Malicious Domain Name Detection Using ML Algorithms

Lamis Alshehri, Samah Alajmani
Dept. of Cybersecurity, Taif University, Taif, Saudi Arabia

***Abstract*—With the ever-increasing rate of cyber threats, especially through malicious domain names, the need for their effective detection and prevention becomes very urgent. This study mainly investigates the classification of domain names into either benign or malicious classes based on DNS logs using machine learning. We evaluated five strong ML models: XGBoost, LightGBM, CatBoost, Stacking, and Voting Classifier, in an effort to obtain high accuracy, F1 score, AUC, recall, and precision. The challenge in that direction is to achieve a very good solution, without using deep learning techniques for low computational cost. Moreover, this project has an obligation to upgrade the cybersecurity landscape by embedding the best-performing model into the DNS firewall to enable protection against harmful domains in real time. Our dataset was collected and curated to include 90,000 domain names, including an equal number of safe and harmful, respectively, extracting 34 features from DNS logs and further enriched using publicly available data.**

***Keywords*—DNS Security; machine learning; malicious domain detection; XGBoost; LightGBM; CatBoost**

I. Introduction

With the advancement of the digital age, the use of the internet has become extremely common for communication, the exchange of important information, and even for commerce. This has led to an increased demand for cybersecurity and the search for precise security mechanisms that can be implemented and utilized. Among the many common threats, the Domain Name System (DNS) is one of the crucial elements in the internet's infrastructure, as it converts domain names into IP addresses. However, it lacks appropriate protection mechanisms, which allows cybercriminals to exploit these vulnerabilities, helping them spread malware, conduct phishing attacks, or gain unauthorized access to data on servers. Therefore, there is an urgent need for methods that achieve a balance between efficiency, accuracy, and the ability to perform in real-time [1].

The ever-evolving nature of cybersecurity demands continuous upgrading to match newer, sophisticated modes of attack. The Domain Name System (DNS) is an important target for attackers due to the significant losses it can cause. Therefore, any breach of the security of the DNS affects the reliability of the internet greatly, which underscores the importance of securing this system. In the event of any compromise to its foundational structure, institutions will suffer losses and customers will lose their privacy, leading to customer dissatisfaction as well as legal implications and other significant issues. The primary goal of the DNS when it was designed was to provide a scalable and available domain name resolution service, but at that time, security aspects were not adequately emphasized, resulting in many security vulnerabilities that could turn lives upside down globally if exploited by attackers. This issue also calls for an interesting junction of technological advancement, real-time Threat Intelligence, with pragmatic implementation of solutions [2].

This project investigates the capability of ML models in identifying and classifying domain names as either benign or malicious based on DNS log data. Advanced ML algorithms such as XGBoost, LightGBM, CatBoost, Stacking, and Voting Classifiers will be used to develop an efficient cybersecurity solution which is computationally effective. The current approach focuses on lightweight ML models rather than deep learning methods, which require a huge amount of computational resources. The best performance model will be integrated into a DNS firewall for better security of the network. This system not only addresses current limitations in DNS security but also provides a scalable and cost-effective approach for future cybersecurity challenges [3].

The main objective of the research is to propose a lightweight, accurate, and efficient system for malicious domain name detection based on DNS logs. This research also aims to develop ML models that classify domain names with high precision, recall, and F1 scores. It also focuses on designing a non-deep learning system to provide computational cost efficiency and enable real-time applications. In addition, a comparison was made between the performance of five deep learning-based machine learning models to find the best approach. This research also relies on model optimization, using a 34-dimensional feature space derived from enriched DNS features using DNS records to enhance the latest model outcomes by leveraging a high-dimensional dataset. One of the important goals is to deploy the best-performing model on a DNS firewall for implementation in real-world situations and also to ensure that the proposed solution has the capacity to scale to large volumes of DNS traffic in wide-scale environments.

Below are the key contributions the research that makes to the field of cybersecurity and ML-based threat detection:

- Model Performance - Extended Comparison: Performances of XGBoost, LightGBM, CatBoost, Stacking, and Voting Classifier will be presented in this work through extensive testing, providing the best methodology for DNS threat detection.
- Rich Feature Utilization: The study uses a dataset of 34 features extracted from DNS logs, ensuring that the ML

models are adequately informed to carry out the classification with a high degree of accuracy. Features like these will provide in-depth and subtle particulars about the behavior of the domain names.

- Lightweight Solution: By not being dependent on deep learning approaches, the system is computationally light and reachable even by organizations lacking extensive computing resources. This entails a wider applicability across various sectors with different technical capabilities.
- Practical Integration: The integration of the best-selected model into a DNS firewall allows real-time defense against malicious domain names and closes the gap between theory and practice. Experimental insights become, at this stage, an actionable tool.
- Balanced Dataset: The research is based on a dataset containing an equal number of benign and malicious domain names, thus allowing for impartial model evaluation. This will add to the credibility and reliability of the results found from this work.

The rest of this paper is organized as follows: Section II reviews the related work. Section III details the methodology. Section IV presents the results and analysis. Section V provides the discussion, highlighting the key findings and their practical implications. Finally, Section VI concludes the paper.

II. Related Work

Many researchers have proposed various methods for detecting malicious domain names in the literature.

Wagan et al. [4] have developed a single, unifying method for discovering malicious domain names utilizing both numerical and textual information. Traditional DNS firewalls rely on lists of blacklisted maligned domains, but such lists cannot respond to new, emerging malignants. Traditional machine learning approaches have aided in enhancing detections but have not utilized both numerical and textual information of a domain name in its full capacity. To mitigate this, they have developed a deep model with a Hybrid Feed Forward Network (FFN) for numerical and a Long Short-Term Memory (LSTM) for textual information. Features extracted through both numerical and textual information are consolidated in a single, unifying format, and then utilized for classification. They trained a model over a 90,000-domain name corpus and demonstrated its performance to outperform six baseline approaches in terms of accuracy, precision, recall, and F1-score.

Ren et al. [5] proposed a deep model for Domain Generation Algorithm (DGA) domain detection via an integration of an attention mechanism with a combination of Convolutional Neural Networks (CNNs) and Bidirectional Long Short-Term Memory (BiLSTM) networks. With both locality in character structures and long-term relations in domain names in consideration, and leveraging the use of an attention mechanism for prioritization of salient features, proposed model, namely, ATT-CNN-BiLSTM, can accurately discriminate between malignant and innocent domains, in contrast to traditional DGA detection approaches, which have a problem with wordlist-based DGA domains. Experimental evaluation confirms that ATT-CNN-BiLSTM achieves an F1-score of 98.79% in DGA detection, outperforming traditional machine and deep learning approaches. In addition, the model has high generalizability, and thus, proves effective in processing previously unfamiliar DGA families.

Luo et al. [6] proposed a deep learning system for malicious URL identification, utilizing a composite neural network (Comp-block) and an auto-encoder for feature extraction and classification, respectively. First, URLs go through irrelevant information deletion and tokenization of structure. An auto-encoder then transforms URLs into vector representations, and representations go through a deep model constructed with Convolutional Neural Networks (CNN) for anomalous behavior analysis. Manual feature selection is not utilized in the proposed scheme, and it is efficient in contrast with traditional rule-based approaches. Experimental evaluation with the HTTP CSIC 2010 and a custom dataset revealed that the system achieves high accuracy (98.20%) and detects anomalous URLs with low false alarm, outpacing traditional approaches for detection.

Marques et al. [7] proposed a real-time ML-enforced DNS firewall for real-time malignant request domain filtering. Unlike traditional firewalls, utilizing a blocklist, potentially excluding recently generated new-malicious domains, their model employs supervised ML algorithms for distinguishing between malignant and innocent DNS queries.

The system processes DNS logs with 34 key feature extraction and OSINT-enriched feature extraction. Various algorithms for machine learning, including Decision Trees (CART), SVM, Logistic Regression, and KNN, have been compared with a 90,000 record dataset. Experimental performance showed that CART performed best with an accuracy of 96% and a rapid classification time, and can, therefore, be used for real-time filtering of DNS. In this work, it is established that ML-powered DNS firewalls can effectively enhance cybersecurity through efficient detection and filtering out of malice domains over traditional approaches.

Thain et al. [8] proposed a machine learning-based approach to detect malicious domains on the Internet by analyzing domain names and traffic passing through DNS. They used important people information .Then they used techniques such as Random Forest, XGBoost and AdaBoost to find out if the site is malicious or not. After several experiments, they found that the system can identify malicious sites with an accuracy of up to (92.7%) even if the data is small. Then they combined it with semantic analysis and the system became more effective than traditional methods on blacklist.

Samad et al. [9] presented an intelligent system for detecting malicious websites on the internet. The system relies on natural language processing (NLP) in URLs and also the content of web pages. Techniques are employed to better understand words, such as n-grams (which means a set of words), which assist the system in making accurate decisions. The system uses seven mathematical methods, such as Random Forest and XGBoost, to determine whether a website is malicious. After several experiments, they discovered that the system, by integrating the content of pages and URLs, is significantly better than older methods.

D. Ma & Wu. [10] proposed a new method for detecting malicious domain names using a specific intelligent model called (VAE). The main objective is to improve the detection of (DGA) families, which are defined as random domains that are difficult to detect.

The method begins by processing the domain names, where the data is cleaned of impurities and unimportant details. After that, a technique called (Word2Vec) is used to convert the words into vectors for better understanding by the system.

Subsequently, the vectors are input into the (VAE) model, which adjusts itself using backpropagation. The damage probability is then calculated, and the domain is classified as harmful or benign based on a threshold classifier.

Experiments have shown that this method outperforms traditional methods in detecting (DGA) families.

Zhao et al. [11] have proposed an algorithm to detect harmful domain names based on the statistical features of URLs, using a decision tree classifier to enhance detection accuracy. Their method relies on extracting characteristics such as length, special characters, and character distribution to distinguish between legitimate and harmful domains. A decision tree was used to classify the domains based on these features, and the model achieved an accuracy of 90.31% in detecting harmful domains. The study shows that analyzing URL features significantly aids in accurately classifying domains and reduces the need for pre-labeled data.

Compared to previous research, in our study, we made complementary contributions to some similar papers by comparing five machine learning algorithms: XGBoost, LightGBM, CatBoost, Stacking, and Voting Classifier. This makes them easier to interpret and clearer, and with a lighter weight than deep learning techniques. We relied on feature selection techniques such as ANOVA F-value and SelectKBest to identify the most influential features, which reduces the dimensionality of the data and improves the model's performance. We also conducted a comprehensive study of a set of features that includes characteristics related to email security, such as SPF, DKIM, and DMARC records, as they enhance the ability to detect domains that target phishing and email attacks. Additionally, we propose adding the best model for detecting harmful domains to be integrated into a prototype for a DNS firewall.

III. Methodology

It depicts the suggested architecture for detecting malicious domain names. The framework consists of many steps, including dataset and preprocessing, feature engineering, model creation, and the use of ensemble learning like voting classifier and stacking classifier techniques. Each phase substantially improves the overall effectiveness of the malicious domain name detection system. Fig. 1 shows proposed framework.

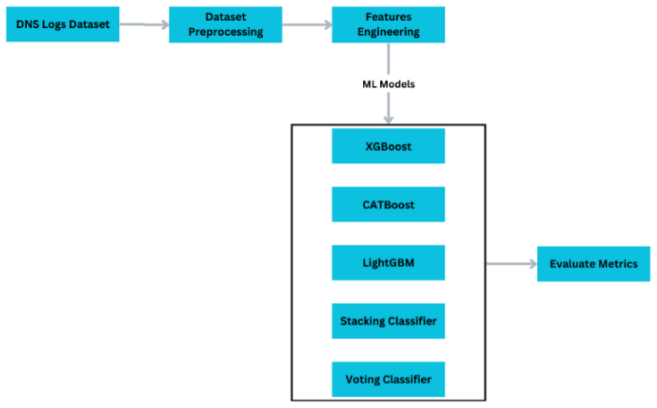

Fig. 1. Proposed framework.

A. Dataset

This study uses the Mendeley Dataset (Table I) that has been collected and processed by Marques et al. which includes both benign and malicious domains retrieved from DNS logs [12]. 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.

TABLE I. Dataset Features with Description and Data Types

Feature Name	Description	Type	Count
Domain	Baseline DNS used to enrich data, e.g., derive features	int64	90000
DNSRecordType	DNS record type queried	object	90000
MXDnsResponse	The response from a DNS request for the record type MX	bool	90000
TXTDnsResponse	The response from a DNS request for the record type TXT	bool	90000
HasSPFInfo	If the DNS response has Sender Policy Framework attribute	bool	90000
HasDkimInfo	If the DNS response has Domain Keys Identified Email attribute	bool	90000
HasDmarcInfo	If the DNS response has Domain-Based Message Authentication	bool	90000
IP	The IP address for the domain	int64	90000
DomainInAlexaDB	If the domain is registered in the Alexa DB	bool	90000
CommonPorts	If the domain is available on common ports	bool	90000
CountryCode	The country code associated with the IP of the domain	object	60948
RegisteredCountry	The country code from domain registration (WHOIS)	object	12226

Feature Name	Description	Type	Count
CreationDate	The creation date of the domain (WHOIS)	int64	90000
LastUpdateDate	The last update date of the domain (WHOIS)	int64	90000
ASN	The Autonomous System Number for the domain	int64	90000
HttpResponseCode	The HTTP/HTTPS response status code for the domain	int64	90000
RegisteredOrg	The organization name from domain registration (WHOIS)	object	54609
SubdomainNumber	The number of subdomains for the domain	int64	90000
Entropy	The Shannon entropy of the domain name	int64	90000
EntropyOfSubDomains	The mean entropy of the subdomains	int64	90000
StrangeCharacters	The number of non-alphabetic characters	int64	90000
TLD	The Top-Level Domain for the domain	object	89830
IpReputation	The result of the blocklisted search for the IP	bool	90000
DomainReputation	The result of the blocklisted search for the domain	bool	90000
ConsoantRatio	The ratio of consonant characters in the domain	float64	90000
NumericRatio	The ratio of numeric characters in the domain	float64	90000
SpecialCharRatio	The ratio of special characters in the domain	float64	90000
VowelRatio	The ratio of vowel characters in the domain	float64	90000
ConsoantSequence	Max number of consecutive consonants in the domain	int64	90000
VowelSequence	Max number of consecutive vowels in the domain	int64	90000
NumericSequence	Max number of consecutive numeric characters in the domain	int64	90000
SpecialCharSequence	Max number of consecutive special characters in the domain	int64	90000
DomainLength	The length of the domain	int64	90000
Class	The class of the domain (0 = malicious, 1 = non-malicious)	int64	90000

In this study, 34 features were carefully selected based on their significance in distinguishing between malicious and benign domains.

Behavioral features such as Entropy, NumericRatio, and SpecialCharRatio measure the degree of randomness within a domain name. Higher values of these features typically indicate that the domain was automatically generated using algorithms like Domain Generation Algorithms (DGA), which are commonly utilized in malicious activities.

Reputation and registration features, including IpReputation and DomainReputation, verify whether the domain or its associated IP address is listed in known blacklists. Similarly, CountryCode and ASN provide geographical and network-related context, as certain regions and service providers are statistically linked to hosting malicious domains.

Structural features such as DomainLength, SubdomainNumber, and StrangeCharacters assess the composition of the domain name itself. Malicious domains often adopt long, complex names or incorporate unusual symbols to mimic legitimate websites while evading detection.

Additionally, Email-related Security features like HasSPFInfo, HasDkimInfo, and HasDmarcInfo examine the existence of standard email protection protocols. Malicious domains used in phishing or spam messages typically lack these protective protocols.

Finally, accessibility and response behavior features such as CommonPorts and HttpResponseCode evaluate how the domain responds to connection attempts which analyze the domain's response when attempting to connect. Malicious domains may use unusual ports or return response codes (e.g., 404 or 503), signaling potentially harmful intent or unreliable behavior.

These combined features provide a comprehensive view that enhances the model's ability to accurately classify domains based on both static attributes and dynamic behavior.

Fig. 2, shows the distribution of the target variable Class, which indicates whether a domain is malicious or benign. The x-axis represents the two classes, and the y-axis represents the count of domains in each class.

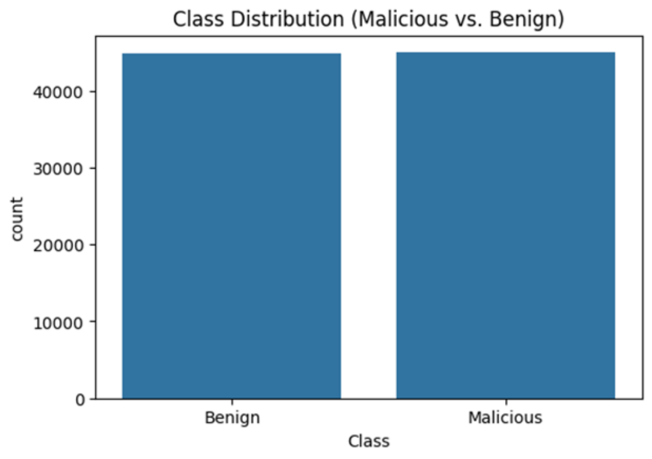

Fig. 2. Class distribution (malicious vs. benign).

The data is evenly distributed between the two classes, meaning there is no significant class imbalance. This is beneficial for training a machine learning model, as it ensures that the model does not become biased toward one class.

Fig. 3, show set of histograms visualizes the distributions of numerical features such as Entropy, DomainLength, SpecialCharSequence, and others. Each histogram shows the frequency of values for a specific feature.

Fig. 4, show these count plots display the distribution of categorical features such as HasSPFInfo, HasDkimInfo, DNSRecordType, and others. Each plot is further divided by the Class (malicious vs. benign) to show how the feature values differ between the two classes.

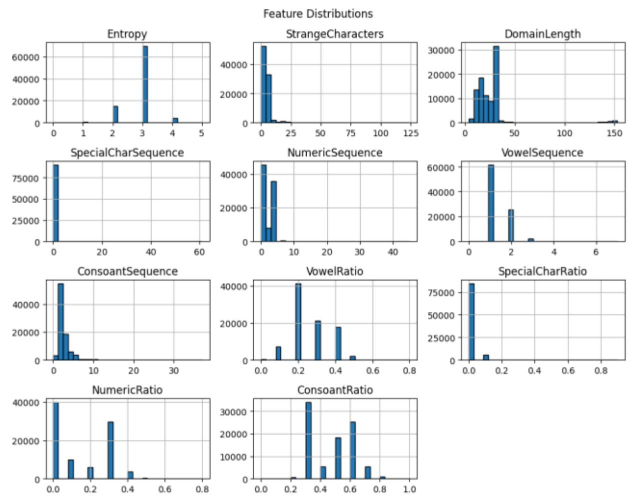
Fig. 3. Distributions of numerical features.

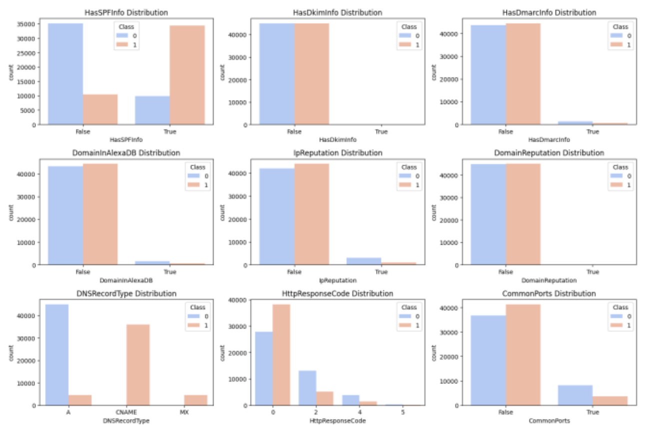
Fig. 4. Distributions of categorical features by class.

The feature DNSRecordType shows that malicious domains predominantly have values of "A" or "CNAME," while benign domains have another value. This suggests that DNSRecordType could be a strong predictor of maliciousness but may also introduce bias so we will drop it. The skewed distribution of DNSRecordType indicates that it may negatively impact the model's performance and should be removed.

B. Dataset Preprocessing

Data preparation involves several steps like data cleaning and data transformation. We removed 170 rows containing null values while performing the data cleaning process. We removed the columns Domain, DNSRecordType, CountryCode, RegisteredOrg, and RegisteredCountry in the data transformation process since they were not helpful for further processing in machine learning algorithms.

For boolean-type and text columns, we employed the Label Encoder of the scikit-learn library. Boolean values were converted to integers (0 and 1). All integer values were also normalized using the Standard Scaler normalization technique, which is common in scientific research. This method scales the data to have a mean of 0 and a standard deviation of 1, thus features are centered and scaled on variance.

Though other normalization techniques exist, Standard Scaler was utilized since it scales the features well without being greatly affected by extreme outliers, making it suitable for this dataset.

Standardization equation:

$$z = \frac{x-\mu}{\sigma} \quad (1)$$

with mean equation:

$$\mu = \frac{1}{N}\sum_{i=1}^{N} (x_i) \quad (2)$$

and standard deviation equation:

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N} (x_i - \mu)^2} \quad (3)$$

The correlation heatmap that shows in Fig. 5. visualizes the pairwise correlation coefficients between all numerical features in the dataset. Each cell in the heatmap represents the correlation between two features, with values ranging from -1 to 1. A value of 1 indicates a perfect positive correlation, -1 indicates a perfect negative correlation, and 0 indicates no correlation. The heatmap reveals that certain features, such as NumericRatio, StrangeCharacters, and ConsoantRatio, are strongly correlated with the target variable Class. These features are likely to be important for predicting whether a domain is malicious or benign.

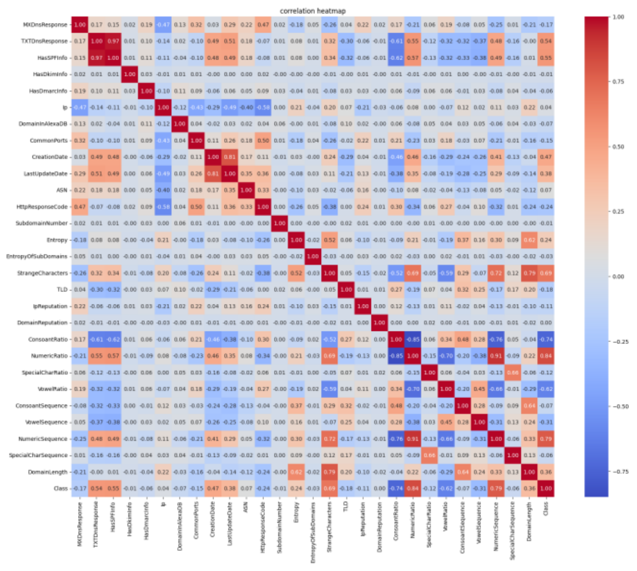
Fig. 5. Correlation heatmap.

C. Features Engineering

Feature selection strategies are used to determine the most discriminating characteristics. To determine the relevance of features to the classification job, as it involves selecting and transforming the most relevant features to improve model performance. In this study, we employed the SelectKBest method with the f_classif scoring function to identify the top 20 features that contribute the most to the predictive power of the model. The f_classif function computes the ANOVA F-value between each feature and the target variable, which helps in selecting features with the strongest statistical relationship to the target. The selected features include a mix of categorical and numerical variables, such as MXDnsResponse, TXTDnsResponse,HasSPFInfo, DomainInAlexaDB, CommonPorts, CreationDate, LastUpdateDate, ASN, HttpResponseCode, Entropy, StrangeCharacters, TLD, IpReputation, ConsoantRatio, NumericRatio, SpecialCharRatio, VowelRatio, VowelSequence, NumericSequence, and DomainLength. These features were chosen based on their ability to distinguish between malicious and benign domains effectively. By reducing the feature space to the most informative variables, we not only improve model efficiency but also mitigate the risk of overfitting, ensuring that the model generalizes well to unseen data.

D. Machine Learning Models

1) XGBoost: XGBoost is an advanced framework based on gradient tree boosting for solving large-scale machine learning problems efficiently. It is highly reputed for its predictive performance and training speed and has been consistently topping Kaggle competitions. The basic concept of the algorithm is to add decision trees iteratively, constantly splitting features to expand and enhance the model. You actually learn a new function to fit the last predicted residual when each time you add a tree [13]. Letting x_i be the input, y_i be true label and z_i be the 'raw prediction' before the sigmoid function, according to study [14], the objective function of the XGB model is:

Standardization equation:

$$L^{(t)} = \sum_{i=1}^{n} l\left(y_i, Z_i^{(t-1)} + f_t(x_i)\right) + \Omega(f_t) + c \qquad (4)$$

Where $l(.,.) denotes the loss function, t$ stands for the t th tree, Ω penalizes the complexity of the model, $\Omega(f_t)$ represents the penalty term of regularization, and c is constant.

The second-order Taylor expansion is:

$$f(x + \Delta x) \approx f(x) + f'(x)\Delta x + 1/2 f''(x)\Delta x^2 \qquad (5)$$

Taking “(2)” into “(1)”, we can get

$$L^{(t)} \approx \sum_{i=1}^{n} \left[l\left(y_i + Z_i^{(t-1)}\right) + g_i f_t(x_i) + \frac{1}{2} h_i \left(f_t(x_i)\right)^2 \right] + \Omega(f_t) + c \qquad (6)$$

where $g_i = \partial L / \partial z_i$, and $h_i = \partial^2 L / \partial z_i^2$. Removing the constant terms, we can obtain the following simplified objective at step t.

$$L^{(t)} \approx \sum_{i=1}^{n} \left[g_i f_t(x_i) + \frac{1}{2} h_i \left(f_i(x_i)\right)^2 \right] + \Omega(f_t) \qquad (7)$$

In this objective function, g_i and h_i are required for fitting the XGB model.

For binary classification problems, the default loss function of XGB is the cross entropy (CE) loss:

$$L = -\sum_{i=1}^{n} \left[y_i \log(\hat{y}_i) - (1 - y_i) \log(1 - \hat{y}_i) \right] \qquad (8)$$

In Eq. (5) , $\hat{y}_i = 1/[1 + \exp(-z_i)]$, that is sigmoid is selected as activation. Therefore, we can get:

$$\frac{\partial \hat{y}_i}{\partial z_i} = \hat{y}_i (1 - \hat{y}_i) \qquad (9)$$

2) LightGBM: LightGBM is a machine learning algorithm that relies on Gradient Boosting Decision Tree (GBDT). It operates by iteratively training several weak classifiers and combining them into a strong classifier capable of performing classification and regression tasks. Compared to traditional GBDT algorithms, LightGBM offers significant advantages such as high training speed, low memory consumption, and effective prediction capability. Additionally, LightGBM has outperformed other algorithms in terms of efficiency [15].

The primary objective function in LightGBM includes two significant components: the loss function and the regularization term, which controls model complexity:

$$\mathcal{L} = \sum_{i=1}^{N} l(y_i, \hat{y}_i) + \sum_{t=1}^{T} \Omega(f_t) \qquad (10)$$

Where $l(y_i, \hat{y}_i)$ is the loss function (for example, log loss for classification), and $\Omega(f_t)$ is the regularization term for preventing overfitting. LightGBM minimizes this function using a second-order Taylor expansion, which approximates the loss function using the first-order and second-order derivatives:

$$\mathcal{L}^{(t)} \approx \sum_{i=1}^{N} \left[g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i) \right] + \Omega(f_t) \qquad (11)$$

where g_i and h_i are the gradient and Hessian of the loss function, respectively. Under this formulation, it is possible to perform more accurate and efficient optimization than with traditional gradient boosting methods.

One of the primary advantages of LightGBM is the leaf-wise growth strategy, which grows the tree by selecting the leaf with the maximum loss reduction instead of growing the tree level-wise. The split gain is computed as:

$$\Gamma\alpha\iota\nu = \frac{1}{2}\left(\frac{G_L^2}{H_L} + \frac{G_R^2}{H_R} - \frac{(G_L + G_R)^2}{H_L + H_R} \right) - \gamma \qquad (12)$$

where, G_L, G_R and H_L, H_R are the sums of gradients and Hessians for the left and right child nodes, respectively, and γ is a regularization parameter.

In order to further boost training efficiency, LightGBM uses histogram-based feature binning, which discretizes continuous features into a given number of bins:

$$\text{bin}(x) = \left\lfloor (x - x_{min}) \times \frac{\beta\iota\nu\_\chi o\upsilon\nu\tau}{x_{max} - x_{min}} \right\rfloor \qquad (13)$$

This reduces computation time while making the best splits more easily discovered without any loss in accuracy. Overall, LightGBM’s new techniques make it one of the fastest and most scalable boosting algorithms available, with uses in everything from fraud detection to recommendation systems.

3) CatBoost: CatBoost (Categorical Boosting) is a gradient boosting algorithm developed specifically to handle categorical features with high quality and efficiency. The CatBoost algorithm uses Ordered Target Statistics instead of One-Hot Encoding, as its method computes category values based only on previous data points, rather than on the entire dataset at once. This reduces the chances of overfitting and improves computational performance. The CatBoost algorithm operates like existing boosting algorithms but excels when there is a mix of categorical and continuous data [16].

CatBoost's objective function is in the gradient boosting general form, with a loss function and a regularization term:

$$\mathcal{L} = \sum_{i=1}^{N} l(y_i, \hat{y}_i) + \sum_{t=1}^{T} \Omega(f_t) \tag{14}$$

where $l(y_i, \hat{y}_i)$ is the loss function (e.g., log loss for classification, squared error for regression), and $\Omega(f_t)$ is a regularization term for controlling model complexity. CatBoost minimizes this function using ordered boosting, which avoids overfitting caused by target leakage during training.

For efficiency, CatBoost utilizes a symmetric tree structure, i.e., all splits at a specific depth are created simultaneously in all the branches. This ensures balanced trees and prevents bias toward certain features, which leads to better generalization. The optimal split is computed based on the gain formula:

$$\Gamma\alpha\iota\nu = \frac{1}{2}\left(\frac{G_L^2}{H_L} + \frac{G_R^2}{H_R} - \frac{(G_L+G_R)^2}{H_L+H_R}\right) - \lambda \tag{15}$$

where G_L, G_R and H_L, H_R are the sums of gradients and Hessians for the left and right child nodes, respectively, and λ is a regularization parameter.

CatBoost possesses a significant edge in handling categorical data, removing overfitting, and accelerating training without a loss in accuracy. Ordered boosting, symmetric trees, and novel categorical encoding make CatBoost highly effective in practical machine learning applications.

4) Stacking classifier: The Stacking Classifier is an ensemble technique in machine learning that uses a stacking method aimed at combining several different base models to create a more accurate and powerful model. The Stacking Classifier trains a set of base models on the same dataset to obtain different predictions specific to each model. It then trains a meta-classifier on the results of the base models to merge them in the best way. Each base model can be given a different weight based on its performance or accuracy, ultimately testing the stacking classifier to produce the final prediction that is most accurate. We use stacking because it combines different models, resulting in a final model that is more accurate, better at generalizing, and less susceptible to error or bias towards a single model [17].

The objective function of a stacking classifier contains two layers. In the first layer, we have MMM base models, each of which is trained on the original data set:

$$\hat{y}_m = f_m(X), m = 1,2,\dots,M \tag{16}$$

where f_m represents each base model, and X represents the input feature set. The models predict, and these predictions are new features for the second layer, where a meta-classifier f_{meta} is trained:

$$\hat{y} = f_{meta}(\hat{y}_1, \hat{y}_2, \dots, \hat{y}_M) \tag{17}$$

The final prediction $\hat{y}$ is found by combining all the outputs of the base models in the best possible manner. The meta-classifier is usually a simple model (e.g., logistic regression or decision tree) that learns to weight and combine the predictions of the base models to get optimal performance.

To prevent overfitting and improve generalization, stacking typically employs K-fold cross-validation, where base models are trained on different folds of the data, and their predictions on unseen data are used to train the meta-classifier:

$$\hat{y}_m^{(i)} = f_m\left(X^{(i)}\right), \forall i \in \{1,2,\dots,K\} \tag{18}$$

where $X^{(i)}$ is the training fold in the K-fold cross-validation process. In this way, the meta-classifier is trained on out-of-fold predictions, and the models are not allowed to memorize the training data and be biased.

Overall, stacking is a powerful technique that improves accuracy by ensembling multiple models. It is computationally demanding and requires careful tuning of base models and the meta-classifier to prevent overfitting. Despite these drawbacks, stacking is a widely used technique for high-performance predictive modeling in a variety of domains, including finance, healthcare, and recommendation systems.

5) Voting classifier: Voting is a popular ensemble learning method that combines predictions of several base classifiers to improve general prediction accuracy and strength. It is based on the premise that the collective decision of numerous classifiers can result in improved performance than that of any single classifier. Majority Voting is especially useful when the basis classifiers are heterogeneous and commit uncorrelated errors. Majority voting is a straightforward ensemble approach in which the final prediction is determined by the majority of the individual classifier votes [18] [19].

The Voting Classifier is an ensemble learning technique that combines a number of machine learning models to improve the accuracy and stability of predictions. Unlike stacking, which learns a meta-classifier over the base model predictions, voting combines predictions from a number of classifiers directly through hard voting or soft voting. The method is particularly useful when the base models are heterogeneous, capturing different nuances of the data.

For hard voting, the final prediction is decided by a majority vote of the base classifiers:

$$\hat{y} = \text{mode}(\hat{y}_1, \hat{y}_2, \dots, \hat{y}_M) \tag{19}$$

where $\hat{y}_m$ is the m-th model's prediction, and the majority class is selected as the final output.

For soft voting, the ultimate prediction is taken from the average of the predicted probabilities of all the base models:

$$\hat{y} = \arg\max \sum_{m=1}^{M} w_m P_m(y \mid X) \tag{20}$$

where:

$P_m(y \mid X)$ is the predicted probability of class y by model m.

w_m is an optional weight assigned to each model based on its importance.

Soft voting is generally better than hard voting, especially if the base models are well-calibrated, as it allows the classifier to take into account the confidence levels of the different models.

The Voting Classifier is particularly useful when you need to ensemble models with complementary strengths. For example, decision trees can learn complicated interactions in data, logistic regression can ensure stability, and gradient boosting models can provide good generalization.

In this study, we used a soft voting ensemble with scikit-learn's Voting Classifier. Soft voting takes the predicted probabilities from each classifier and selects the class with the highest average probability, which performs better than hard voting. To build the ensemble, we initialized six LightGBM classifiers with different learning rates (0.1, 0.09, 0.2, 0.08, 0.3, and 0.07). The learning rates were changed to introduce diversity in the base models because altering hyperparameters can reduce correlation between the classifiers' errors. The classifiers were then passed to a Voting Classifier with the estimators parameter, which takes a list of tuples containing the model names and instances. We set the voting parameter to 'soft' for voting based on probabilities. This approach takes the best of each model and removes their worst parts, resulting in a stronger and more accurate ensemble model.

IV. RESULT AND ANALYSIS

In this section, we present the results obtained from classifying DNS logs into malicious and benign categories using various machine learning algorithms. The evaluation of each model is based on accuracy, precision, recall, F1-score, and AUC. Additionally, confusion matrices and ROC curves provide further insights into model performance.

1) Evaluation metrics: In this research, we used standard classification metrics such as Accuracy, Precision, Recall, F1-Score, and Area Under the ROC Curve (AUC). We selected these metrics because they provide a comprehensive evaluation of the performance of the chosen models to facilitate the assessment of which performs better than others.

Despite the common reporting of accuracy, it can be misleading in cases where unbalanced datasets are included, where the number of benign domains is greater than that of malicious ones. This is because models can achieve high accuracy simply by predicting the majority class in the group. Therefore, we integrated Precision and Recall.

Precision measures the proportion of domains predicted to be malicious that are indeed malicious, which is very important for reducing false positive results and avoiding the blocking of legitimate domains. Recall reflects the model's effectiveness in accurately identifying harmful domains, contributing to the reduction of false negative results.

Also, we used F1-Score because it gives us a consistent average between precision and recall, making the positive and negative false results balanced. Finally, we added AUC-ROC because it is considered an independent measure of the model's ability to discriminate between the two classes. AUC is very important for understanding overall performance across different classification thresholds, especially when there is a dataset containing varied class distributions.

2) Performance evaluation: Our model's predictions can result in four possible outcomes:

- True Positive (TP): A malicious domain name is correctly identified as malicious.
- True Negative (TN): A non-malicious domain name is correctly identified as non-malicious.
- False Negative (FN): A malicious domain name is incorrectly classified as non-malicious.
- False Positive (FP): A non-malicious domain name is incorrectly classified as malicious.

Using these outcomes, we can calculate key performance evaluation metrics such as accuracy, recall, precision, and F1-score, as outlined below.

Accuracy is one of the most straightforward metrics for evaluating the performance of a binary classification model. It represents the percentage of correctly classified samples out of the total samples. Using the previously introduced notation, accuracy is defined in the equation as follows:

$$\text{Accuracy} = \frac{\text{TP+TN}}{\text{TP+TN+FP+FN}} \tag{21}$$

As another measure of classifier performance, precision assesses the accuracy of positive predictions. It is the proportion of correctly predicted positive instances to all predicted positive instances. Precision is defined by the formula:

$$\text{Precision} = \frac{TP}{TP+FP} \tag{22}$$

Precision is always combined with a measure called recall because precision measurement would be very high for models which predict few positives. Recall specifies the proportion of positive examples that are correctly identified by the classifier, given by the formula:

$$\text{Recall} = \frac{TP}{TP+FN} \tag{23}$$

The F1-Score is the harmonic mean of precision and recall, as defined by Equation (number x). A large F1-Score can only be obtained if both recall and precision are high.

$$\text{F1} - \text{Score} = \frac{2\times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{24}$$

The XGBoost classifier demonstrated exceptional performance, achieving an accuracy of 98.58% with an AUC of 0.9991. The confusion matrix reveals that the model correctly classified 8889 malicious and 8821 benign instances while misclassifying 160 benign samples as malicious (false positives) and 96 malicious samples as benign (false negatives). The low false negative rate suggests that the model is highly effective in detecting malicious domains, minimizing the risk of overlooking threats, as illustrated in Fig. 6 and Fig. 7.

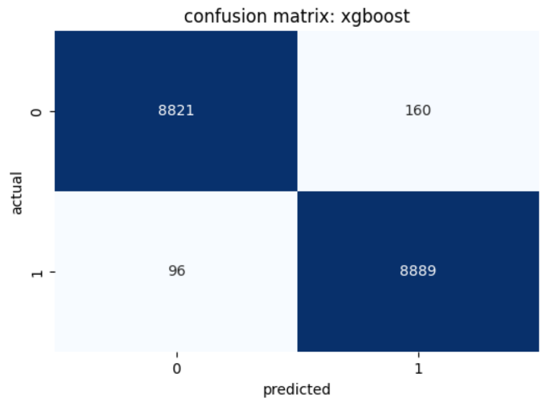

Fig. 6. Confusion matrices illustrating the classification performance of XGBoost on DNS log data.

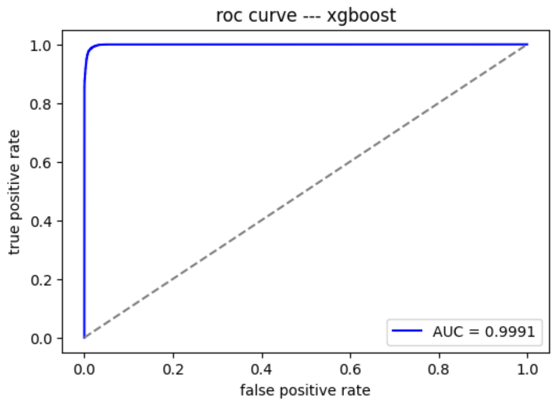

Fig. 7. ROC curves depicting the AUC scores for XGBoost.

Similarly, the LightGBM classifier produced comparable results, attaining an accuracy of 98.51% and an AUC of 0.9990. Although LightGBM performed slightly below XGBoost, the marginal difference in AUC suggests that both models are highly effective. The confusion matrix shows 8889 correctly classified malicious instances and 8821 correctly classified benign instances. However, it misclassified 160 benign samples as malicious and 96 malicious samples as benign. These results indicate that LightGBM performs slightly below XGBoost in distinguishing between the two classes but remains a strong candidate for DNS log classification, as shown in Fig. 8 and Fig. 9.

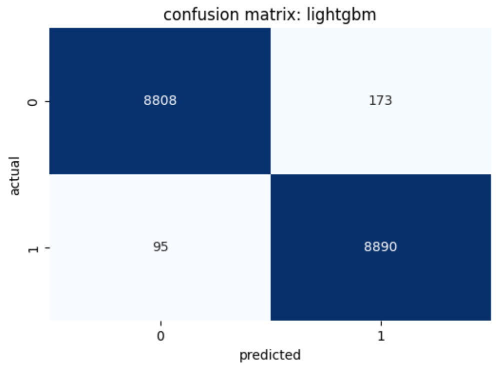

Fig. 8. Confusion matrices illustrating the classification performance of LightGBM on DNS log data.

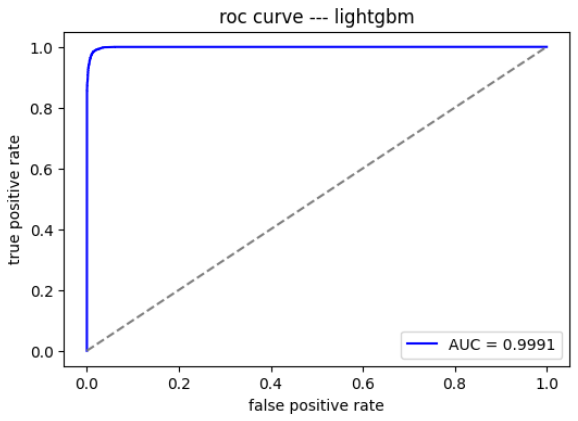

Fig. 9. ROC Curves depicting the AUC scores for LightGBM.

The CatBoost classifier emerged as the best-performing model, achieving the highest accuracy of 98.71% and an AUC of 0.9992. The confusion matrix highlights its superior classification capability, with 8901 correctly identified malicious domains and 8833 correctly identified benign domains. Additionally, it recorded 148 false positives and 84 false negatives, the lowest among all models. The reduced number of false negatives implies that CatBoost is the most effective in correctly identifying malicious domains, making it a highly reliable option, as shown in Fig. 10 and Fig. 11.

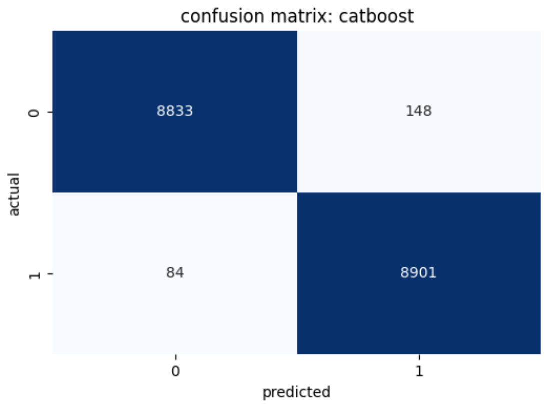

Fig. 10. Confusion matrices illustrating the classification performance of CatBoost on DNS log data.

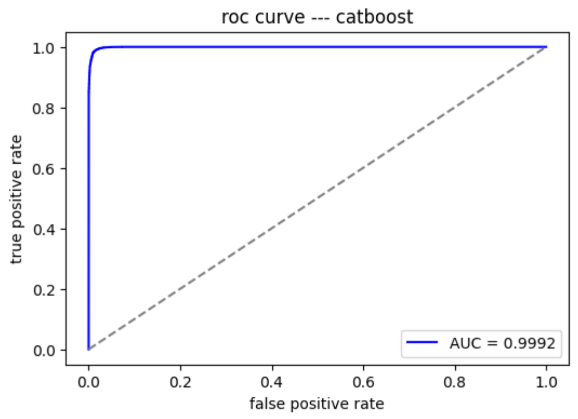

Fig. 11. ROC curves depicting the AUC scores for CatBoost.

The Voting Classifier, which combines multiple models, achieved an accuracy of 98.59% with an AUC of 0.9991. Its confusion matrix indicates that 8901 malicious and 8812 benign domains were correctly classified, while 169 benign samples were incorrectly flagged as malicious, and 84 malicious samples were misclassified as benign. Although it performed well, the slightly higher false positive rate compared to CatBoost suggests that it may generate more false alerts, as illustrated in Fig. 12 and Fig. 13.

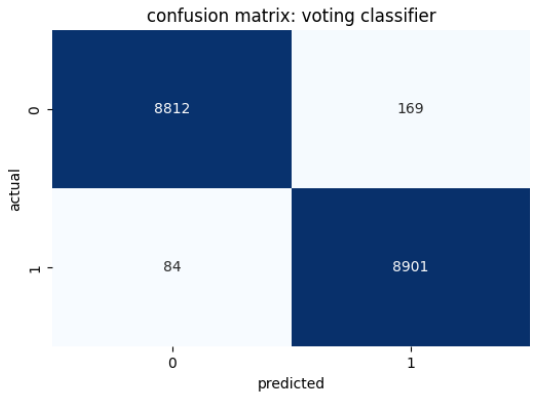

Fig. 12. Confusion matrices illustrating the classification performance of Voting Classifier on DNS log data.

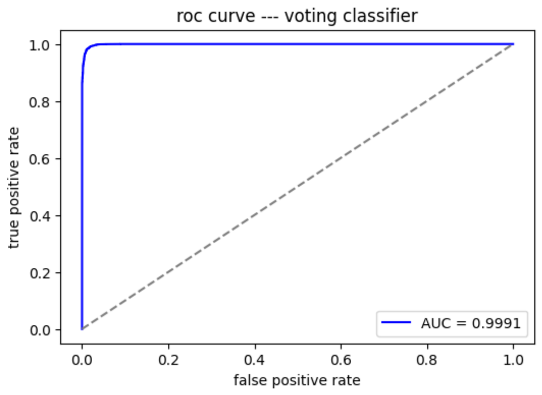

Fig. 13. ROC Curves depicting the AUC scores for voting classifier.

The Stacking Classifier attained an accuracy of 98.53% and an AUC of 0.9979. The confusion matrix analysis shows 8880 correctly classified malicious domains and 8822 correctly classified benign domains. It produced 159 false positives and 105 false negatives, indicating a higher false negative rate compared to the other models. This suggests that the Stacking Classifier, while still effective, may not be the optimal choice for minimizing undetected threats, as shown in Fig. 14 and Fig. 15.

A comparative analysis of the models highlights that all classifiers performed exceptionally well, with accuracy surpassing 98%. CatBoost emerged as the best-performing model, delivering the highest accuracy and AUC, making it the most suitable choice for DNS log classification. These findings suggest that ensemble methods such as CatBoost and XGBoost are highly effective in detecting malicious domains, reinforcing their potential for real-world cybersecurity applications. Table II shows comparison table of models.

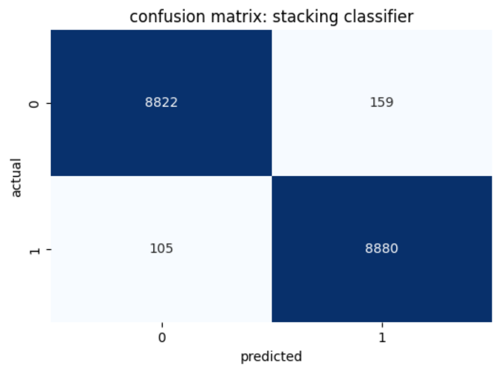

Fig. 14. Confusion matrices illustrating the classification performance of Stacking Classifier on DNS log data.

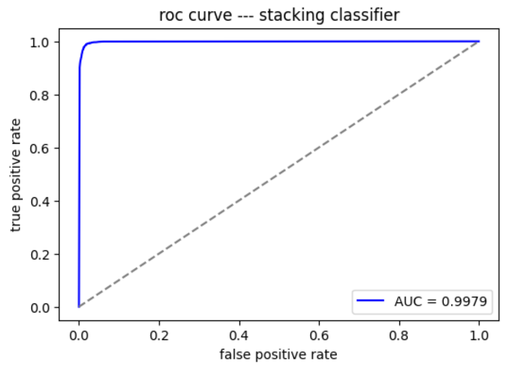

Fig. 15. ROC Curves depicting the AUC scores for stacking classifier.

TABLE II. COMPARISON TABLE OF MODELS

Model	Accuracy	Precision	Recall	F1-Score	AUC
XGBoost	0.9858	0.9858	0.9857	0.9857	0.9991
LightGBM	0.9851	0.9851	0.9850	0.9850	0.9991
CatBoost	0.9871	0.9871	0.9870	0.9871	0.9992
Voting Classifier	0.9859	0.9859	0.9859	0.9859	0.9991
Stacking Classifier	0.9853	0.9853	0.9853	0.9853	0.9979

V. DISCUSSION AND SUMMARY

The malicious domain detection systems rely on the quality of feature selection, the performance of machine learning models, and their applicability in a real-world environment. In this research, five machine learning models (XGBoost, LightGBM, CatBoost, Stacking, and Voting Classifier) were evaluated using a balanced dataset containing 90,000 domain names, with 34 features extracted from DNS records.

The results showed that the CatBoost model outperformed all other models, achieving the highest accuracy of 98.71% and the best performance in other metrics such as F1-score and AUC-ROC. This demonstrates CatBoost's high capability in handling data and dealing with categorical data effectively while reducing bias during training. In comparison, the performance of both XGBoost and LightGBM was similar, as they achieved accuracy exceeding 98.5%, indicating that boosting techniques are effective in detecting malicious domains.

On the other hand, both the Voting Classifier and Stacking Classifier showed strong performance, but without significant improvement compared to the other models. This indicates that combining models did not result in a clear and noticeable enhancement, which may be one reason for the similarity of the basic models' errors in classification.

A. Error Analysis

Despite the high performance of the models, there are challenges that should be taken into consideration.

1) High false positive rate: Some safe domains have been classified as malicious domains, especially those that contain unfamiliar but legitimate names.

2) Errors in detecting malicious domains (false negatives): Despite the small percentage, some harmful domains have not been discovered, especially those that use obfuscation techniques or domain names that are similar to legitimate sites.

3) Impact of data features: The correlation analysis showed that some of the features used, such as DNSRecordType, might be biased, making their removal important to enhance the overall model performance.

B. Practical Implementation

This research may serve as an impetus for practical steps towards enhancing network security through malicious domain detection systems. However, there are still some challenges that are important to address when applying the model in real-world environments.

1) Adapting to emerging threats: The model can be improved through continuous data updates and the addition of new and important features based on the developments in cyberattack technologies.

2) Real-time performance analysis: Although this research focuses on integration with the DNS firewall, studying the impact of the model on network performance and response time may be necessary.

3) Scalability: When applying the model in large-scale systems, improving resource consumption without affecting network performance to ensure quick responsiveness may be essential.

C. Summary

The results of this study show that boosting models such as CatBoost and XGBoost can achieve high performance in detecting malicious domains without the need for deep learning techniques. However, integrating these models into real-world security systems requires important additional improvements to enhance performance and ensure security, such as reducing false positives, adapting to new or emerging threats, and analyzing real-time performance.

This research can be further developed in the future by using hybrid models that combine machine learning and deep learning, along with improving data processing techniques and feature analysis to increase classification accuracy and reduce biases.

VI. Conclusion and Future Work

In this project, we explored the use of machine learning in classifying domain names into benign or malicious based on DNS log data. By comparing several machine learning algorithms—XGBoost, LightGBM, CatBoost, Stacking, and Voting Classifiers—we identified CatBoost as the best-performing model with the highest accuracy, precision, recall, and AUC score. The results indicate that ML-based DNS security solutions can be effective at preventing and detecting cyber threats in real time.

Our solution provides a lightweight and computationally less intensive alternative to deep learning-based models, making it a feasible solution for real-world deployment in resource-constrained environments. By integrating the best-performing model in a DNS firewall, we enhance cybersecurity defenses by reducing the risk of malicious domains, which lowers the risk of phishing, malware spread, and data breaches.

Future work can be oriented in the direction of optimizing feature engineering methods, incorporating real-time threat intelligence, and using diversified datasets for the better generalization of the model. Additionally, the fusion of deep learning models and traditional ML models can be incorporated to obtain a hybrid solution that can provide a balance between efficiency and accuracy.

This project contributes to the developing field of AI-driven cybersecurity, offering an affordable and scalable solution to the evolving nature of cyber threats. As cybersecurity and machine learning advance, the implementation of intelligent DNS security solutions will be critical in safeguarding digital infrastructure.

References

[1] Toorn, O. V., Müller, M. C., Dickinson, S., Hesselman, C., Sperotto, A., & Rijswijk-Deij, R. V. (2022). Addressing the challenges of modern DNS: A comprehensive tutorial. *Computer Science Review, 45,* 100469. https://doi.org/10.1016/j.cosrev.2022.100469.

[2] Jalalzai, M. H., Shahid, W. B., & Iqbal, M. M. W. (2015). DNS security challenges and best practices to deploy secure DNS with digital signatures. *2015 12th International Bhurban Conference on Applied Sciences and Technology (IBCAST),* 280–285. https://doi.org/10.1109/IBCAST.2015.7058517.

[3] Marques, C., Malta, S., & Magalhães, J. (2021). DNS firewall based on machine learning. *Future Internet, 13*(12), Article 309. https://doi.org/10.3390/fi13120309

[4] Wagan, A. A., Li, Q., Zaland, Z., Marjan, S., Bozdar, D. K., Hussain, A., Mirza, A. M., & Baryalai, M. (2023). A Unified Learning Approach for Malicious Domain Name Detection. *Axioms*, 12(5), Article 5. https://doi.org/10.3390/axioms12050458

[5] Ren, F., Jiang, Z., Wang, X., & Liu, J. (2020). A DGA domain names detection modeling method based on integrating an attention mechanism and deep neural network. *Cybersecurity, 3*(1), Article 4. https://doi.org/10.1186/s42400-020-00046-6

[6] Luo, C., Su, S., Sun, Y., Tan, Q., Han, M., & Tian, Z. (2020). A Convolution-Based System for Malicious URLs Detection. *Computers, Materials & Continua,* 62(1), 399–411. https://doi.org/10.32604/cmc.2020.06507

[7] Marques, C., Malta, S., & Magalhães, J. (2021). DNS Firewall Based on Machine Learning. *Future Internet,* 13(12), Article 12. https://doi.org/10.3390/fi13120309

[8] Thein, T. T., Shiraishi, Y., & Morii, M. (2023). Malicious Domain Detection Based on Decision Tree. *IEICE Transactions on Information*

and Systems, E106.D(9), 1490–1494. https://doi.org/10.1587/transinf.2022OFL0002

[9] Samad, S. R. A., Ganesan, P., Al-Kaabi, A. S., Rajasekaran, J., M, S., & Basha, P. S. (2024). Automated Detection of Malevolent Domains in Cyberspace Using Natural Language Processing and Machine Learning. *International Journal of Advanced Computer Science and Applications, 15(10).* https://doi.org/10.14569/IJACSA.2024.0151036

[10] Ma, D., & Wu, X. (2024). A malicious domain name detection method based on variational autoencoder. *2024 IEEE 2nd International Conference on Control, Electronics and Computer Technology*, 1206–1210. https://doi.org/10.1109/ICCECT60629.2024.10545732

[11] Zhao, H., Chen, Z., & Yan, R. (2022). Malicious domain names detection algorithm based on statistical features of URLs. *2022 IEEE 25th International Conference on Computer Supported Cooperative Work in Design*, 11–16. https://doi.org/10.1109/CSCWD54268.2022.9776264

[12] Marques, C. (2021). Benign and malicious domains based on DNS logs (*Version 5*) [Data set]. *Mendeley Data*. https://doi.org/10.17632/623sshkdrz.5

[13] He, S., Li, B., Peng, H., Xin, J., & Zhang, E. (2021). An effective cost-sensitive XGBoost method for malicious URLs detection in imbalanced dataset. *IEEE Access, 9*, 93089–93096. https://doi.org/10.1109/ACCESS.2021.3093094

[14] Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 785–794). https://doi.org/10.1145/2939672.2939785

[15] Cai, Z., Huang, H., Sun, G., Li, Z., & Ouyang, C. (2023). Advancing predictive models: Unveiling LightGBM machine learning for data analysis. *2023 4th International Conference on Computer, Big Data and Artificial Intelligence (ICCBD+AI)*, 109–112. https://doi.org/10.1109/ICCBD-AI62252.2023.00027.

[16] Malashin, I., Tynchenko, V., Gantimurov, A., Nelyub, V., & Borodulin, A. (2025). Boosting-Based Machine Learning Applications in Polymer Science: A Review. *Polymers, 17*(4), 499. https://doi.org/10.3390/polym17040499.

[17] Reddy, S. N., Krishna, D. V., Asritha, I., & Charitha, L. (2024). Ensemble stacking classifier for cardiovascular risk prediction. *2024 International Conference on Inventive Computation Technologies (ICICT)*, 534–540. https://doi.org/10.1109/ICICT60155.2024.10544597.

[18] Ruta, D., & Gabrys, B. (2005). Classifier selection for majority voting. *Information Fusion, 6*(1), 63–81. https://doi.org/10.1016/j.inffus.2004.04.008.

[19] Patil, D. R., Pattewar, T. M., Punjabi, V. D., & Pardeshi, S. M. (n.d.). Detecting fake social media profiles using the majority voting approach. *EAI Endorsed Transactions on Scalable Information Systems.* https://doi.org/10.4108/eetsis.4264.