through using the fitness function.
 4. While termination conditions not met: Do
 5. Assess the fitness of each solution in the population.
 6. Apply genetic operators to the solutions
 7. Produce a new population
 8. End While
 9. Return the best solution
10. End
```

This research aims to create a classification tree that more accurately identifies spam. evtree is employed as an evolutionary algorithm for the categorization of trees. The tree algorithm is implemented via an evolutionary approach grounded in Darwinian concepts. Grubinger et al. first introduced it in 2014 [41]. evtree generates candidate solutions in the form of trees and systematically modifies them after each generation to improve the classification tree for the specific environment. The root nodes are initialized via a random, valid splitting criterion in the initial phases of population development. The assessment function reflects the requirements of the population. Individuals are evaluated according to their misclassification rate and complexity, as determined by the evtree algorithm, specifically the utilized terminal nodes. The aim is to increase accuracy in the training data while minimizing complexity. The effectiveness of a classification tree is primarily assessed through its misclassification rate (MC), whereas its complexity is determined by the number of terminal nodes (M). evtree employs $2N \cdot MC(Y, f(X, \theta))$ as its loss function. The complexity of trees is measured by the number of terminal nodes, adjusted by $log\ N$ and a user-defined parameter α. Equation (2) aims to identify trees θ that minimize misclassification loss while balancing a BIC-type trade-off with the number of terminal nodes [41]. Consequently, it may generate effective solutions with decreased complexity, which is beneficial for resource-constrained settings.

$$\begin{aligned} loss\big(Y, f(X,\theta)\big) &= 2N.MC\big(Y, f(X,\theta)\big) \\ &= 2.\sum_{n=0}^{N} I\big(Y_n \neq f(X_n,\theta)\big), \\ comp(\theta) &= \alpha\,.M\,.\,log\,N \end{aligned} \tag{10}$$

Genetic operators modify the tree at each iteration of the $evtree$ algorithm, which employs one crossover operator and four types of mutation operators. The crossover operation generates two new trees by recombining two subtrees extracted from randomly selected trees. Mutation involves several processes:

- Split: A terminal node is randomly selected and assigned a valid, randomly generated split rule, converting it into an internal node with two new terminal nodes.
- Pruning: A random internal node with two terminal children is selected and pruned, becoming a terminal node.
- Major split rule mutation: A random internal node is chosen to modify both the split variable and the split point.
- Minor Split Rule Mutation: This mutation is similar to the major split rule mutation but with only slight modifications to the split point.

The newly generated trees are reintegrated into the population, and this process continues until a predefined termination criterion, such as a set number of generations, is met.

This paper involves training ML algorithms on a uniform dataset to develop base models. Initially, multiple models are trained via the dataset, and their predictions are combined to form a final decision. In the second phase, evtree is applied to construct the stacked model. The stacked models are represented as trees, with each solution illustrating a possible configuration of the available base learners. Acquiring the stacking procedure via the suggested method may result in nonlinear combination mechanisms that can more effectively utilize the outputs of various learners. Stacking systems that utilize weighting or majority voting strategies, which linearly integrate several learners, are not the most effective option for ensemble construction [38]. In the stacking process, we employed the prediction probabilities of the base learners to train the event tree (i.e., meta-learner) rather than the prediction labels. evtree performs two operations during the assembly of stacked models: picking members from a pool of base models and combining the selected base models (i.e., fusing the base models). Hence, the members of the stacked models are automatically selected instead of a predetermined size (i.e., fixed). The stacking method has demonstrated efficacy in overcoming numerous machine learning challenges. However, extra time is needed because of the requirement for meta-learning training. To mitigate this limitation, the training portion is partitioned into 80% for training base learners and 20% for training the stacking algorithm (i.e., evtree). The computing effort is manageable when the stacked model is reconstructed in response to data alterations. The overall structure of the proposed framework utilizing the evtree method is illustrated in Figure 1. The algorithm is given in Algorithm 3.

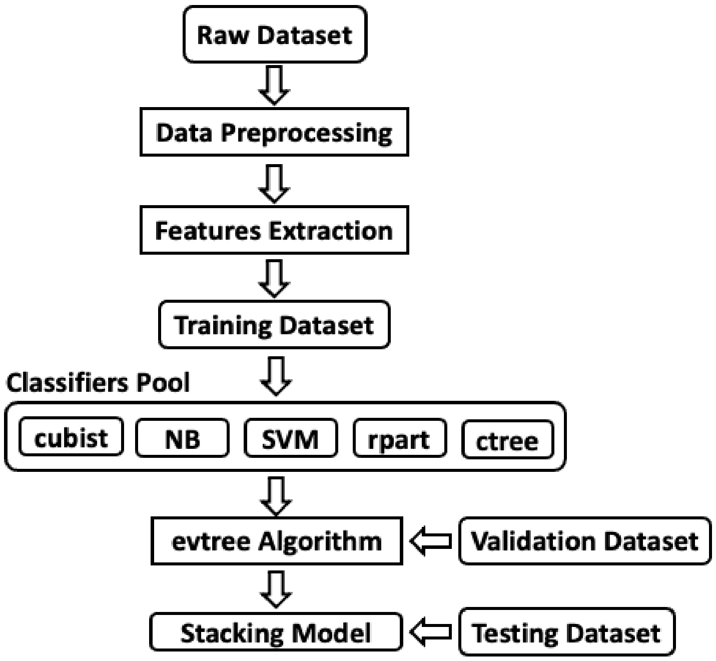

Fig.1 Workflow of the proposed stacking schema

Algorithm 3: Stacking using evtree

Input: train_X: train attributes, train_y: train label, test_X: test attributes, test_y: test label
Output: best performing stacked tree

```
        base_classifiers = ['cubist, 'NB, 'SVM, 'rpart, 'ctree']
 1.  Step 1: learn base classifiers
 2.  foreach classifier ∈ base_classifiers do
 3.      Train classifier on train_X and train_y
 4.      Examine classifier performance using test_X and test_y
 5.      Save performance metrics
 6.  end
 7.  Step 2: formulate a training set for meta-learning classifier.
 8.  foreach classifier ∈ base_classifiers do
 9.      Obtain probability = a list of predicted probability on 20% of train_X using classifier
10.      Attach probability in new_dataset
11.  end
12.  Append the 20% train_y in new_dataset
13.  Step3: learn a meta-classifier
14.  Run evtree algorithm on new_dataset to stack base_classifiers as a tree
15.  Return best performing stacked tree
```

6. EXPERIMENT DETAILA AND RESULTS

The studies were conducted on a 2.9 GHz Intel Core i7 with 8 GB of RAM operating on macOS Mojave. The detectors were developed via open-source machine learning packages in R, specifically Cubist, naïvebayes, rpart, e1071, partykit, and evtree. Upon training evtree, the standard metrics for assessing the performance of the evolved tree include accuracy, recall, precision, and the F1 score. These measurements are derived from a confusion matrix that encompasses the four types of outputs from a binary classifier. In this context, the classifications are as follows: (1) true positive (TP) indicates how a model accurately detects a spam email; (2) true negative (TN) indicates the correct detection of a legitimate email; (3) false positive (FP) indicates the erroneous classification of a legitimate email as spam; and (4) false negative (FN) reveals that the model mistakenly categorizes a spam email as legitimate. The performance score can be computed on the basis of the aforementioned four values as follows:

1. Accuracy [(TP + TN)/Total]: the proportion of correct classification determinations.
2. Recall [TP/(FN + TP)]: the metric indicating the proportion of spam emails accurately identified by the model. Recall is often referred to as the detection rate.
3. Precision [TP/(FP + TP)]: the metric assessing the frequency with which the model accurately identifies spam emails.
4. F1 score [2 * (accuracy ∗ Recall)/(Precision + Recall)]: the harmonic mean of accuracy and recall.

.

6.1 Experimental Results

To evaluate the proposed schema on spam datasets, we report the performance of trained models—both individual classifiers and stacking—on the testing dataset. We employed a 5-fold cross-validation technique, where the reported results represent the average of five runs along with the standard deviation (SD).

Table 1 presents key classification metrics from the testing phase, including accuracy, recall, precision, and F1 score, with the best results highlighted in bold. The Cubist model achieved the highest accuracy, closely followed by evtree. In terms of spam detection, evtree demonstrated superior performance. Additionally, the highest test precision (98.36) and F1 score (98.42) were obtained by evtree, whereas the lowest values (83.25 for precision and 86.57 for F1 score) were recorded for the average and majority vote techniques. In contrast, the simpler models, such as NB, average, and majority vote, consistently underperformed across all the metrics compared with the other classifiers. Table 1 highlights the strength of stacking via evtree, which effectively integrates the strengths of multiple base learners to achieve better spam detection performance. Overall, the results show that while individual classifiers, particularly SVM and DistilBERT, are strong performers, carefully designed stacking methods can increase performance even more.

TABLE I. TECHNIQUES PERFORMANCE FOR THE ENRON DATASET (%). ± REFERS TO THE STD VALUE. THE BETTER RESULTS ARE IN BOLD TEXT.

Technique	Accuracy	Recall	Precision	F1-score
Cubist	**98.42 ± 0.003**	98.23 ± 0.01	93.16 ± 0.007	95.63 ± 0.005
NB	93.17 ± 0.009	93.58 ± 0.004	93.08 ± 0.025	93.33 ± 0.015
SVM	97.57 ± 0.001	97.86 ± 0.005	97.40 ± 0.004	97.63 ± 0.002
rpart	90.58 ± 0.006	96.51 ± 0.017	86.56 ± 0.009	91.26 ± 0.009
ctree	92.94 ± 0.006	97.69 ± 0.016	89.44 ± 0.012	93.38 ± 0.007
DistilBERT	97.70 ± 0.005	97 ± 0.004	97.60 ± 0.025	97.24 ± 0.005
Average	85.75 ± 0.011	90 ± 0.016	83.35 ± 0.025	86.57 ± 0.016
Majority Vote	85.88 ± 0.011	90.53 ± 0.016	83.25 ± 0.025	86.74 ± 0.017
evtree Stacking	98.39 ± 0.003	**98.49 ± 0.012**	**98.36 ± 0.011**	**98.42 ± 0.005**

The performance metrics tested on the SpamAssassin dataset for our proposed system are presented in Figure 2 and are shown in graphical form. Each bar plot represents the average result over five runs, with an error bar on top indicating the SD around the average. Among the evaluated models, SVM achieved the highest performance in terms of accuracy, precision, and F1 score, whereas evtree obtained the best recall score. In contrast, the stacked models based on merging without learning delivered the lowest performance across all the metrics when tested on the SpamAssassin dataset.

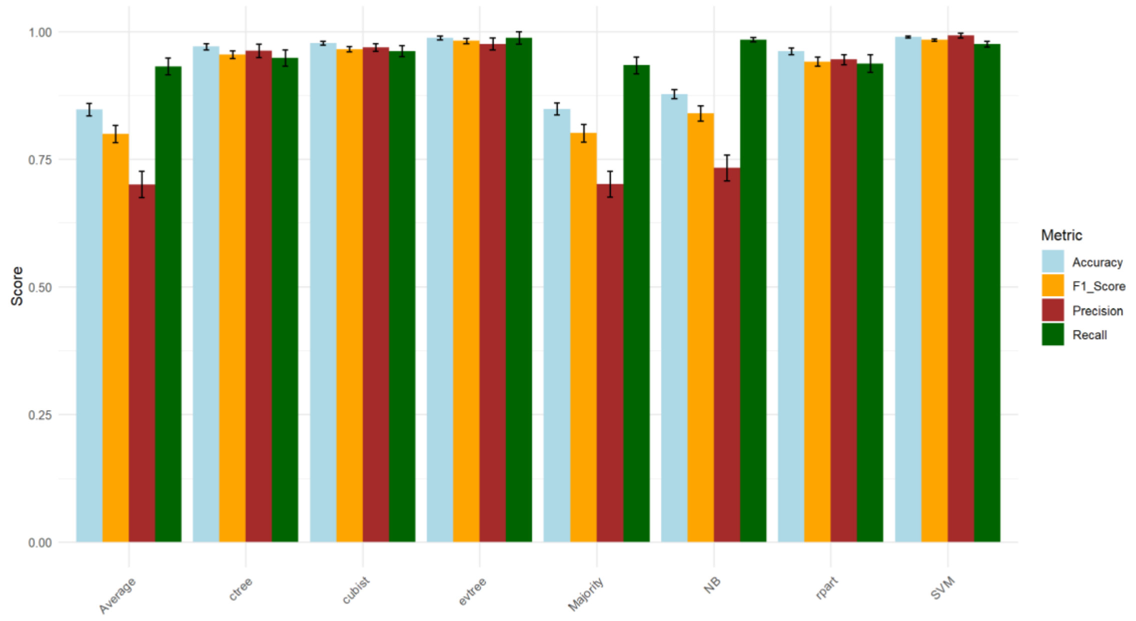

Fig. 2 Barplots for implemented technique performance for the SpamAssassin dataset with an error bar on top, which shows the SD around the average

Certainly, the ability to produce human-readable outputs helps explain the decision-making process. Figure 3 shows one of the best trees assembled via the evtree algorithm. This tree evolved using 2 base classifiers only, with a total size of 7 nodes. This tree is based on probabilities from the SVM and Cubist models. The first split occurs at svm_Prob = 0.269, followed by further splits based on cubist_Prob = 0.632 and another svm_Prob threshold of 0.96. The leaf nodes display bar charts representing the proportions of class 0 and class 1 nodes, with darker bars indicating a greater presence of class 1 nodes. The tree automatically selects and assembles base classifiers to form a stacking classifier with the best spam email detection ability models dynamically instead of stacking all of them. Hence, the computational cost when deployed for such a classifier is reduced.

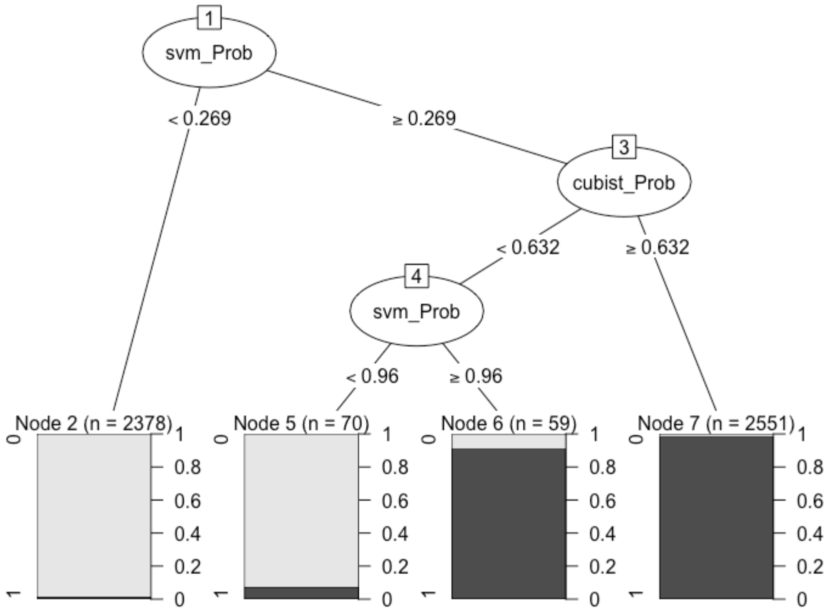

Fig 3. Tree for spam email detection evolved by evtree

Figure 4 presents a bar chart comparing the performance of a stacked ensemble model before and after performing an ablation analysis on each of its five base learners. Ablating the learners, namely, ctree, rpart, or naïve Bayes, results in only marginal declines across all performance metrics. In contrast, the removal of SVM and Cubist leads to the most significant drop in performance, indicating that while all the models contribute to the ensemble's effectiveness, the SVM is the most critical component.

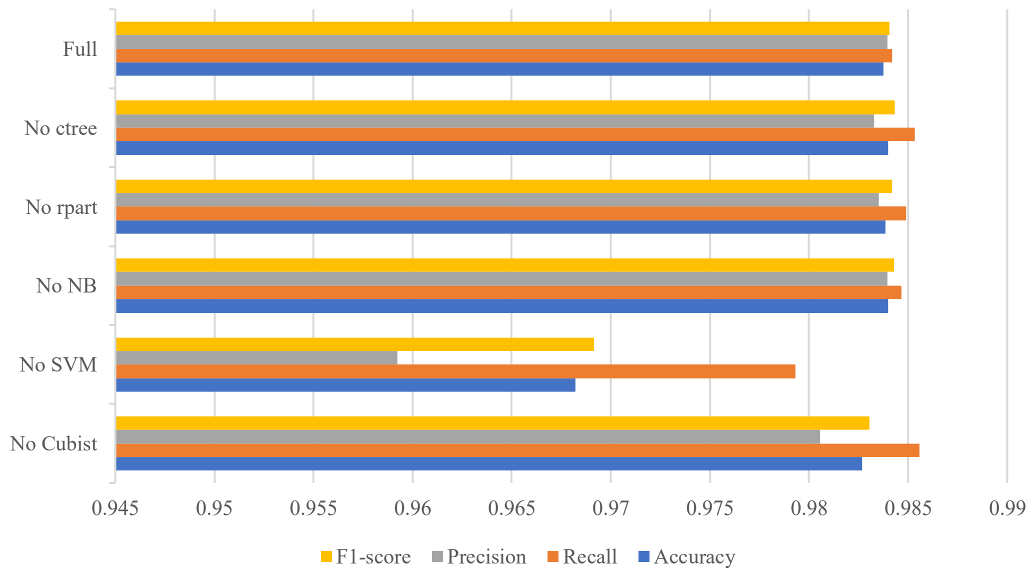

Fig. 4 Comparative Analysis of the Ablation Study

To further examine how each base model contributes to individual email classification outcomes, we selected two emails from the testing dataset, one labelled spam and the other labelled nonspm, for illustration. We computed breakdown profiles via the DALEX package. DALEX is a model-agnostic tool, meaning that it can extract valuable insights from any predictive model regardless of its internal mechanics [10]. Figure 5 displays the breakdown profiles, showing how the evtree meta-learner integrates predictions from various base models (SVM, Cubist, NB, rpart, and ctree) to generate its final decision. On the left (representing the ham email), the baseline (intercept) begins at approximately 0.517, but the SVM's strongly negative contribution (with a probability near zero) dominates, lowering the final prediction to approximately 0.014. The other models exert little to no noticeable influence. On the right (representing the spam email), the SVM probability is close to one, providing a strong positive effect (+0.44), whereas Cubist adds a modest positive contribution (+0.03), raising the final output to approximately 0.986. Again, NB, rpart, and ctree contribute insignificantly. Overall, these plots emphasize that evtree relies heavily on SVM predictions in both cases, largely disregarding the other base models. This finding indicates that SVM plays the most influential role in the ensemble's decision-making for these examples.

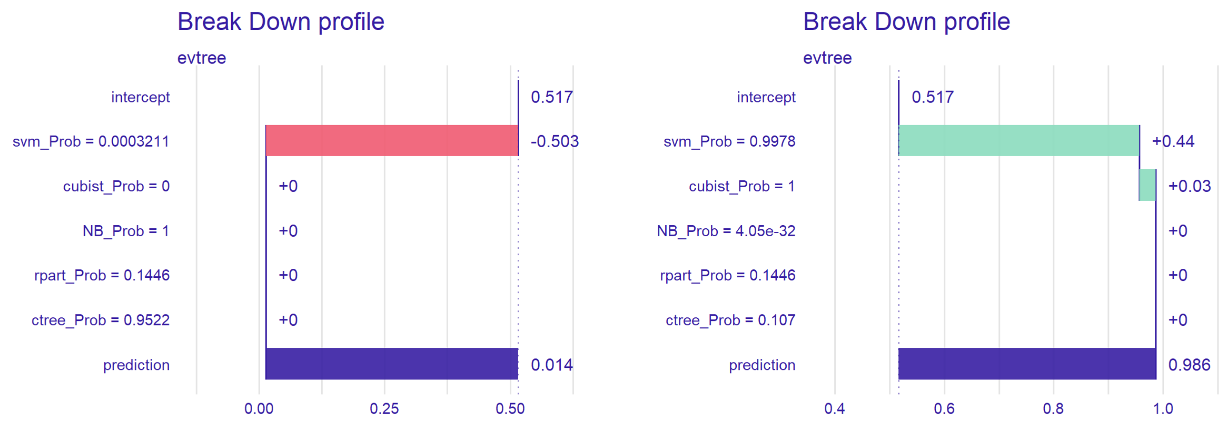

Fig. 5 This figure was made with the DALEX package for explaining evtree model predictions relying on base models where the left instance (ham) and right instance (spam)

Table 2 compares the proposed approach with recently published works using ML and DL studies that performed experiments on the same datasets. According to the outcomes, the proposed approach is comparable and, in some metrics, even outperforms the alternatives. As shown in Table 2, evtree outperforms various published classification techniques, including ensemble, hybrid, and deep learning approaches, on the Enron dataset, which is recognized for its comprehensiveness and up-to-date sample. Although the work in [25] achieved higher recall than evtree on the SpamAssassin dataset did, evtree outperformed it in all other metrics. Furthermore, the suggested algorithm automatically constructs ensemble models without the constant need for manual intervention. Hence, the research presented here reduces highly computationally intensive training demands, which require substantial resources.

TABLE II. COMPARISON OF THE PROPOSED APPROACH WITH THE ML AND DL METHODS (%).

Ref.	Year	Method	Dataset	Accuracy	Recall	Precision	F1-score
Douzi et al. [24]	2020	PV-DM + SVM	Enron	96.16	95.59	96.55	96.07
Junnarkar et al. [23]	2021	Gensim + SVM	Enron	97.83	95	97.5	95.5
Tida and Hsu [25]	2022	BERT	Enron	97	93	96	97.2
			SpamAssassin	98	99	96	97.64
Zavrak and Yilmaz [27]	2023	FT + HAN	Enron	95.8	93.7	98.1	95.8
			SpamAssassin	95.5	97.8	89.3	93.3
Zuhdi et al. [22]	2024	Random Forest	Enron	NA	96.7	96.9	96.8
This study	2024	Stacking	Enron	98.39	98.49	98.36	98.42
			SpamAssassin	98.79	98.76	97.6	98.17

6.2 Limitations

Despite its strong effectiveness in detecting spam emails, the evtree model is currently limited to this specific application (i.e., email spam). To broaden its applicability, future work should extend the approach to other sources of similar threats, such as social network spam and SMS spam. Moreover, while the stacking system enhances the classification accuracy, it also increases the detection time because of the use of multiple models. Optimizing the system through feature selection techniques is essential for reducing model complexity and improving efficiency. Finally, although the proposed system effectively mitigates computational demands by leveraging appropriate algorithms, further refinements may be necessary to ensure scalability and robustness in real-world deployments.

7. CONCLUSION AND FUTURE WORK

Spam emails have become a favoured technique for cybercriminals to collect sensitive information. Techniques based on machine learning algorithms are promising for preventing such threats, and this research is not yet mature. The models are validated via K-fold cross-validation, which is assessed through accuracy, recall, precision, and the F1 score. By conducting comprehensive analyses and benchmarking against state-of-the-art models, this research seeks to advance the field of email spam detection. In this study, we used the evtree algorithm to evolve stacking solutions to address the issue of recognizing spam emails. The suggested approach is useful since it gives security teams a less complex and effective detector. The ensemble arising via the proposed technique permits the automatic construction of detectors. These results showed that the suggested stacking approach is more profitable than nonensemble techniques and published works using examined datasets.

We plan to extend our evaluation as part of our future work in several ways, such as implementing feature selection techniques that can be implemented to address high-dimensional datasets. For further testing, we are planning to consider existing models with different types of cybersecurity threats.

Conflicts of interest

The authors declare that they have no conflicts of interest.

Funding

There is no funding for the paper.

Acknowledgement

None.

References

[1] L. Ceci, “Emails sent per day 2025,” Statista. Accessed: Sep. 29, 2024. [Online]. Available: https://www.statista.com/statistics/456500/daily-number-of-e-mails-worldwide/

[2] N. N. Nicholas and V. Nirmalrani, "An enhanced mechanism for detection of spam emails by deep learning technique with bio-inspired algorithm," *e-Prime - Advances in Electrical Engineering, Electronics and Energy*, vol. 8, 2024, doi: 10.1016/j.prime.2024.100504.
[3] A. Petrosyan, "Spam e-mail traffic share 2023," Statista. Accessed: Sep. 29, 2024. [Online]. Available: https://www.statista.com/statistics/420400/spam-email-traffic-share-annual/
[4] M. Raza, N. D. Jayasinghe, and M. M. A. Muslam, "A Comprehensive Review on Email Spam Classification using Machine Learning Algorithms," in *International Conference on Information Networking*, 2021, pp. 327–332. doi: 10.1109/ICOIN50884.2021.9334020.
[5] K. Yusupov, M. R. Islam, I. Muminov, M. Sahlabadi, and K. Yim, "Comparative Analysis of Machine Learning and Deep Learning Models for Email Spam Classification Using TF-IDF and Word Embedding Techniques," *Lecture Notes on Data Engineering and Communications Technologies*, vol. 231, pp. 114–122, 2025, doi: 10.1007/978-3-031-76452-3_11.
[6] L. Breiman, "Bagging predictors," *Mach Learn*, vol. 24, no. 2, pp. 123–140, 1996, doi: 10.1007/bf00058655.
[7] D. H. Wolpert, "Stacked generalization," *Neural Networks*, vol. 5, no. 2, pp. 241–259, Jan. 1992, doi: 10.1016/S0893-6080(05)80023-1.
[8] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, Springer Verlag, 1995, pp. 23–37. doi: 10.1007/3-540-59119-2_166.
[9] R. Meyes, M. Lu, C. W. de Puiseau, and T. Meisen, "Ablation Studies in Artificial Neural Networks," Jan. 2019, Accessed: May 02, 2025. [Online]. Available: https://arxiv.org/pdf/1901.08644
[10] B. Przemyslaw, "DALEX: Explainers for complex predictive models in R," *Journal of Machine Learning Research*, vol. 19, pp. 1–5, 2018, Accessed: May 25, 2025. [Online]. Available: http://www.jmlr.org/papers/v19/18-416.html
[11] G. Kumar and K. Kumar, "The Use of Artificial-Intelligence-Based Ensembles for Intrusion Detection: A Review," *Applied Computational Intelligence and Soft Computing*, vol. 2012, pp. 1–20, 2012, doi: 10.1155/2012/850160.
[12] G. Folino and P. Sabatino, "Ensemble based collaborative and distributed intrusion detection systems: A survey," 2016. doi: 10.1016/j.jnca.2016.03.011.
[13] S. Sen, "A Survey of Intrusion Detection Systems Using Evolutionary Computation," in *Bio-Inspired Computation in Telecommunications*, 2015, pp. 73–94. doi: 10.1016/B978-0-12-801538-4.00004-5.
[14] H. Alyasiri, J. A. Clark, and D. Kudenko, "Evolutionary computation algorithms for detecting known and unknown attacks," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, Springer Verlag, 2019, pp. 170–184. doi: 10.1007/978-3-030-12942-2_14.
[15] S. Khanchi, M. I. Heywood, A. Vahdat, and A. Nur Zincir-Heywood, "On botnet detection with genetic programming under streaming data, label budgets and class imbalance," in *GECCO 2018 Companion - Proceedings of the 2018 Genetic and Evolutionary Computation Conference Companion*, 2018, pp. 21–22. doi: 10.1145/3205651.3208206.
[16] T. A. Pham, Q. U. Nguyen, and X. H. Nguyen, "Phishing attacks detection using genetic programming," in *Advances in Intelligent Systems and Computing*, Springer Verlag, 2014, pp. 185–195. doi: 10.1007/978-3-319-02821-7_18.
[17] H. Alyasiri, J. A. Clark, A. Malik, and R. De Frein, "Grammatical Evolution for Detecting Cyberattacks in Internet of Things Environments," in *Proceedings - International Conference on Computer Communications and Networks, ICCCN*, Institute of Electrical and Electronics Engineers Inc., Jul. 2021. doi: 10.1109/ICCCN52240.2021.9522283.
[18] F. G. Abdulkadhim, Z. Yi, A. N. Onaizah, F. Rabee, and A. M. A. Al-Muqarm, "Optimizing the Roadside Unit Deployment Mechanism in VANET with Efficient Protocol to Prevent Data Loss," *Wirel Pers Commun*, vol. 127, no. 1, pp. 815–843, Nov. 2022, doi: 10.1007/S11277-021-08410-6.
[19] B. Almusawy and A. A. H. Alrammahi, "Insider Detection Using Combination of Machine Learning and Expert Policies," *International Journal of Electrical and Electronic Engineering and Telecommunications*, vol. 13, no. 5, pp. 389–396, 2024, doi: 10.18178/IJEETC.13.5.389-396.
[20] A. Orfila, J. M. Estevez-Tapiador, and A. Ribagorda, "Evolving high-speed, easy-to-understand network intrusion detection rules with genetic programming," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2009, pp. 93–98. doi: 10.1007/978-3-642-01129-0_11.
[21] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, "DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter," Oct. 2019, Accessed: May 25, 2025. [Online]. Available: https://arxiv.org/pdf/1910.01108
[22] M. Akmal Afif Mohd Zuhdi, I. A. Rahmi Hamid, and F. Sains Komputer dan Teknologi Maklumat, "Classification of Spear Phishing Email using Machine Learning Approach," *Applied Information Technology And Computer Science*, vol. 5, no. 1, pp. 34–51, Aug. 2024, doi: 10.30880/aitcs.2024.05.01.003.
[23] A. Junnarkar, S. Adhikari, J. Fagania, P. Chimurkar, and D. Karia, "E-mail spam classification via machine learning and natural language processing," in *Proceedings of the 3rd International Conference on Intelligent Communication Technologies and Virtual Mobile Networks, ICICV 2021*, Institute of Electrical and Electronics Engineers Inc., Feb. 2021, pp. 693–699. doi: 10.1109/ICICV50876.2021.9388530.

[24] Samira. Douzi, F. A. AlShahwan, Mouad. Lemoudden, and Bouabid. El Ouahidi, “Hybrid Email Spam Detection Model Using Artificial Intelligence,” *Int J Mach Learn Comput*, vol. 10, no. 2, pp. 316–322, 2020, doi: 10.18178/ijmlc.2020.10.2.937.
[25] V. S. Tida and S. H. Hsu, “Universal Spam Detection using Transfer Learning of BERT Model,” in *Proceedings of the 55th Hawaii International Conference on System Sciences*, Hawaii International Conference on System Sciences, Jan. 2022. doi: 10.24251/hicss.2022.921.
[26] Y. Guo, Z. Mustafaoglu, and D. Koundal, “Spam Detection Using Bidirectional Transformers and Machine Learning Classifier Algorithms,” *Journal of Computational and Cognitive Engineering*, vol. 2, no. 1, pp. 5–9, 2023, doi: 10.47852/bonviewJCCE2202192.
[27] S. Zavrak and S. Yilmaz, “Email spam detection using hierarchical attention hybrid deep learning method,” *Expert Syst Appl*, vol. 233, 2023, doi: 10.1016/j.eswa.2023.120977.
[28] “The Enron-spam datasets.” Accessed: Sep. 29, 2024. [Online]. Available: https://www.cs.cmu.edu/~enron/
[29] “The Spam Assassin Email Classification Dataset.” Accessed: Sep. 29, 2024. [Online]. Available: https://www.kaggle.com/datasets/ganiyuolalekan/spam-assassin-email-classification-dataset
[30] S. Al-Augby and K. Nermend, “Using rule text mining based algorithm to support the stock market investment decision,” *Transformations in Business and Economics*, vol. 14, no. 3C, pp. 448–469, 2015.
[31] S. Kaddoura, G. Chandrasekaran, D. E. Popescu, and J. H. Duraisamy, “A systematic literature review on spam content detection and classification,” *PeerJ Comput Sci*, vol. 8, p. e830, Jan. 2022, doi: 10.7717/PEERJ-CS.830/TABLE-14.
[32] M. Parmar and A. Tiwari, “Enhancing Text Classification Performance using Stacking Ensemble Method with TF-IDF Feature Extraction,” in *Proceedings - 2024 5th International Conference on Mobile Computing and Sustainable Informatics, ICMCSI 2024*, 2024, pp. 166–174. doi: 10.1109/ICMCSI61536.2024.00031.
[33] D. Pakpahan, V. Siallagan, and S. Siregar, “Classification of E-Commerce Product Descriptions with The Tf-Idf and Svm Methods,” *sinkron*, vol. 8, no. 4, pp. 2130–2137, 2023, doi: 10.33395/sinkron.v8i4.12779.
[34] D. S. Ramdan, R. D. Apnena, and C. A. Sugianto, “Film Review Sentiment Analysis: Comparison of Logistic Regression and Support Vector Classification Performance Based on TF-IDF,” *Journal of Applied Intelligent System*, vol. 8, no. 3, pp. 341–352, 2023, doi: 10.33633/jais.v8i3.9090.
[35] M. Rajman and R. Besançon, “Text Mining: Natural Language techniques and Text Mining applications,” in *Data Mining and Reverse Engineering*, Springer US, 1998, pp. 50–64. doi: 10.1007/978-0-387-35300-5_3.
[36] M. Kuhn and K. Johnson, *Applied predictive modeling*. 2013. doi: 10.1007/978-1-4614-6849-3.
[37] M. Kuhn and S. Weston, “Package ‘Cubist’ R Package Version 0.4,” *mirror.las.iastate.edu*, 2024, Accessed: Oct. 20, 2024. [Online]. Available: https://mirror.las.iastate.edu/CRAN/web/packages/Cubist/Cubist.pdf
[38] R. M. Neal, “Pattern Recognition and Machine Learning,” *Technometrics*, vol. 49, no. 3, pp. 366–366, 2006, doi: 10.1198/tech.2007.s518.
[39] T. M. Therneau and E. J. Atkinson, “An introduction to recursive partitioning using the RPART routines,” 1997. Accessed: Oct. 06, 2024. [Online]. Available: http://stat.ethz.ch/R-manual/R-patched/library/rpart/doc/longintro.pdf
[40] T. Hothorn, K. Hornik, W. Wien, and A. Zeileis, “ctree: Conditional Inference Trees,” *The Comprehensive R Archive Network;*, no. Quinlan 1993, pp. 1–34, 2015, Accessed: Oct. 06, 2024. [Online]. Available: http://cran.irsn.fr/web/packages/partykit/vignettes/ctree.pdf
[41] T. Grubinger, A. Zeileis, and K. P. Pfeiffer, “Evtree: Evolutionary learning of globally optimal classification and regression trees in R,” *J Stat Softw*, vol. 61, no. 1, pp. 1–29, 2014, doi: 10.18637/jss.v061.i01.