

A Framework for Malicious Domain Names Detection using Feature Selection and Majority Voting Approach

Dharmaraj R. Patil

Department of Computer Engineering, R.C.Patel Institute of Technology, Shirpur, India

dharmaraj.patil@rcpit.ac.in

Keywords: Malicious domain names, Cybersecurity, Feature selection, Majority voting, Machine learning.

Received:

As cyber attacks become more sophisticated, identifying and mitigating bad domain names has become critical to assuring the security of online environments. This paper presents a framework for detecting malicious domain names using a feature selection strategy and a majority vote method. The suggested methodology begins with the extraction of important features from domain names and their related characteristics, followed by a rigorous feature selection procedure to determine the most discriminating attributes. To accomplish feature selection, a variety of feature selection techniques are used, including chi-square statistics, information gain, gain ratio, and correlation-based feature selection, to analyse the value of each characteristic in distinguishing benign and malicious domain names. In addition, a majority voting strategy is utilised to improve the detection system's overall accuracy and reliability by combining the predictions of different classifiers such as AdaBoost, logistic regression, k-nearest neighbours, naive bayes, and multilayer perceptron. The ensemble of classifiers is trained on the ideal features, yielding a complete and robust model capable of accurately recognising malicious domain names while minimising false positives. The proposed approach is evaluated against real-world examples of harmful domain names. The suggested framework employing Chi-square feature selection and majority voting detects malicious domain names with an accuracy of 99.44%, precision of 99.44%, recall of 99.44%, and f-measure of 99.44%. The use of feature selection and a majority voting technique improves the system's adaptability and resilience in the face emerging cyber threats.

Povzetek:

1 Introduction

In an era characterized by the pervasive influence of the digital landscape, the escalating sophistication of cyber threats poses a significant challenge to the security of online environments. Malicious actors exploit various avenues to compromise the integrity and confidentiality of information, and one such vector is the utilization of malicious domain names. These deceptive entities serve as a pivotal component in orchestrating cyber-attacks, making their timely detection an imperative aspect of cybersecurity. As per the Cybercrime Information Center, there has been a substantial rise in the utilization of domain names within

malware URLs. A study conducted by Interisle revealed a significant surge, indicating a 121% increase in the prevalence of domain names in the fourth quarter of 2022 [1].

Additionally, as outlined in the CSC Domain Security 2023 report [2]:

- 43% of .AI domains are registered by entities not associated with the original owners.
- 21% of DNS records from subdomains direct to unresolvable content, exposing organizations to the risk of subdomain hijacking.
- 79% of registered domains bearing resemblance to global 2000 brands (homoglyphs)

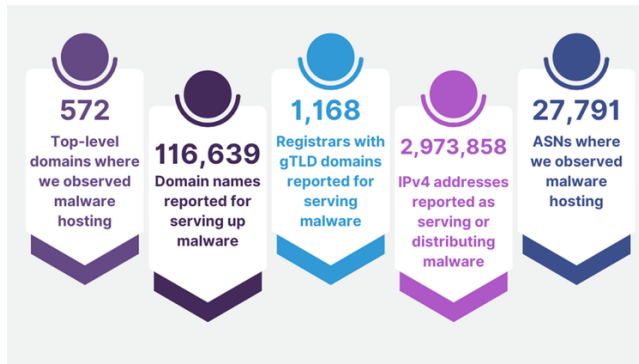


Figure 1: Interisle malicious domain names statistics 4Q 2022. *Source:* <https://www.cybercrimeinfocenter.org/malware-andscape-2023>

are under the ownership of third parties.

- 46% of enterprises employing enterprise-class registrars have also implemented registry locks.
- A noteworthy 112 companies exhibit a domain security score of 0%.
- Domain-based message authentication, reporting, and conformance have grown by 6%, the greatest increase in the past four years.

This study focuses on the design of a detection system for fraudulent domain names that takes advantage of the synergy between optimal feature selection and a majority voting technique. The rising complexity of cyber threats needs new and adaptable methods for identifying and mitigating possible dangers. By tackling the issues of detecting bad domain names, this study helps to improve the overall resilience of online systems to online attacks. To build a strong framework, the suggested methodology combines machine learning concepts, feature selection, and ensemble classifiers. It is the goal of a careful feature selection procedure to uncover the most discriminatory qualities that discriminate between benign and malicious domain names. This ideal feature subset not only improves the detection system's performance, but it also helps it adapt to changing threat landscapes. Furthermore, using a majority voting mechanism strengthens the detection system's dependability. By aggregating several classifier predictions, the model improves its ability to make accurate decisions, reduce false pos-

itives, and improve overall system performance. The ensemble of classifiers, trained on the optimal characteristics chosen, forms a cohesive defence system against the ever-changing nature of cyber attacks. The suggested framework, which uses Chi-square feature selection and majority voting, detects malicious domain names with an accuracy of 99.44%, precision of 99.44%, recall of 99.44%, and f-measure of 99.44%. The main outcomes of this study are as follows:

- This study introduces a strong methodology for detecting fraudulent domain names, providing a solution to the expanding issues provided by cyber attacks in online environments.
- The proposed system includes an optimal feature selection strategy based on modern machine learning algorithms and statistical techniques. Using Chi-square feature selection and majority voting, the framework achieves exceptional performance, with accuracy, precision, recall, and f-measure all registering at 99.44%.
- The proposed framework's efficacy is carefully evaluated using a variety of datasets, including real-world examples of malicious domain names. This extensive test assures the detection system's practical usefulness and resilience in a variety of tough settings.

The remaining sections of this work are organised as follows. Section 2 presents the motivation for the malicious domain names detection. Section 3 offered a brief summary of pertinent past studies. Section 4 described the methodology that included feature selection techniques and supervised batch machine learning algorithms. Section 5 has a full account of the findings. Section 6 presents the conclusive findings.

2 Motivation

The identification and mitigation of malicious domain names are crucial cybersecurity efforts. With an increased reliance on the internet for different parts of everyday life, ranging from communication and commerce to critical services and infrastructure, the threat presented by hostile actors using domain names for evil purposes has

grown dramatically. Malicious domain names serve as a conduit for a variety of cyber dangers, such as phishing attempts, malware distribution, and botnet command and control infrastructure. As a result, comprehensive and effective procedures for detecting and combating these threats are urgently required to protect persons, organisations, and the integrity of the digital ecosystem.

Traditional techniques to malicious domain name identification frequently depend on human analysis or static rule-based systems, both of which are fundamentally constrained in their capacity to react to the dynamic and complex nature of current cyber threats. Furthermore, the sheer amount of domain names produced everyday renders human examination impracticable and resource-intensive. As a result, there is a rising need to create automated, data-driven approaches for detecting malicious domain names while minimising false positives and reacting to changing threat environments.

The proposed approach is motivated by the need to solve these difficulties by using the capabilities of machine learning, feature selection approaches, and ensemble methods to improve the accuracy and scalability of malicious domain name identification. Our approach seeks to distinguish benign and malicious domain names with high precision and efficiency by utilising powerful algorithms and relevant data retrieved from domain name properties, DNS traffic, and contextual information. Furthermore, the use of a majority voting strategy allows for the combining of predictions from several classifiers, enhancing robustness and resilience against adversarial evasion tactics. This ensemble technique not only improves detection accuracy but also provides a mechanism for responding to emergent threats in real time.

3 Related Work

The field of malicious domain name identification has received a lot of interest in recent years, with academics and practitioners working to build efficient solutions to tackle emerging cyber threats. This section provides a quick summary of important past research, emphasising major contributions and approaches used in the topic.

Hong Zhao et al. devised an N-Gram-

based technique for identifying fraudulent domain names. The method makes use of Alexa's top 100,000 domain names from 2013, with N-Gram segmentation applied to each domain name except the top-level domain. Substrings of varying lengths (3 to 7) are then formed based on the domain levels. A substring set is defined, and the weight of each substring is calculated by its frequency in the set. To detect malicious assaults, the N-Gram technique is used to segment a given domain name, and its reputation value is calculated using the weights of its substrings. Thresholding determines the domain's harmful character. Experiments using Alexa 2017 and the Malware domain list showed that the algorithm was successful, with an accuracy rate of 94.04%, a false negative rate of 7.42%, and a false positive rate of 6.14%. Notably, the suggested technique demonstrates reduced temporal complexity than existing current fraudulent domain name identification algorithms [3].

Ali Soleymani et al. investigated the DNS network protocol using machine learning methods and text mining techniques, with a special focus on botnet detection. The investigation involves extracting and labelling domain name datasets that contained both healthy and contaminated Domain Generation Algorithm botnet data. To support this research, text-mining-based data preparation methods such as n-gram analysis and Principal Component research were used. The use of PCA includes the extraction of statistical characteristics to improve model performance. The suggested model's performance was evaluated using a variety of machine learning classifiers, including decision tree, support vector machine, random forest, and logistic regression. Experimental results reveal that the random forest algorithm demonstrates significant efficacy in botnet identification, demonstrating the greatest accuracy across classifiers [4].

Luhui Yang et al. presented a domain name syntax model to improve the identification accuracy of algorithmically created domain names. This approach examines many aspects inside domain names and their syntactic links. An adaptive embedding approach is presented to make domain name element processing more efficient. The authors also provide a parallel convolutional model with a feature selection module and an up-

graded dynamic loss function based on curriculum learning. This approach performs well in recognising multi-element fraudulent domain names. In a series of tests, the suggested model is tested against five current methods. The findings show that the suggested model outperforms the comparison methods in recognising multiple-element malicious domain names [5].

Shaojie Chen et al. implemented a meaningful word segmentation technique to define the structure of dictionary-based Algorithmically Generated Domains (AGDs). In this study, they propose using standard deviation to improve the assessment of word distribution properties. In addition, an 11-dimensional statistical feature set is created to enhance the findings of word segmentation. The authors then improve detection performance for both character- and dictionary-based AGDs by using 3-gram and 1-gram sequence characteristics. The final phase is feature fusion, which combines the four types of features stated above, resulting in an end-to-end detection approach for both character-based and dictionary-based AGD. The proposed technique achieved an overall accuracy of 97.24% based on experimental assessments. Specifically, it beat previous approaches in terms of accuracy and F1 values on both dictionary-based and character-based AGD datasets [6].

Atif Ali Wagan et al. have developed a new unified learning technique that uses both numerical and linguistic aspects of domain names to determine if a particular domain name combination is harmful. The trials were conducted on a benchmark dataset of 90,000 domain names. The experimental findings show that the suggested strategy outperforms six comparative approaches in terms of accuracy, precision, recall, and F1-Score. This study makes a significant addition to domain name categorization by presenting a unified learning framework that outperforms previous algorithms across several performance measures [7].

Leyla Bilge et al. introduced EXPOSURE, a system aimed to detect rogue domains using large-scale, passive DNS analysis techniques. The method is based on 15 unique characteristics retrieved from DNS data, which allow for the characterisation of various DNS name attributes and querying patterns. The approach's scalability was evaluated using a large real-world dataset includ-

ing 100 billion DNS queries. A two-week practical implementation within an ISP also confirmed the system's capacity to automatically detect previously undiscovered harmful domains used in different malicious activities, such as botnet command and control, spamming, and phishing. This work demonstrates the scalability and efficiency of the EXPOSURE system in detecting and mitigating harmful domains across varied malicious actions [8].

Zhaoshan Fan et al. have presented PUMD, a unique framework for detecting malicious domains. To solve the issue of insufficient label information, this system uses a novel Positive and Unlabeled (PU) learning method. To address class imbalance, a customised sample weight technique is used, and evidence features are efficiently built using resource overlapping to reduce the intra-class distance of malicious samples. Furthermore, a feature selection technique based on permutation significance and binning is given to discover the most informative detection features. Experiments on the open-source actual DNS traffic dataset given by QI-ANXIN Technology Group evaluate the PUMD framework's usefulness in collecting probable C&C domains for harmful activity. These trials show that PUMD consistently provides higher detection performance across various label frequencies and class imbalance ratios [9].

Luhui Yang et al. investigated inter-word and inter-domain correlations using semantic analysis methods, with a special emphasis on word embedding and part-of-speech considerations. The researchers provide a detection framework customised for word-based Domain Generation Algorithms, which incorporates word frequency distribution and part-of-speech into feature set creation. To assess the suggested technique, the ensemble classifier, which includes Naive Bayes, Extra-Trees, and Logistic Regression, is used with both malicious and valid domain samples retrieved from publicly available datasets. When compared to three state-of-the-art DGA detection techniques, the experimental results show that the suggested scheme is much more accurate in identifying word-based DGAs [10].

Yong Shi et al. presented a machine learning framework for detecting malware domain names that makes use of the Extreme Learning Machine

(ELM). Their technique is built on ELM, which is well-known for its high accuracy and quick learning capabilities. The researchers used ELM to categorise domain names based on information derived from several sources. Experiments reveal that the suggested detection approach has a high detection rate and accuracy, exceeding 95%. Furthermore, comparison trials demonstrate the quick learning pace of their ELM-based technique. As a result, the researchers say that their approach using ELM is not only successful but also efficient in identifying fraudulent sites [11].

Yu Fu et al. developed two Domain Generation Algorithms that use Hidden Markov Models and Probabilistic Context-Free Grammars, respectively. Their experimental results indicate that traditional DGA detection metrics such as KL, JI, and ED, as well as detection tools such as BotDigger and Pleiades, have difficulties in recognising domain names formed using these methods. To overcome this, the researchers use game theory to optimise methods for botmasters and security professionals. The findings show that, for optimal DGA detection, security personnel should prioritise ED detection with a probability of 0.78 and JI detection with a probability of 0.22. Botmasters should choose the HMM-based DGA with a probability of 0.67 and the PCFG-based DGA with a probability of 0.33, respectively. This study improves DGA detection tactics by using game theory ideas to optimise detection methodologies for both security staff and botmasters [12].

Xiaochun Yun et al. proposed Khaos, a unique Domain Generation Algorithm that uses neural language models and the Wasserstein Generative Adversarial Network to provide strong anti-detection capabilities. The researchers' major finding concerns the construction of genuine domain names, which frequently consist of legible syllables and acronyms. Using this knowledge, Khaos organises syllables and acronyms in neural language models to simulate genuine domain names. Using the Khaos framework, the researchers first determine the most prevalent n-grams in actual domain names. They then tokenize these domain names into n-grams and create new domain names by learning n-gram arrangements from existing domain names. Experimental assessments were carried out employing a variety of cutting-edge DGA detection method-

ologies, including statistics-based, distribution-based, LSTM-based, and graph-based methods. The experimental results show that Khaos poses significant challenges for existing detection approaches, with an average distance of 0.64 for detecting Khaos using the distribution-based detection approach, AUCs of 0.76 and 0.57 for the statistics-based and LSTM-based detection approaches, and a precision of 0.68 for Khaos using the graph-based detection approach. This emphasises Khaos' greater anti-detection performance compared to state-of-the-art DGAs [13].

Luhui Yang et al. examined the character-level properties of Sub-domain Domain Generation Algorithm domain names and suggested a new detection framework called Heterogeneous Deep Neural Network. The HDNN framework employs a novel Improved Parallel Convolutional Neural Network architecture that incorporates multi-sized convolution kernels to extract multi-scale local information from domain names. The system also includes a revolutionary Self-Attention-based Bidirectional Long Short-Term Memory architecture, which uses an attention mechanism to extract bidirectional global information from domain names. Furthermore, the researchers adopt a focused loss function to overcome the sample quantity imbalance during the training period. Benchmark studies were carried out on a database of benign domain names, real-world DGAs, and SDGAs. Six popular deep-learning-based DGA detection techniques were compared. The findings show that the suggested technique outperforms state-of-the-art detection for SDGAs while also excelling in binary and multiclass classification for standard DGAs. This research adds to the improvement of DGA detection by presenting a unique framework suited exclusively for SDGAs, demonstrating higher performance in comparison to previous approaches [14].

Congyuan Xu et al. devised a unique strategy by integrating n-gram analysis with deep convolutional neural networks, resulting in the establishment of a novel n-gram combined character-based domain classification model. This model is end-to-end, therefore no manually extracted characteristics or domain name system (DNS) contextual information are required. It just requires the domain name as input, allowing for the automated assessment of the chance that the domain

name was generated by Domain Generation Algorithms (DGAs). Experiments on real-world data show that the proposed technique effectively detects domain names created by DGAs, with an average detection rate of 98.69% and an average F-measure of 0.9829. The suggested method outperforms state-of-the-art algorithms in recognising pronounceable and wordlist-based DGA domain names, with a detection rate of over 93.89%. Thus, the suggested detection approach is resilient and versatile in recognising various forms of domain names created by DGAs. This study marks a substantial leap in DGA detection strategies [15].

R. Vinayakumar et al. obtained DNS logs only from client PCs on a local area network (LAN) and stored them on a central server. The researchers recommended using deep learning to determine if a domain name is benign or harmful. They performed a comparison analysis to assess the performance of several deep learning approaches, such as recurrent neural networks (RNN), long short-term memory (LSTM), and classic machine learning classifiers. The study found that deep learning-based techniques outperformed standard machine learning classifiers. This advantage is due to deep learning algorithms' inherent ability to implicitly collect significant characteristics. Notably, LSTM outperformed other deep learning algorithms in terms of malicious detection rate across all studies. This study highlights the effectiveness of deep learning approaches in DNS log analysis for detecting malicious domains within LAN settings [16].

Luhui Yang et al. constructed a semantic element representation model for domain names using a probabilistic context-free grammar model and a collection of semantic components linked with domain names. The model starts by analysing and categorising the domain names' basic parts. It then proposes a syntax tree analysis approach for establishing semantic links between these components, allowing for the efficient encoding of many items inside domain names. This methodology classifies harmful domain names into four types: random character-based, word-based, predicted character-based, and multi-element hybrid. The researchers discovered considerable differences between harmful and normal domain names, as well as between

other types of malicious domain names, using tests aimed to analyse anomalies and concealing patterns in domain names. The comparative experimental findings demonstrate the efficiency of the suggested methodology in improving the accuracy of malicious domain name detection. This study adds to the improvement of domain name analysis approaches by presenting a semantic element representation model that efficiently distinguishes between malicious and lawful domain names based on their structural properties [17].

4 Methodology

Figure 2 depicts the suggested architecture for detecting malicious domain names. The framework consists of many steps, including data collection and preprocessing, feature selection, model creation, and the use of a majority voting technique. Each phase substantially improves the overall effectiveness of the malicious domain name detection system.

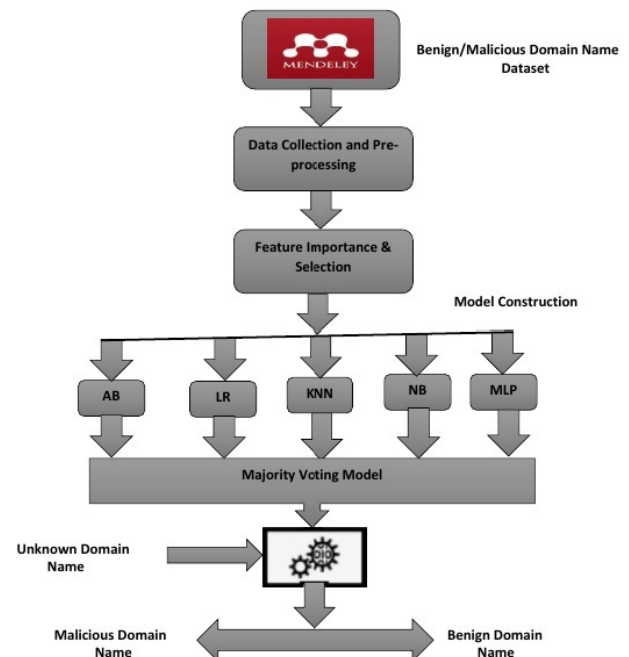


Figure 2: Proposed Framework for Malicious Domain Name Detection System

4.1 Data Collection and Preprocessing

This study uses the Mendeley Dataset, which includes both benign and malicious domains re-

trieved from DNS logs [18]. This dataset is especially designed for supervised machine learning research to differentiate between harmful and non-malicious domain names. It was rigorously curated by combining publicly accessible DNS logs from both sorts of domain names. Each domain name is used as an input in the dataset, resulting in 34 characteristics. Domain name properties such as entropy, the occurrence of unique characters, and domain name length are examples of features that are directly extracted. Furthermore, supplemental details such as domain creation date, related IP address, open ports, and geolocation were obtained by data enrichment methods that used Open Source Intelligence methodologies. This collection of 90,000 domain names is rigorously balanced, providing an equal mix of 50% non-malicious and 50% malicious domains.

Five features are removed from the training and testing datasets during the preparation stage: DNSRecordType, CountryCode, RegisteredCountry, RegisteredOrg, and TLD. As a result, the final experimental dataset has 29 characteristics that will be used for model training and testing.

4.2 Feature Selection Techniques used for Malicious Domain Names Detection

Feature selection strategies are used to determine the most discriminating characteristics. To determine the relevance of features to the classification job, many approaches were used, including Chi-square, Gain Ratio, Information Gain, and Correlation-based feature selection. These strategies were used to rank the significance of traits based on their discriminating potential. As a result, we chose a collection of attributes that offer the most important contributions to detecting fraudulent domain names while reducing repetition. We use the Weka implementation of the aforementioned feature selection approaches [19].

4.2.1 Chi-square (χ^2) Feature Selection Technique

Chi-square (χ^2) feature selection is an effective strategy for selecting the most informative features in classification problems. It computes the

chi-square statistic to assess the relationship between each characteristic and the target variable. This statistic measures the amount of correlation between categorical variables, allowing us to determine which traits are most relevant for predicting the target variable [20]. The chi-square statistic is computed using the formula below.

$$\chi^2 = \sum \frac{(O - E)^2}{E} \quad (1)$$

Where:

- χ^2 is the chi-square test statistic.
- O is the observed frequency (actual count) of each category or combination of categories in the dataset.
- E is the anticipated frequency (theoretical count) of each category or combination of categories, assuming feature and target variable independence.

Once computed, the chi-square statistic is used to determine the degree of independence between the feature and the target variable. A greater chi-square value implies a stronger link between the characteristic and the goal, whereas lower values imply a weaker connection. In the context of feature selection, characteristics with higher chi-square values are judged more meaningful for predicting the target variable. Table 1 shows the results of the Chi-square (χ^2) Feature Selection, which highlights the most important traits. The Chi-square Feature Selection approach has a threshold of 26067.5297. According to Table 1, a thorough selection procedure resulted in the identification of the 12 most relevant attributes from a total of 28 in the Mendeley DNS dataset.

4.2.2 Gain Ratio Feature Selection Technique

Gain Ratio feature selection stands out as a useful strategy for prioritising characteristics that contribute the most to classification problems. This approach evaluates each feature's importance by computing the gain ratio, which considers its inherent qualities and capacity to minimise uncertainty in predicting the target variable [21]. It is computed using the formula below.

Feature No.	Chi-square Statistic	Feature Name
f1	90000	Domain
f2	73423.4809	NumericSequence
f3	72239.5233	NumericRatio
f4	70611.0287	IP
f5	59096.0642	StrangeCharacters
f6	59074.1194	ASN
f7	55600.5647	ConsoantRatio
f8	51850.5355	DomainLength
f9	45977.5991	VowelRatio
f10	43233.5773	SubdomainNumber
f11	27028.8843	HasSPFInfo
f12	26067.5297	TXTDnsResponse

Table 1: Feature selection using Chi-square Statistic on Mendeley DNS Dataset

$$IGR(X, Y) = \frac{IG(X, Y)}{H(X)} \quad (2)$$

Where:

- $IG(X, Y)$ is the Information Gain between feature X and target variable Y .
- $H(X)$ is the entropy of feature X .

The entropy $H(X)$ for a discrete feature X is calculated as:

$$H(X) = - \sum_i P(x_i) \cdot \log_2(P(x_i)) \quad (3)$$

Where:

- $P(x_i)$ represents the likelihood of a feature X having a value x_i .

In the area of Information Gain Ratio, the intrinsic information of characteristics is taken into account, yielding a normalised measure. This normalisation is useful when dealing with characteristics that have varied sizes or numbers of possible values. The Information Gain Ratio balances any bias towards features with a large number of different values, which may have a high Information Gain owing to variability. Table 2 illustrates how the information gain ratio is used to identify the most relevant attributes. The information gain ratio feature selection approach, which has a threshold of 0.105, helps with this procedure. A rigorous selection procedure resulted in the identification of the 12 most significant characteristics from the Mendeley DNS Dataset displayed in Table 2.

4.2.3 Information Gain Feature Selection Technique

Among the several feature selection approaches, Information Gain feature selection is a crucial way for determining the most significant qualities in classification problems. It uses information theory ideas to measure how much information is acquired about the target variable by incorporating a certain feature into the model. This strategy prioritises characteristics that reduce uncertainty about the target variable, increasing the model's predictive power [22]. The Information Gain (IG) for a given characteristic X with regard to a target variable Y is commonly computed using the following formula:

$$IG(X, Y) = H(Y) - H(Y|X) \quad (4)$$

Where:

- $H(Y)$ represents the entropy of the target variable Y without regard for any special attribute.
- $H(Y|X)$ represents the conditional entropy of Y in the presence of the feature X .

The entropy $H(Y)$ is calculated as:

$$H(Y) = - \sum_i P(y_i) \cdot \log_2(P(y_i)) \quad (5)$$

Where:

Feature No.	Gain Ratio	Feature Name
f1	0.437532	NumericSequence
f2	0.393146	NumericRatio
f3	0.364636	SubdomainNumber
f4	0.257425	ConsoantRatio
f5	0.22911	HasSPFInfo
f6	0.220428	TXTDnsResponse
f7	0.218858	StrangeCharacters
f8	0.213575	VowelRatio
f9	0.162423	CreationDate
f10	0.148529	ASN
f11	0.129045	IP
f12	0.105742	DomainLength

Table 2: Feature selection using Gain Ratio method on Mendeley DNS Dataset

- $P(y_i)$ represents the probability of class y_i in the target variable Y .

The conditional entropy $H(Y|X)$ is calculated as:

$$H(Y|X) = \sum_j P(x_j) \cdot H(Y|X = x_j) \quad (6)$$

Where:

- $P(x_j)$ is the probability of the occurrence of feature X having value x_j .
- $H(Y|X = x_j)$ is the target variable's entropy when the feature X has the value x_j .

In practice, Information Gain is used as a metric to analyse the efficacy of a certain characteristic in distinguishing between distinct classes within the target variable. Higher information gain values indicate more discriminating power. As a result, the feature with the largest Information Gain is judged the most useful and is usually prioritised for further analysis or model training. Table 3 illustrates how information gathering was used to pick the most relevant attributes. The selection of information gain features is guided by a predefined criterion of 0.220. After examining the Mendeley DNS Dataset presented in Table 3, a thorough selection procedure resulted in the identification of the 12 most relevant attributes.

4.2.4 Correlation-based Feature Selection (CFS) Technique

Correlation-based Feature Selection is an important approach in machine learning that helps

identify the most relevant variables from a dataset. This approach uses correlation to determine the degree of link between characteristics and the target variable, making it easier to pick the features that contribute the most to predictive modelling tasks. CFS is especially useful in situations when datasets include both numerical and categorical information and are used for classification or regression tasks. CFS prioritises features with significant predictive potential and minimises duplication among selected characteristics by evaluating their correlation with the target variable [23]. The CFS algorithm generally consists of two basic steps:

- Determine the correlation between each attribute and the target variable.
- Evaluating the redundancy among selected features.

Define X as the feature set, Y as the target variable, and S as the chosen subset of features. Find the correlation between each characteristic X_i and the target variable Y . The Pearson correlation coefficient, which is calculated as:

$$\rho_{X_i, Y} = \frac{\sum_{j=1}^n (X_{ij} - \bar{X}_i)(Y_j - \bar{Y})}{\sqrt{\sum_{j=1}^n (X_{ij} - \bar{X}_i)^2 \sum_{j=1}^n (Y_j - \bar{Y})^2}} \quad (7)$$

where:

- X_{ij} represents the value of feature X_i in the j th observation.

Feature No.	Information Gain	Feature Name
f1	1.000000000000000256	Domain
f2	0.777969675090184576	NumericSequence
f3	0.75109695664116224	NumericRatio
f4	0.749260387531234944	IP
f5	0.606173981999299456	ASN
f6	0.5782910953568256	StrangeCharacters
f7	0.55113539106572576	ConsoantRatio
f8	0.499455615169670976	DomainLength
f9	0.44302627161621632	SubdomainNumber
f10	0.413287777249203968	VowelRatio
f11	0.229060631434872096	HasSPFInfo
f12	0.22041276495032848	TXTDnsResponse

Table 3: Feature selection using Information Gain method on Mendeley DNS Dataset

- \bar{X}_i represents the mean of the feature X_i .
- Y_j represents the target variable's value in the j th observation.
- \bar{Y} represents the mean of the target variable.
- n denotes the number of observations.

Rank the features by their correlation coefficients with the target variable.

- Add features to the selected subset S in a stepwise manner.
- At each step, add the feature with the highest correlation coefficient that is not highly correlated with features already in S (to minimize redundancy). One common criterion to measure redundancy is to compute the average pairwise correlation between features in S .
- Repeat step 3 until a stopping requirement is satisfied.

Feature selection prioritises features with better absolute correlation coefficients for predicting the target variable. Positive correlation values imply that when the feature value increases, so does the target variable, whilst negative correlation coefficients indicate the reverse. After calculating correlation coefficients for all characteristics, the top k features with the highest absolute correlation coefficients are selected. The chosen attributes are then used as input for machine

learning models. After painstakingly analysing the Mendeley DNS Dataset, as shown in Table 4, selected the four most relevant attributes.

Feature No.	Feature Name
f1	LastUpdateDate
f2	SubdomainNumber
f3	NumericRatio
f4	NumericSequence

Table 4: Feature selection using Correlation-based Feature Selection (CFS) method on Mendeley DNS Dataset

4.3 Machine Learning Techniques used for Malicious Domain Names Detection

Because of their capacity to analyse enormous datasets and uncover patterns indicating harmful behaviour, machine learning techniques have emerged as viable tools for detecting rogue domain names. These methods use various algorithms and models to categorise domain names as benign or dangerous based on attributes gathered from their properties. Supervised learning is a typical machine learning strategy for detecting harmful domain names, in which models are trained on labelled datasets that include both benign and malicious domain names. These models learn to distinguish between the two groups by extracting characteristics including domain age, length, lexical qualities, and historical DNS information. Several supervised learning approaches

were used in this work, including AdaBoost, Logistic Regression, K-nearest Neighbours, Naive Bayes, Multilayer Perceptron, and Majority Voting. Additional information on every approach is provided below [24].

4.3.1 AdaBoost

AdaBoost, or Adaptive Boosting, is a pioneering ensemble learning algorithm commonly used in machine learning for categorization applications. Its innovation stems from its capacity to train a sequence of weak learners consecutively, iteratively concentrating on cases that were misclassified in prior rounds. AdaBoost emphasises difficult-to-classify occurrences by providing larger weights to misclassified examples, resulting in a robust and accurate classification model [25].

AdaBoost is a sort of ensemble learning that combines numerous weak learners to create a robust learner. A mathematical representation of the AdaBoost algorithm is shown here. AdaBoost uses a number of rounds to train weak classifiers and apply weights based on their performance. The final prediction is calculated as the weighted sum of the weak classifiers' predictions. Consider a training dataset written as (X, y) , where X signifies the feature vectors and y represents the labels (usually encoded as -1 and 1 for binary classification). The mathematical description of the AdaBoost method is as follows:

Notations:

- D : Dataset with n instances.
- $X = \{x_1, x_2, \dots, x_n\}$: Feature vectors.
- $Y = \{y_1, y_2, \dots, y_n\}$: Corresponding labels.
- T : Number of boosting rounds (iterations).
- $h_t(x)$: Weak classifier at iteration t .
- α_t : Weight for the weak classifier $h_t(x)$ at iteration t .
- $H(x)$: Final strong classifier.
- w_i^t : Weight of instance x_i at iteration t .
- ϵ_t : Weighted error of weak classifier $h_t(x)$.

Algorithm:

1. Initialize weights: Set initial weights $w_i^1 = \frac{1}{n}, i = 1, 2, \dots, n$.
2. For $t = 1$ to T :
 - Train weak classifier: Train weak classifier $h_t(x)$ using weights w_i^t .
 - Calculate error: Calculate weighted error $\epsilon_t = \sum_{i=1}^n w_i^t \cdot 1\{h_t(x_i) \neq y_i\}$, where 1 is the indicator function.
 - Calculate weight for classifier: Compute $\alpha_t = \frac{1}{2} \ln \left(\frac{1-\epsilon_t}{\epsilon_t} \right)$.
 - Update instance weights: Update weights for next iteration: $w_i^{t+1} = \frac{w_i^t \cdot \exp(-\alpha_t \cdot y_i \cdot h_t(x_i))}{Z_t}$, where Z_t is a normalization factor to ensure weights sum up to 1.
 - Construct final model: $H(x) = \text{sign} \left(\sum_{t=1}^T \alpha_t \cdot h_t(x) \right)$

AdaBoost offers additional weight to misclassified cases with each iteration, pushing weak classifiers to focus more on them. The weight α_t of each weak classifier is given by its weighted error rate. The final model $H(x)$ combines the weak classifier predictions using weighted majority voting. AdaBoost is a strategy that combines predictions from several weak classifiers, giving preference to those with superior performance. It uses weighted voting to aggregate the individual classifier results, resulting in a final prediction. The main idea behind AdaBoost is to prioritise difficult cases and change data weights during training to highlight misclassified samples. This iterative method is intended to improve the overall performance of the ensemble model.

4.3.2 Logistic Regression

Logistic Regression stands as a fundamental and widely used statistical technique for binary classification tasks. Despite its name, logistic regression is a classification algorithm rather than a regression one. It models the probability of a binary outcome by fitting the data to a logistic function, allowing for efficient estimation of the likelihood of a sample belonging to a particular class. LR is a statistical model used for binary classification. Here's the mathematical representation of logistic regression [26].

Notations:

- m : Number of training examples.
- n : Number of features.
- X : Matrix of input features with dimensions $m \times (n + 1)$ (including the intercept term).
- y : Vector of labels with dimensions $m \times 1$.
- θ : Vector of parameters (weights) with dimensions $(n + 1) \times 1$.
- $h_{\theta}(x)$: Hypothesis function for logistic regression.
- $g(z)$: Sigmoid function $g(z) = \frac{1}{1+e^{-z}}$.

Algorithm:

1. Initialize Parameters: Initialize the parameter vector θ to zeros or small random values.
2. Hypothesis Function: Define the hypothesis function, $h_{\theta}(x) = g(\theta^T x)$, where $g(z)$ is the sigmoid function.
3. Cost Function: Define the logistic cost function, $J(\theta) = -\frac{1}{m} \sum_{i=1}^m [y^{(i)} \log(h_{\theta}(x^{(i)})) + (1 - y^{(i)}) \log(1 - h_{\theta}(x^{(i)}))]$.
4. Gradient Descent:
 - Repeat until convergence: $\theta_j := \theta_j - \alpha \frac{\partial J(\theta)}{\partial \theta_j}$, for $j = 0, 1, \dots, n$, where α is the learning rate.
 - Update rule for each parameter θ_j : $\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)}) x_j^{(i)}$
5. Feature Scaling: Scale the features to improve convergence speed if needed.
6. Prediction: Given a new input x , predict the probability that $y = 1$ using $h_{\theta}(x)$: $h_{\theta}(x) = g(\theta^T x)$.

Logistic regression uses gradient descent to minimise the logistic cost function and optimise the parameters θ . The sigmoid function $g(z)$ reduces input to the range $(0, 1)$, making it appropriate for describing probabilities. During training, the algorithm iteratively modifies the parameters θ to obtain the appropriate decision boundary. After training, the model may use the hypothesis function $h_{\theta}(x)$ to estimate the likelihood of an input belonging to the positive class ($y = 1$).

4.3.3 K-nearest Neighbours

K-nearest Neighbours is a powerful and easy-to-use technique for machine learning classification and regression. It falls under the domain of instance-based learning, in which predictions are produced based on the similarity between the input instance and its neighbours in the feature space. KNN works on the assumption that instances with comparable feature values are likely to belong to the same class or have similar target values. KNN is a basic and effective machine learning technique that may be used for both classification and regression. Here's the mathematical depiction of the KNN algorithm: KNN is a nonparametric, instance-based technique that predicts the majority class or the average of the k -nearest data points in the feature space [27].

Notations:

- D : Training dataset consisting of n instances.
- $X = \{x_1, x_2, \dots, x_n\}$: Feature vectors in the dataset.
- K : Number of nearest neighbors to consider.
- x : Input feature vector for which prediction is to be made.

Algorithm:

1. Calculate Distance: For each instance x_i in the training set, Compute the distance between x and x_i using a distance metric (e.g., Euclidean distance, Manhattan distance).
2. Find K Nearest Neighbors: Select the K instances with the smallest distances to x .
3. Majority Vote: For classification, assign the class label that is most frequent among the K nearest neighbors.

KNN is considered a lazy learning method since it does not explicitly construct a model during the training phase. Instead, it saves the whole training dataset. During prediction, it locates the K closest neighbours to the input instance and makes predictions based on them. The distance metric (e.g., Euclidean, Manhattan, Minkowski) and the value of K are critical hyperparameters that determine algorithm performance. Since KNN is sensitive to feature scale, feature scaling is frequently used to normalise the features.

4.3.4 Naive Bayes

Naive Bayes is a basic yet strong probabilistic classification method commonly employed in machine learning due to its efficiency and efficacy in dealing with huge datasets. It is founded on Bayes' theorem, which describes the likelihood of a hypothesis given the data. Despite its "naive" assumption of independence between features, Naive Bayes frequently works extremely well in practice, notably in text classification and spam filtering tasks [28].

Notations:

- D : Dataset with n instances.
- $X = \{x_1, x_2, \dots, x_n\}$: Feature vectors in the dataset.
- $Y = \{y_1, y_2, \dots, y_n\}$: Corresponding class labels in the dataset.
- x : Input feature vector for which prediction is to be made.
- X_i : Value of the i -th feature.
- $P(X_i|Y)$: Probability of feature X_i given class Y .
- $P(Y)$: Prior probability of class Y .
- $P(Y|X)$: Posterior probability of class Y given features X .

Algorithm:

1. Compute Class Priors: Calculate the prior probabilities $P(Y = y)$ for each class y in the dataset. $P(Y = y) = \frac{\text{number of instances with class } y}{\text{total number of instances}}$.
2. Compute Feature Likelihoods: For each feature X_i and each class y , Calculate the likelihood $P(X_i|Y = y)$ of feature X_i given class y . This can be done using different probability density functions (e.g., Gaussian distribution, multinomial distribution for discrete features).
3. Compute Posterior Probabilities: For a new input feature vector x . For each class y , calculate the posterior probability $P(Y = y|X = x)$ using Bayes' theorem: $P(Y = y|X = x) = \frac{P(Y=y) \cdot \prod_{i=1}^n P(X_i=x_i|Y=y)}{\sum_{y'} P(Y=y') \cdot \prod_{i=1}^n P(X_i=x_i|Y=y')}$.

4. Predict the Class: Assign the class label \hat{y} with the highest posterior probability: $\hat{y} = \arg \max_y P(Y = y|X = x)$.

Based on the class, Naive Bayes assumes that the features are conditionally independent. This is a strong and frequently impractical assumption, yet it simplifies the computation and typically works well in practice. The approach calculates the class priors (prior probability for each class), the feature likelihoods (conditional probabilities for each feature given each class), and then uses Bayes' theorem to obtain the posterior probabilities. The input feature vector's output class is anticipated to be the one with the highest posterior probability. Naive Bayes is efficient and performs well with high-dimensional data, but it is sensitive to the independence assumption's quality and may overfit with small datasets.

4.3.5 Multilayer Perceptron

The Multilayer Perceptron is a basic artificial neural network design that has been widely applied to machine learning applications such as classification, regression, and pattern recognition. MLPs are made up of several layers of linked neurons, each with one or more neurons that execute calculations on the input data. MLPs are well-known for their capacity to learn complicated patterns from data using the supervised learning method [29]. A Multilayer Perceptron (MLP) is a form of artificial neural network made up of numerous layers of nodes (neurons), with each layer completely linked to the next. Here is the method in several notations:

Notations:

- D : Dataset with n instances.
- $X = \{x_1, x_2, \dots, x_n\}$: Feature vectors in the dataset.
- $Y = \{y_1, y_2, \dots, y_n\}$: Corresponding labels in the dataset.
- L : Number of layers in the network.
- $\mathbf{W}^{(l)}$: Weight matrix for layer l .
- $\mathbf{b}^{(l)}$: Bias vector for layer l .
- $\mathbf{a}^{(l)}$: Activation vector for layer l .

- $\mathbf{z}^{(l)}$: Pre-activation vector for layer l .
- \mathbf{y}_{pred} : Predicted output vector.
- $\sigma(\cdot)$: Activation function, e.g., sigmoid, ReLU, tanh.
- \mathbf{J} : Loss function, e.g., cross-entropy for classification, mean squared error for regression.

Algorithm:

1. Initialize Weights and Biases: Initialize the weights $\mathbf{W}^{(l)}$ biases $\mathbf{b}^{(l)}$ for each layer l in the network.
2. Forward Propagation:
 - For each instance x_i in the dataset:
 - Set the input layer's activation $\mathbf{a}^{(1)} = x_i$.
 - For $l = 2, 3, \dots, L$:
 - Compute the pre-activation $\mathbf{z}^{(l)} = \mathbf{W}^{(l)} \cdot \mathbf{a}^{(l-1)} + \mathbf{b}^{(l)}$,
 - Compute the activation $\mathbf{a}^{(l)} = \sigma(\mathbf{z}^{(l)})$.
3. Compute Loss: Compute the loss \mathbf{J} based on the predicted output \mathbf{y}_{pred} and the true labels \mathbf{Y} .
4. Backpropagation:
 - Compute the gradient of the loss function with respect to the weights and biases using backpropagation.
 - Update the weights and biases to minimize the loss function using gradient descent or its variants (e.g., Adam, RM-Sprop): $\mathbf{W}^{(l)} := \mathbf{W}^{(l)} - \alpha \frac{\partial \mathbf{J}}{\partial \mathbf{W}^{(l)}}$, $\mathbf{b}^{(l)} := \mathbf{b}^{(l)} - \alpha \frac{\partial \mathbf{J}}{\partial \mathbf{b}^{(l)}}$ where, α is the learning rate.
5. Repeat steps 2-4 until convergence or for a fixed number of iterations.
6. Prediction: For a new input x , perform forward propagation to compute the predicted output \mathbf{y}_{pred} .

Forward propagation is the process of sending input through a network to get the expected outcome. Backpropagation calculates the loss function's gradients with respect to the weights and

biases, which are then utilised to update the parameters. This is done iteratively until the model converges or a predetermined stopping threshold is satisfied. MLPs can have numerous hidden layers (thus the term "multilayer") with various activation functions in each layer. Training an MLP entails determining the best weights and biases to minimise the loss function, which is commonly accomplished through the use of gradient-based optimisation techniques.

4.3.6 Majority Voting

Majority Voting is a popular ensemble learning approach that combines predictions from numerous base classifiers to increase overall prediction accuracy and resilience. It works on the premise that combining the decisions of numerous classifiers can result in higher performance than any single classifier alone. Majority Voting is especially useful when the basis classifiers are varied and produce uncorrelated mistakes [30]. Majority voting is a basic ensemble approach in which the final prediction is chosen by the majority of individual classifier votes. Here's the algorithm with several notations [31, 32, 33, 34].

Notations:

- C : Set of classifiers.
- n : Number of classifiers in C .
- y_i : Predicted label of the i -th classifier.

Algorithm:

1. Predictions: For each instance in the dataset, each classifier in C predicts the label for the instance.
2. Majority Vote:
 - For each instance:
 - Count the number of votes for each class label across all classifiers.
 - Choose the class label with the highest count as the final prediction.
 - In case of ties, break the tie using a pre-defined rule (e.g., randomly, based on class probabilities, etc.).

Classification problems are generally solved by majority vote. The label for each event is predicted individually by each classifier in the ensemble. The final prediction for each instance is made by picking the class label with the highest votes from all classifiers. Majority voting may be applied to any classifier, such as decision trees, logistic regression, support vector machines, and so on. It is a basic but effective ensemble approach that frequently increases classification accuracy, particularly when the various classifiers have varying predictions. Majority voting may alternatively be converted into soft voting, in which the probabilities predicted by each classifier are averaged rather than calculating the hard votes.

5 Experimental Results

5.1 Dataset

This study uses the Mendeley Dataset, a large database of benign and malicious domains retrieved from DNS records [18]. This dataset, specifically designed for supervised machine learning research targeted at distinguishing between dangerous and non-malicious domain names, was methodically produced by pooling publically available DNS logs belonging to both types of domain names. Each domain name in the dataset serves as input, yielding a total of 34 characteristics. These characteristics include direct attributes taken from domain names, such as entropy, the occurrence of uncommon characters, and domain length. Additional information, such as domain creation date, related IP address, open ports, and geolocation, were gathered using data enrichment methods that employed Open Source Intelligence methodologies. This dataset, consisting of 90,000 domain names, is carefully balanced to provide an equal distribution of 50% non-malicious and 50% malicious domains. During the preprocessing step, five characteristics are omitted from both the training and testing datasets: DNSRecordType, CountryCode, RegisteredCountry, RegisteredOrg, and TLD. As a consequence, the final dataset used for experimentation has 29 characteristics that serve as inputs for model training and testing.

5.2 Measures Used for Performance Evaluation of Learning Classifiers on Mendeley DNS Dataset

The usefulness of different machine learning models, including AdaBoost, Logistic Regression, K-Nearest Neighbours, Naive Bayes, Multilayer Perceptron, and Majority Voting, in detecting fraudulent domain names was investigated. The following metrics are used to measure the proposed approach's success in detecting malicious domain names. Figure 3 depicts the confusion matrix, which includes True Positives (TP), True Negatives (TN), False Positives (FP), and False Negatives (FN). These criteria, which provide a measurable evaluation of accuracy, precision, recall, and overall performance, are often used to evaluate the success of classification algorithms [31].

		Actual Class	
		1	0
Predicted Class	1	True Positive	False Positive
	0	False Negative	True Negative

Figure 3: Confusion Matrix for malicious domain names detection

- **Accuracy:** Accuracy is a statistic that measures a classification model's overall efficacy. It expresses the proportion of accurately predicted cases to the total number of examples in the dataset. It is computed using the equation shown below.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (8)$$

- **Precision:** Precision evaluates the accuracy of the model's favourable predictions. It is computed as the ratio of genuine positive predictions to total positive forecasts.

$$Precision = \frac{TP}{TP + FP} \quad (9)$$

- **Recall:** Recall measures the model's ability to properly identify positive cases. It is the ratio of genuine positive forecasts to all instances of positive events. It is computed using the equation shown below.

$$Recall = \frac{TP}{TP + FN} \quad (10)$$

- **F-Measure (F1-Score):** The F1-score, which is the harmonic mean of accuracy and recall, gives a fair evaluation of the model's performance. It is computed using the equation shown below.

$$F1-Measure = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall} \quad (11)$$

5.3 Performance Evaluation of Machine Learning Classifiers on Mendeley DNS Dataset using Full Feature Set

Table 5 compares the performance of each machine learning classifier against a comprehensive feature set consisting of 29 characteristics from the Mendeley DNS Dataset. AdaBoost outperformed other classifiers, with 99.96% detection accuracy, precision, recall, and F-measure. Furthermore, the AdaBoost classifier has fast training and testing durations, taking 3.581 seconds to train and 0.225 seconds to test. Figure 4 and 5 shows the performance evaluation and ROC curve for the classifiers.

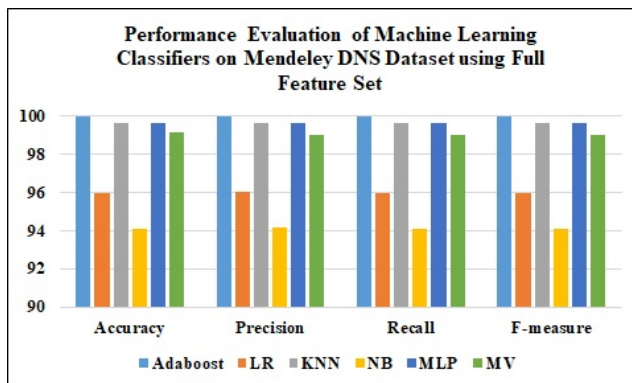


Figure 4: Performance Evaluation of Machine Learning Classifiers on Mendeley DNS Dataset using Full Feature Set

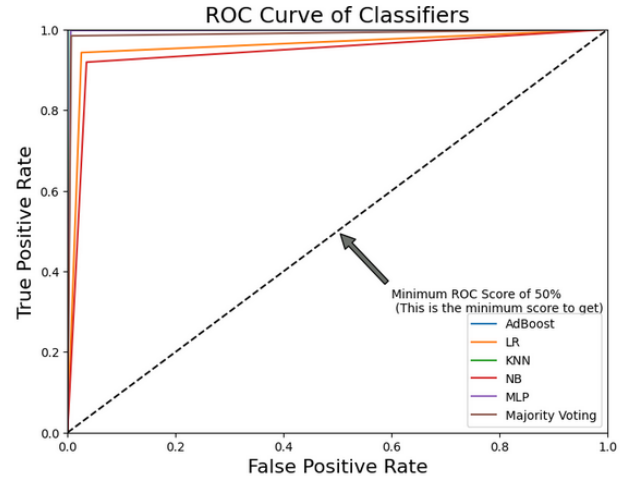


Figure 5: ROC Curve of Machine Learning Classifiers on Mendeley DNS Dataset using Full Feature Set

5.4 Performance Evaluation of Machine Learning Classifiers on Mendeley DNS Dataset using Chi-square (χ^2) Feature Selection Technique

Table 6 demonstrates the performance evaluation of individual machine learning classifiers on the Mendeley DNS Dataset, using 12 features and the Chi-square Feature Selection Method. The AdaBoost, Naive Bayes, Multilayer Perceptron, and Majority Voting classifiers outperformed each other in detecting malicious DNS, as indicated by better accuracy, precision, recall, and F-measure. This increase is accomplished while employing only 12 features as opposed to the complete set of 29 features, resulting in low system overhead. The AdaBoost classifier obtained 99.97% accuracy, precision, recall, and F-measures. Similarly, the KNN classifier achieved 99.67% accuracy, precision, recall, and F-measure. Figure 6 and 7 illustrates the classifiers' performance evaluation and ROC curves.

5.5 Performance Evaluation of Machine Learning Classifiers on Mendeley DNS Dataset using GainRatio Feature Selection Technique

Table 7 compares the performance of individual machine learning classifiers on the Mendeley DNS Dataset using 12 features and the GainRatio Feature Selection Technique.

Classifier	Accuracy (%)	Precision (%)	Recall (%)	F-measure (%)	Train Time (s)	Test Time (s)
AdaBoost	99.96	99.96	99.96	99.96	3.581	0.225
LR	96	96.04	96	96	1.035	0.0103
KNN	99.67	99.67	99.67	99.67	0.0195	6.337
NB	94.06	94.15	94.06	94.06	0.04	0.012
MLP	99.61	99.61	99.61	99.61	92.504	0.032
MV	99.15	99	99	99	106.222	5.003

Table 5: Performance Evaluation of Machine Learning Classifiers on Mendeley DNS Dataset using Full Feature Set

Classifier	Accuracy (%)	Precision (%)	Recall (%)	F-measure (%)	Train Time (s)	Test Time (s)
AdaBoost	99.97	99.97	99.97	99.97	2.73	0.186
LR	93.81	93.97	93.81	93.81	0.757	0.007
KNN	99.67	99.67	99.67	99.67	0.144	2.957
NB	94.1	94.15	94.1	94.1	0.0246	0.007
MLP	99.74	99.74	99.74	99.74	67.467	0.0456
MV	99.44	99.44	99.44	99.44	48.468	5.299

Table 6: Performance Evaluation of Machine Learning Classifiers on Mendeley DNS Dataset using Chi-square (χ^2) Feature Selection Technique

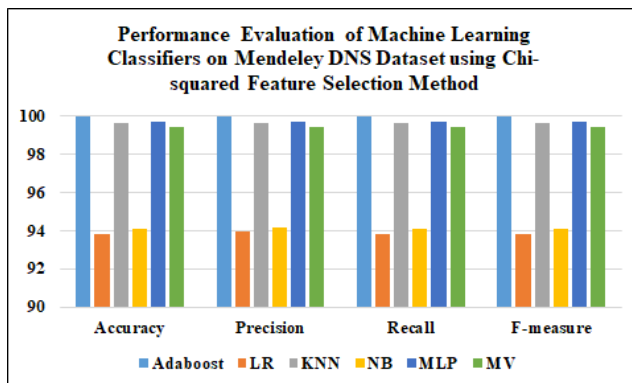


Figure 6: Performance Evaluation of Machine Learning Classifiers on Mendeley DNS Dataset using Chi-squared (χ^2) Feature Selection Technique

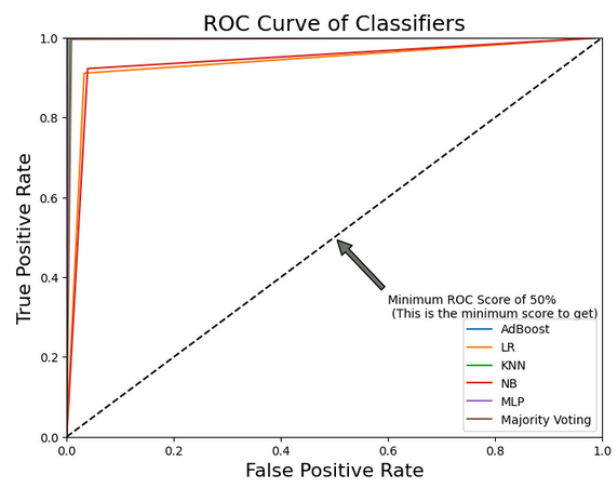


Figure 7: ROC Curve for classifiers using Chi-squared (χ^2) Feature Selection Technique

ture Selection technique. The AdaBoost classifier had the maximum detection accuracy, precision, recall, and F-measure (all at 96.96%). Most classifiers showed gains in training and testing time, with the exception of the Multilayer Perceptron and Majority Voting classifier. Figure 8 and 9 shows the performance evaluation and the ROC curve for the classifiers.

5.6 Performance Evaluation of Machine Learning Classifiers on Mendeley DNS Dataset using Information Gain Feature Selection Technique

Table 8 shows the performance evaluations of individual machine learning classifiers on the Mendeley DNS Dataset, using 12 features and the Information Gain Feature Selection Method. The AdaBoost, KNN, Multilayer Perceptron, and

Classifier	Accuracy (%)	Precision (%)	Recall (%)	F-measure (%)	Train Time (s)	Test Time (s)
AdaBoost	96.96	96.96	96.96	96.96	2.247	0.1889
LR	90.01	90.37	90.01	89.99	0.8859	0.0071
KNN	94.44	94.52	94.44	94.44	0.1465	6.3003
NB	90.62	90.74	90.62	90.61	0.0244	0.00688
MLP	97.9	97.9	97.9	97.9	123.188	0.058
MV	96.55	96.55	96.55	96.55	112.42	6.377

Table 7: Performance Evaluation of Machine Learning Classifiers on Mendeley DNS Dataset using GainRatio Feature Selection Technique

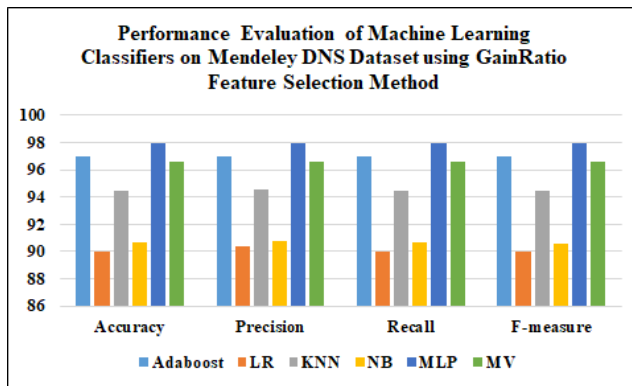


Figure 8: Performance Evaluation of Machine Learning Classifiers on Mendeley DNS Dataset using GainRatio Feature Selection Technique

Majority Voting classifiers all performed better at detecting malicious DNS. This improvement is demonstrated by higher accuracy, precision, recall, and F-measure measures. Interestingly, these classifiers achieve these results while using only 12 characteristics rather than the entire set of 29, reducing system overhead. The AdaBoost classifier performs with 99.97% accuracy, precision, recall, and F-measure. Figure 10 and 11 shows the performance evaluation and the ROC curve for the classifiers.

5.7 Performance Evaluation of Machine Learning Classifiers on Mendeley DNS Dataset using Correlation based Feature Selection (CFS) Technique

Table 9 compares the performance of individual machine learning classifiers on the Mendeley DNS Dataset, using 4 features and the Correlation-based Feature Selection approach. The AdaBoost classifier outperformed others with 94.91% detec-

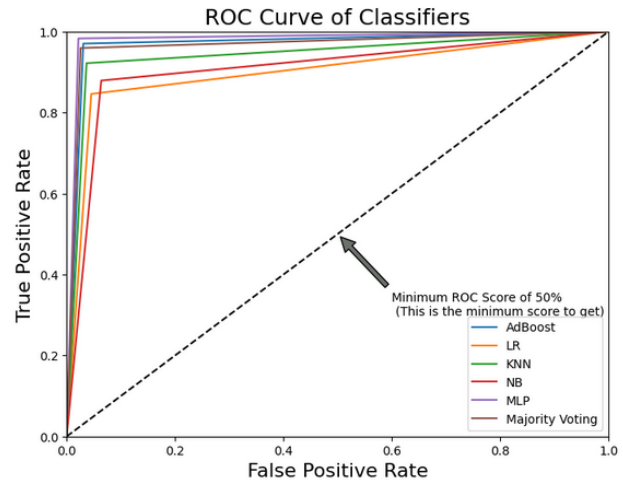


Figure 9: ROC Curve of classifiers using GainRatio Feature Selection Technique

tion accuracy, 95.10% precision, 94.91% recall, and 94.91% F-measure. Except for the Multi-layer Perceptron classifier, most classifiers improved their training and testing times. Figure 12 and 13 shows the performance evaluation and ROC curve for the classifiers.

6 Conclusions

This paper describes a method for identifying harmful domain names that combines feature selection approaches with a majority vote approach. Several approaches, including Chi-square, Gain Ratio, Information Gain, and Correlation-based feature selection, are used to discover the most relevant information for detecting harmful domain names while minimising repetition. The experimental results demonstrate the usefulness of the proposed framework, notably the majority voting strategy, which uses several classifiers to improve detection accuracy. The results show

Classifier	Accuracy (%)	Precision (%)	Recall (%)	F-measure (%)	Train Time (s)	Test Time (s)
AdaBoost	99.97	99.97	99.97	99.97	2.7555	0.19073
LR	92.96	93.07	92.96	92.95	0.472	0.007
KNN	99.69	99.69	99.69	99.69	0.144	2.897
NB	94.04	94.12	94.04	94.04	0.0235	0.006
MLP	99.75	99.75	99.75	99.75	97.564	0.0555
MV	99.4	99.4	99.4	99.4	61.0559	5.264

Table 8: Performance Evaluation of Machine Learning Classifiers on Mendeley DNS Dataset using Information Gain Feature Selection Technique

Classifier	Accuracy (%)	Precision (%)	Recall (%)	F-measure (%)	Train Time (s)	Test Time (s)
AdaBoost	94.91	95.10	94.91	94.91	1.221	0.1704
LR	93.13	93.13	93.13	93.13	0.8254	0.0045
KNN	94.16	94.16	94.16	94.16	0.0513	5.913
NB	91.68	91.68	91.68	91.68	0.0182	0.0041
MLP	95.41	95.52	95.41	95.41	101.938	0.1046
MV	94.73	94.73	94.73	94.73	62.4239	4.751

Table 9: Performance Evaluation of Machine Learning Classifiers on Mendeley DNS Dataset using Correlation based Feature Selection (CFS) Technique

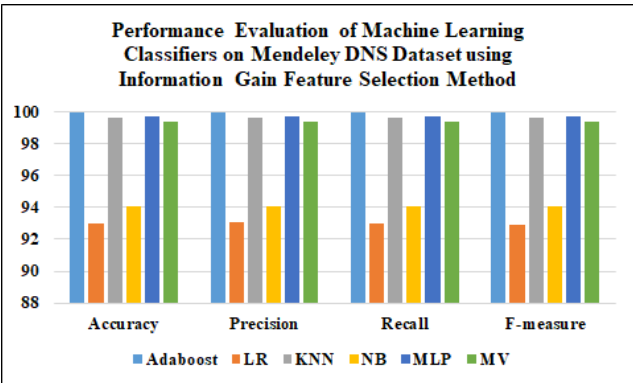


Figure 10: Performance Evaluation of Machine Learning Classifiers on Mendeley DNS Dataset using Information Gain Feature Selection Technique

impressive performance metrics, including accuracy, precision, recall, and F-measure, which are more acceptable compared to individual classifiers, of 99.44%. This was achieved through the combination of Majority Voting and Chi-squared Feature Selection Method, using only 12 domain name features. This demonstrates the framework’s ability to correctly identify malicious domain names while minimising system overhead. Furthermore, a comparison of various feature selection approaches and machine learning classi-

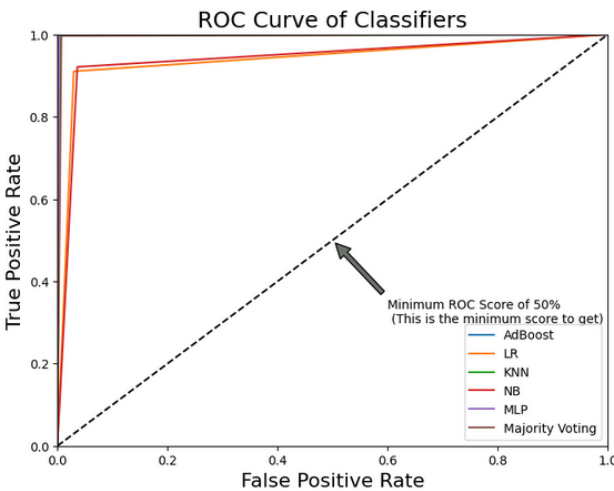


Figure 11: ROC Curve for classifiers using Information Gain Feature Selection Technique

fiers provides useful insights into the best combinations for obtaining higher detection performance. The framework’s adaptability to various forms of malicious domain names is proved by thorough experimentation on real-world datasets.

References

[1] Interisle malicious domain names statistics 4Q 2022. Available online,

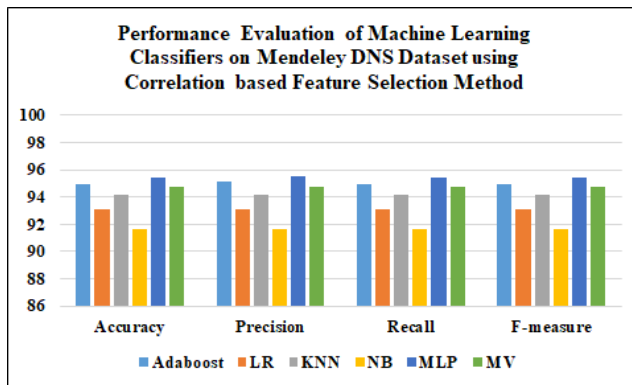


Figure 12: Performance Evaluation of Machine Learning Classifiers on Mendeley DNS Dataset using Correlation based Feature Selection (CFS) Technique

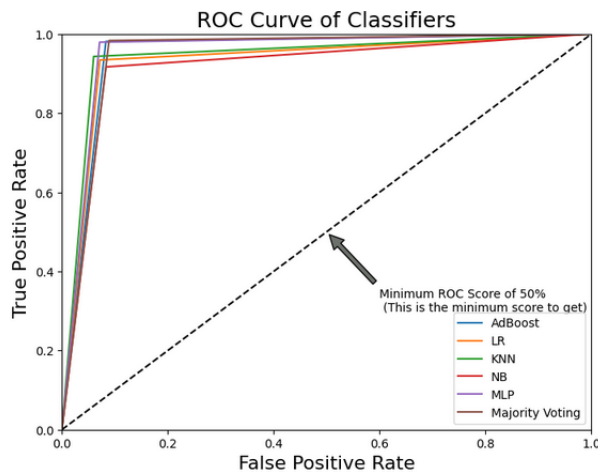


Figure 13: ROC Curve for classifiers using Correlation based Feature Selection (CFS) Technique

<https://www.cybercrimeinfocenter.org/malware-landscape-2023>.

- [2] CSC domain security 2023 report. Available online, <https://www.cscdbs.com/assets/pdfs/2023-Domain-Security-Report.pdf>.
- [3] Zhao, Hong, Zhaobin Chang, Guangbin Bao, and Xiangyan Zeng, Malicious domain names detection algorithm based on N-gram. *Journal of Computer Networks and Communications* 2019.
- [4] Soleymani, Ali, and Fatemeh Arabgol, A novel approach for detecting DGA-based botnets in DNS queries using machine learn-

ing techniques. *Journal of Computer Networks and Communications*, 2021, 1–13.

- [5] Yang, Luhui, Guangjie Liu, Weiwei Liu, Huiwen Bai, Jiangtao Zhai, and Yuewei Dai, Detecting Multielement Algorithmically Generated Domain Names Based on Adaptive Embedding Model, *Security and Communication Networks*, 2021, 1–20.
- [6] Chen, Shaojie, Bo Lang, Yikai Chen, and Chong Xie, Detection of Algorithmically Generated Malicious Domain Names with Feature Fusion of Meaningful Word Segmentation and N-Gram Sequences, *Applied Sciences*, 13, no. 7, 2023, 4406.
- [7] Wagan, Atif Ali, Qianmu Li, Zubair Zaland, Shah Marjan, Dadan Khan Bozdar, Aamir Hussain, Aamir Mehmood Mirza, and Mehmood Baryalai, A Unified Learning Approach for Malicious Domain Name Detection, *Axioms*, 12, no. 5, 2023, 458.
- [8] Bilge, Leyla, Engin Kirda, Christopher Kruegel, and Marco Balduzzi, Exposure: Finding malicious domains using passive DNS analysis, *In Ndss*, pp. 1–17, 2011.
- [9] Fan, Zhaoshan, Qing Wang, Haoran Jiao, Junrong Liu, Zelin Cui, Song Liu, and Yuling Liu, PUMD: a PU learning-based malicious domain detection framework, *Cybersecurity*, 5, no. 1, 2022, 1–22.
- [10] Yang, Luhui, Jiangtao Zhai, Weiwei Liu, Xiaopeng Ji, Huiwen Bai, Guangjie Liu, and Yuewei Dai, Detecting word-based algorithmically generated domains using semantic analysis, *Symmetry*, 11, no. 2, 2019, 176.
- [11] Shi, Yong, Gong Chen, and Juntao Li, Malicious domain name detection based on extreme machine learning, *Neural Processing Letters*, 48, 2018, 1347–1357.
- [12] Fu, Yu, Lu Yu, Oluwakemi Hambolu, Ilker Ozcelik, Benafsh Husain, Jingxuan Sun, Karan Sapra, Dan Du, Christopher Tate Beasley, and Richard R. Brooks, Stealthy domain generation algorithms, *IEEE Transactions on Information Forensics and Security*, 12, no. 6, 2017, 1430–1443.

- [13] Yun, Xiaochun, Ji Huang, Yipeng Wang, Tianning Zang, Yuan Zhou, and Yongzheng Zhang, Khaos: An adversarial neural network DGA with high anti-detection ability, *IEEE transactions on information forensics and security*, 15, 2019,, 2225–2240.
- [14] Yang, Luhui, Guangjie Liu, Yuewei Dai, Jinwei Wang, and Jiangtao Zhai, Detecting stealthy domain generation algorithms using heterogeneous deep neural network framework, *IEEE Access*, 8, 2020,82876–82889.
- [15] Xu, Congyuan, Jizhong Shen, and Xin Du, Detection method of domain names generated by DGAs based on semantic representation and deep neural network, *Computers & Security*, 85, 2019,77–88.
- [16] Vinayakumar, R., K. P. Soman, and Prabaharan Poornachandran, Detecting malicious domain names using deep learning approaches at scale, *Journal of Intelligent & Fuzzy Systems*, 34, no. 3, 2018,1355–1367.
- [17] Yang, Luhui, Guangjie Liu, Jinwei Wang, Jiangtao Zhai, and Yuewei Dai, A semantic element representation model for malicious domain name detection, *Journal of Information Security and Applications*, 66, 2022,103148.
- [18] Marques, Claudio, Benign and malicious domains based on DNS logs, *Mendeley Data*, V5, 2021, doi: 10.17632/623sshkdrz.5.
- [19] Hall M, Frank E, Holmes G, Pfahringer B, Reutemann P, Witten IH, The WEKA data mining software: an update, *ACM SIGKDD explorations newsletter*, 2009, Nov 16, 11(1),10–8.
- [20] Zhai Y, Song W, Liu X, Liu L, Zhao X, A chi-square statistics based feature selection method in text classification, In *2018 IEEE 9th International conference on software engineering and service science (ICSESS)*, 2018, Nov 23,pp. 160–163, IEEE.
- [21] Prasetyo B, Muslim MA, Baroroh N, Evaluation of feature selection using information gain and gain ratio on bank marketing classification using Naïve bayes, In *Journal of physics: conference series*, 2021, Jun 1,Vol. 1918, No. 4, pp. 042153, IOP Publishing.
- [22] Qu K, Xu J, Hou Q, Qu K, Sun Y., Feature selection using Information Gain and decision information in neighborhood decision system, *Applied Soft Computing*, 2023, Mar 1, 136,110100.
- [23] Hall, Mark A., Correlation-based feature selection of discrete and numeric class machine learning, 2000.
- [24] Patil, Dharmaraj R., Tareek M. Patterwar, Vipul D. Punjabi, and Shailendra M. Pardeshi, Detecting Fake Social Media Profiles Using the Majority Voting Approach, *EAI Endorsed Transactions on Scalable Information Systems*,2024.
- [25] Schapire RE., Explaining AdaBoost, In *Empirical Inference: Festschrift in Honor of Vladimir N. Vapnik*, 201,3 Oct 9, pp. 37–52,. Berlin, Heidelberg: Springer Berlin Heidelberg.
- [26] Stoltzfus JC., Logistic regression: a brief primer, *Academic emergency medicine*, 2011, Oct, 18(10), 1099–104.
- [27] Peterson LE., K-nearest neighbor, *Scholarpedia*, 2009, Feb 21, 4(2),1883.
- [28] Rish, Irina., An empirical study of the naive Bayes classifier, In *IJCAI 2001 workshop on empirical methods in artificial intelligence*, vol. 3, no. 22, pp. 41–46. 2001.
- [29] Tang, Jiexiong, Chenwei Deng, and Guang-Bin Huang, Extreme learning machine for multilayer perceptron, *IEEE transactions on neural networks and learning systems*, 27, no. 4, 2015, 809–821.
- [30] Ruta D, Gabrys B., Classifier selection for majority voting, *Information fusion*, 2005, Mar 1, 6(1), 63–81.
- [31] Patil, Dharmaraj R., Tareek M. Patterwar, Vipul D. Punjabi, and Shailendra M. Pardeshi, *Detecting Fake Social Media Profiles Using the Majority Voting Approach*, EAI Endorsed Transactions on Scalable Information Systems, 2024.

- [32] Patil, Dharmaraj R., and Tareek M. Patterwar, *Majority Voting and Feature Selection Based Network Intrusion Detection System*, EAI Endorsed Transactions on Scalable Information Systems 9, no. 6, 2022: e6-e6.
- [33] Patil, Dharmaraj R., *Fake news detection using majority voting technique*, arXiv preprint arXiv:2203.09936, 2022.
- [34] Patil, Dharmaraj R., and Jayantro B. Patil, *Malicious URLs detection using decision tree classifiers and majority voting technique*, Cybernetics and Information Technologies 18, no. 1, 2018: 11-29.
- [35] Sokolova M, Lapalme G., A systematic analysis of performance measures for classification tasks, *Information processing & management*, 2009, Jul 1, 45(4), 427–37.